CSc 566, Computer Security
Research Reports

Junxiao Shi, Sara Saleem,
Mathias Gibbens, Harsha vardhan Rajendran,
Balaji Prasad T.K, Nupur Maheshwari,
Paul Mueller, Babak Yadegari,
Matthew Ward, Paul Jennas II,
Keith Fligg, Genevieve Max,
Sam Martin and Mark Tokutomi,
Eric DeBusschere, Mike McCambridge

April 2012

TABLE OF CONTENTS

Phishing

Junxiao Shi, Sara Saleem

1 Introduction

Phishing is a form of social engineering in which an attacker, also known as a phisher, attempts to fraudulently retrieve legitimate users' confidential or sensitive credentials by mimicking electronic communications from a trustworthy or public organization in an automated fashion [19]. The word "phishing" appeared around 1995, when Internet scammers were using email lures to "fish" for passwords and financial information from the sea of Internet users; "ph" is a common hacker replacement of "f", which comes from the original form of hacking, "phreaking" on telephone switches during 1960s [16]. Early phishers copied the code from the AOL website and crafted pages that looked like they were a part of AOL, and sent spoofed emails or instant messages with a link to this fake web page, asking potential victims to reveal their passwords [4].

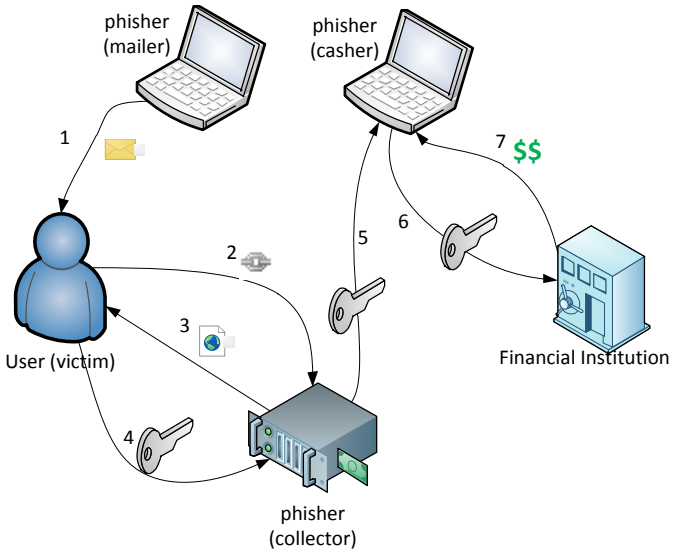


Figure 1: Phishing information flow

A complete phishing attack involves three roles of phishers. Firstly, *mailers* send out a large number of fraudulent emails (usually through botnets), which direct users to fraudulent websites. Secondly, *collectors* set up fraudulent websites (usually hosted on compromised machines), which actively prompt users to provide confidential information. Finally, *cashers* use the confidential information to achieve a pay-out. Monetary exchanges often occur between those phishers. The information flow is shown in Figure 1.

Before delving further into phishing it's important to clarify what is not phishing. Nigerian 419 scam (sending emails to trick recipients into giving money to the scammer) and Internet auction

fraud (non-delivery, misrepresentation, fee stacking, or selling stolen goods) are not considered phishing since they don't involve obtaining users' credentials [20].

The latest statistics reveal that banks and financial institutions along with the social media and gaming sites continue to be the main focus of phishers. Some loyalty programs are also becoming popular among phishers because with them phishers can not only breech the financial information of victim but also use existing reward points as currency. U.S. remains the largest host of phishing, accounting for 43% of phishing sites reported in January 2012. Next was Germany at 6%, followed by Australia, Spain, Brazil, Canada, the U.K., France, Netherlands, and Russia [29]. A study of demographic factors suggests that women are more susceptible to phishing than men and users between the ages of 18 and 25 are more susceptible to phishing than other age groups [30]. Phishing attacks that initially target general consumers are now evolving to include high-profile targets, aiming to steal intellectual property, corporate secrets, and sensitive information concerning national security.

2 Types of Phishing

Phishing has spread beyond email to include VOIP, SMS, instant messaging, social networking sites, and even multiplayer games. Below are some major categories of phishing.

2.1 Clone Phishing

In this type phisher creates a cloned email. He does this by getting information such as content and recipient addresses from a legitimate email which was delivered previously, then he sends the same email with links replaced by malicious ones. He also employs address spoofing so that the email appears to be from the original sender. The email can claim to be a re-send of the original or an updated version as a trapping strategy [31].

2.2 Spear Phishing

Spear phishing targets at a specific group. So instead of casting out thousands of emails randomly, spear phishers target selected groups of people with something in common, for example people from the same organization [28].

Spear phishing is also being used against high-level targets, in a type of attack called "whaling". For example, in 2008, several CEOs in the U.S. were sent a fake subpoena along with an attachment that would install malware when viewed [24]. Victims of spear phishing attacks in late 2010 and early 2011 include the Australian Prime Minister's office, the Canadian government, the Epsilon mailing list service, HBGary Federal, and Oak Ridge National Laboratory [18].

2.3 Phone Phishing

This type of phishing refers to messages that claim to be from a bank asking users to dial a phone number regarding problems with their bank accounts. Traditional phone equipment has dedicated lines, so Voice over IP, being easy to manipulate, becomes a good choice for the phisher. Once the phone number, owned by the phisher and provided by a VoIP service, is dialed, voice prompts tell the caller to enter her account numbers and PIN. Caller ID spoofing, which is not prohibited by law, can be used along with this so that the call appears to be from a trusted source [1].

3 Phishing Techniques and Countermeasures

Various techniques are developed to conduct phishing attacks and make them less suspicious. Email spoofing is used to make fraudulent emails appear to be from legitimate senders, so that recipients are more likely to believe in the message and take actions according to its instructions. Web spoofing makes forged websites look similar to legitimate ones, so that users would enter confidential information into it. Pharming attracts traffic to those forged websites. Malware are installed into victims' computers to collect information directly or aid other techniques. PDF documents, which supports scripting and fillable forms, are also used for phishing.

3.1 Email Spoofing

A spoofed email is one that claims to be originating from one source when it was actually sent from another [19]. Email spoofing is a common phishing technique in which a phisher sends spoofed emails, with the sender address and other parts of the email header altered, in order to deceive recipients.

Spoofed emails usually appear to be from a website or financial institution that the recipient may have business with, so that an unsuspecting recipient would probably take actions as instructed by the email contents, such as:

- reply the email with their credit card number
- click on the link labelled as "view my statement", and enter the password when the (forged) website prompts for it
- open an attached PDF form, and enter confidential information into the form (Section 3.5)

3.1.1 Sending a spoofed email

On a sendmail-enabled UNIX system, one line of command is all you need to send a spoofed email that appears to be from Twitter:

```
cat body.htm | mail -a 'From: Twitter <support@twitter.com>' -a '
    Content-Type: text/html' -s 'Reset your Twitter password'
    victim@example.net
```

The file body.htm contains the mail contents in HTML format. The result is shown in Figure 2.

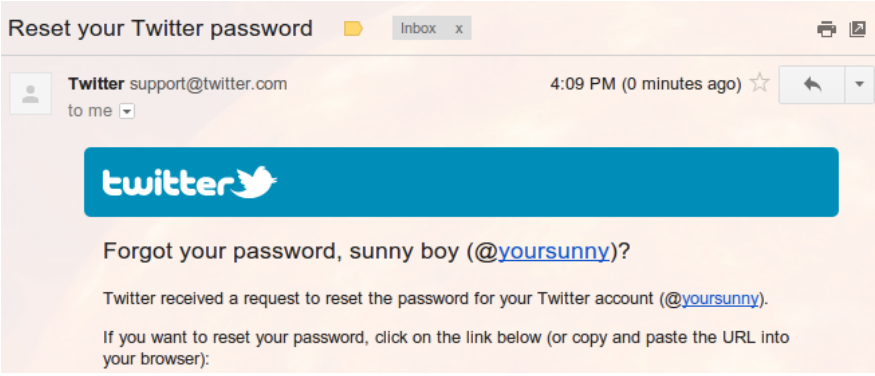


Figure 2: Fake Twitter password reset email received in Gmail

3.1.2 Why it's possible

Simple Mail Transfer Protocol [21] is the Internet standard protocol used for electronic mails. Its objective is to transfer mail reliably and efficiently, but core SMTP doesn't provide any authentication. An important feature of SMTP is its capability to transport mail across multiple networks, referred to as "SMTP mail relaying". Basically, receiving and relaying SMTP servers need to trust the upstream server; so it is feasible for a malicious user to construct spoofed messages, and talk with receiving or relaying SMTP servers directly to deliver such a message.

As RFC 5321 [21] suggests, SMTP mail inherently cannot be authenticated at the transport level; real mail security lies only in end-to-end methods involving the message bodies, such as Pretty Good Privacy (PGP) and Multipurpose Internet Mail Extensions (S/MIME). However, there is a high cost to deploy those digital signature based countermeasures, because users are reluctant to install an additional piece of software, and they don't have enough knowledge on how to manage the trust.

3.1.3 SPF

Sender Policy Framework (SPF) is an open standard specifying a technical method to prevent sender address forgery [25]. Since most SMTP servers are mutually-TCP-addressible hosts on the public Internet, receiving and relaying SMTP servers are able to see the IP address of the sending host. SPFv1 protects the *envelope sender address*, the `HELO` domain and the `MAIL FROM` address, by verifying sender IP addresses: SPFv1 allows the owner of a domain to specify a list of IP addresses that are allowed to send emails from their domain, and publish this information in the domain's DNS zone; a receiving server may query DNS to check whether the message comes from one of those whitelisted addresses.

For example, cs.arizona.edu publishes the following SPF record:

```
v=spf1 a:gandalf.email.arizona.edu a:frodo.email.arizona.edu a:pacer.emai
l.arizona.edu a:gremlin.email.arizona.edu a:optima.cs.arizona.edu ~all
```

This SPF record lists 5 hostnames, and these hosts are allowed to send emails on behalf of @cs.arizona.edu; "~all" disallows any other hosts to send emails from this domain.

3.1.4 DKIM

DomainKeys Identified Mail (DKIM) allows an organization to take responsibility for transmitting a message in a way that can be verified by a recipient. The author, the originating sending site, an intermediary, or one of their agents can attach digital signatures onto a message [17]. The message headers and body, including the originator address (the `From` header field), are signed. The `DKIM-Signature` header field includes the signature, the signing domain, and information about how to retrieve the public key.

A DKIM signature generated by Gmail looks like:

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
  d=gmail.com; s=gamma;
  h=domainkey-signature:mime-version:received:received:in-reply-to
   :references:date:message-id:subject:from:to:content-type;
  bh=rdk+ZKX52H558uYXf2No2gW+cp8RKaZBZwyOM+LufnE=;
  b=dw0s8c2uuBIqY8msh1266XyGlTDxYGwIBmuVPpkMEUGh2mrhWaUwSWYUnOKHSh
   vlwVBTiLGRQ8t8KYk1XdMveBnE3iaXlOGiGKlQLqIQjyd+sxbc8OSGHxcO05BpO
```

```
3Egb/pf+i8m9iktEjN4PPhLKsyiniNO8vy8LqC33zjyiVw=
```

The signing domain publishes public keys as TXT records in their DNS zone. To verify this signature, a receiving server may query DNS name gamma._domainkey.gmail.com (constructed from tags "s" and "d" of the signature) and get a TXT record such as:

```
k=rsa; p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDIhyR3oItOy22ZOaBrIVe9m/iM
E3RqOJeasANSpg2YTHTYV+Xtp4xwf5gTjCmHQEMOs0qYu0FYiNQPQogJ2t0Mfx9zNu06rfRBD
jiIU9tpx2T+NGlWZ8qhbiLo5By8apJavLyqTLavyPSrvsx0B3YzC63T4Age2CDqZYA+OwSMWQ
IDAQAB
```

Then the signature can be verified using RSA and SHA-256 algorithms, as specified in tag "a".

3.1.5 Other detection methods

Microsoft's SenderID validates the sending server's IP addresses against a TXT record published in the originator address's DNS zone [27, 26].

Heuristic-based detection techniques are proposed to identify phishing emails. For example, a simple heuristic is the observation that emails generated by the same toolkit show a high degree of similarity [33]. Once the heuristic identifies a kind of phishing emails, it can be entered into a blacklist, and further emails will be blocked.

3.2 Web Spoofing

A phisher could forge a website that looks similar to a legitimate website, so that victims may think this is the genuine website and enter their passwords and personal information, which is collected by the phisher.

Modern web browsers have certain built-in security indicators that can protect users from phishing scams, including domain name highlighting and https indicators. However, they are often neglected by careless users.

3.2.1 How web spoofing is done?

Creating a forged website It's trivial to clone the look of a website by copying the front-end code; a little bit of web programming is necessary to redirect user's input into a file or database, then show a "website under maintenance" notice.

Proxy software such as `squid` [8] or `Fiddler2` [22] could be extended to create a fully functional clone. Users can successfully sign in and use all the services provided by the original website, while all the inputs are collected by the server, and all the pages may be modified by the server.

Attracting traffic to forged website Once a forged website is online, the phisher must make potential victims visit it. There are a few ways to do this:

- Send spoofed emails (section 3.1) with a link to the forged website.
- Register a domain that is a common typo of a popular website. For example, register `paypel.com` and create a forged `paypal.com`.
- Register the same domain name in a different TLD. Sometimes people will type in their country-specific TLD and expect to get a "localized" version of the website. For example, register `gmail.com.cn` and create a simplified-Chinese forged version of `gmail.com`.

- Do search engine optimization.
- Use pharming (section 3.3).

3.2.2 Browser security indicator: Domain name highlighting

Figure 3: Different highlighted domain names show that these website are unrelated

Phishers tend to use misleading addresses, such as `http://www.paypal.com.cgi-bin.webcr.example.com/`, to deceive users. With domain name highlighting, users can easily interpret the address and identify the current website at a glance (Figure 3). [2]

With domain name highlighting, most web spoofing attacks can be identified, unless the phisher is using pharming.

3.2.3 Browser security indicator: HTTPS padlock

(a) http, no padlock (b) padlock on https

Figure 4: A padlock icon appears in address bar when visiting an https website

HTTPS, the combination of Hypertext Transfer Protocol and Transport Layer Security, provides encryption and identification through public key infrastructure. Modern web browsers display a padlock icon when visiting an https website (Figure 4).

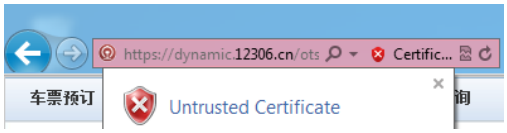


Figure 5: The address bar turns red on invalid certificate

Figure 6: The padlock icon disappears on mixed content

Web browsers verify the certificate presented by the web browser. The certificate is considered invalid if any of the following applies: the certificate is expired; the certificate is not signed by a root CA trusted by the local computer; the certificate is revoked by the CA; the website host name does not match the subject names in the certificate. In this case, there is likely a Man-In-The-Middle attack, so the browser will display a prominent warning (usually a full page), and the address bar would turn red if the user choose to continue onto the website (Figure 5).

Sometimes an https webpage may contain files from http scheme. Every piece of code should be trusted, before a webpage can be trusted. Thus, the padlock icon would disappear (Figure 6).

3.2.4 Effectiveness of browser security indicators and HTTPS

Browser security indicators are not as effective as one might think. A survey [13] reports that 23% of participants used only the content of a webpage to determine legitimacy; an identical-looking clone

under any domain name without https is enough to deceive them. Many users cannot distinguish between a padlock icon in the browser chrome and a padlock icon as the favicon or in the page contents.

Relying on HTTPS is also not sufficient. Malware can install the public key of a phisher's CA to local computer's trusted root CA list, so that certificate signed by this CA would be trusted. When the phishing website is using a similar-looking domain that is registered by the phisher, a real certificate can be requested after domain ownership verification. CAs could be hacked to issue fraudulent certificates [10]. Moreover, if a government is involved in phishing, it can order a CA under its control to issue a certificate for the phishing server.

3.2.5 Other countermeasures

Dynamic Security Skins [12] seems to be a good method. The idea is that the website server generates a unique abstract image for each user, and the web browser also independently computes the same image. The algorithm ensures that a phisher cannot predict this image. The user just needs to compare these two images; if they are identical, the server is legitimate.

3.3 Pharming

Pharming is a type of attack intended to redirect traffic to a fake Internet host. There are different methods for pharming attacks, among which DNS cache poisoning is the most common.

3.3.1 The DNS, and DNS cache poisoning

Domain Name System (DNS) is a critical piece of Internet infrastructure. Designed as a distributed system, DNS publishes a hierarchical database by a hierachy of name servers. To improve performance, clients contact local DNS resolvers maintained by local ISPs, which can cache records from name servers. Clients, resolvers, and name servers talk with each other on UDP port 53. [35]

DNS is critical to Internet security. As shown in Section 3.1, SPF, DKIM, and SenderID all rely on DNS; if DNS is compromised, spoofed emails can get through these signature-based countermeasures. Web spoofing can also be conducted by making DNS respond with the address of phisher's server.

DNS cache poisoning attempts to feed the cache of local DNS resolvers with incorrect records. This is possible because: DNS runs over UDP, and it's easy to spoof the source address of a UDP packet; the DNS packet header contains a 16-bit query ID field, which is relatively short so a birthday attack is feasible.

Domain Name System Security Extension (DNSSEC) [14] is an extension of DNS that provides three distinct services: key distribution, data origin authentication, and transaction and request authentication. Every DNS record can be authenticated via a chain of trust. Cache poisoning is no longer possible, because the phisher cannot produce a correct signature without knowing the private key of the domain. However, DNSSEC is not widely deployed yet.

Google Public DNS, the largest public DNS resolver in the world, mitigates cache poisoning attacks by adding entropy to queries: [7]

- use a random source UDP port
- randomly choose a name server among configured name servers of a zone
- randomize case in the query name. eg. `wWw.eXaMpLe.CoM` and `WwW.ExamPLe.COm` are equivalent

- prepend a nonce label to the query name, if the response is known to be a referral. eg. sending `entriih-f10r3.www.google.com` in a query to root servers

These randomness makes it much harder to construct a matching response than using the 16-bit query ID alone. Thus, cache poisoning is no longer feasible.

3.3.2 Domain hijacking

A more advanced pharming attack is domain hijacking. In domain hijacking, the DNS delegation record at the domain registrar is changed to a name server controller by a hacker, so that all traffic can be redirected globally.

Baidu, the largest search engine in China, was hacked by Iranian Cyber Army in January 2010. [15]

1. The hacker chatted with technical support of Register.com, the domain registrar of baidu.com, to change the email address on file. The change was approved without careful verification.
2. Account password was reset with the new email address.
3. Delegation record was changed to a name server controlled by the hacker.
4. Millions of users were redirected to hacker's server for 4 hours.

A phisher could also use similar techniques to gain control over a domain.

3.3.3 Pharming in smaller scope

DNS cache poisoning and domain hijacking are effective in a large scope, so they would be quickly found and fixed. There are techniques for pharming in a smaller scope, such as the local computer or a home network, that can possibly remain unnoticed for a longer term.

The hosts file is a text file on local computer that contains hostname-to-IP mappings. This file is located at `/etc/hosts` in UNIX systems, or `%WINDIR%\system32\drivers\etc\HOSTS` in Windows systems. TCP/IP stack consults this file before querying DNS. This file could be written by malware for pharming.

ARP spoofing can manipulate traffic in local Ethernet, including redirecting traffic to phishing server. It can be implemented in malware.

In regions of world that Internet is restricted, some people offer solutions that claim to provide uncensored access to Internet. These solutions usually come as software, VPN, proxy server, or hardware home router; they could be either paid or free. A dishonest provider could offer such a solution with pharming built-in, and users looking for uncensored access would end up visiting forged websites without realizing this.

3.4 Malware

Malware is a piece of software developed either for the purpose of harming a computing device or for deriving benefits from it to the detriment of its user [19]. Malware can be used to collect confidential information directly, or aid other phishing techniques.

Client security products are able to detect and remove malware and other potentially unwanted programs, but phishers can make malware undetectable. Financial institutions and online game vendors distribute security programs to protect their customers.

3.4.1 Phishing with malware

Malware can be used to collect confidential information directly, and send them to phishers. Keystrokes, screenshots, clipboard contents, and program activities can be collected. Password input box, where letters are shown as asterisks, can be easily read with a program. Malware can also display a fake user interface to actively collect information. Collected information can be automatically sent to phishers by email, ftp server, or IRC channel.

Malware can also aid other phishing techniques. For web spoofing, it can install phisher's CA public key into local computer's trusted CA list. For pharming, it can change the hosts file or DNS settings, or even run ARP spoofing on local Ethernet. Malware can also enlist the computer into botnets, to send spoofed emails or act as a webserver of forged websites.

3.4.2 Detecting malware with client security products

Client security products are widely deployed. Microsoft Update also pushes "Malicious Software Removal Tool" monthly, which is a lightweight malware scanner.

However, they are not always effective. It's easy to modify a program so that it doesn't contain any known signature, to bypass signature-based detection. There are also techniques to bypass certain behavior-based detection.

3.4.3 Protection for online banking

Figure 7: online banking client from China Merchants Bank

Figure 8: USB token

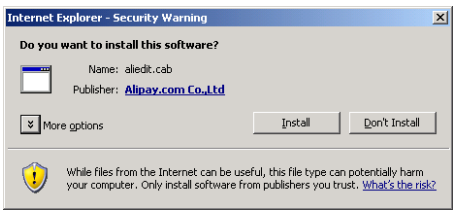


Figure 9: secure text input control from Alipay.com

Financial institutions, the most common targets of phishing, distribute security programs to protect their customers. They usually come in the form of client programs (Figure 7) or browser add-ons (Figure 9). These programs can protect customers in one or more ways:

- implement a secure text input control, usually by using a kernel-space driver, so that (most) keyloggers cannot intercept the keystrokes or read its contents
- encrypt confidential information in memory and in network
- block or remove known malware
- verify the certificate of the financial institution, to protect against Man-In-The-Middle attacks

- make use of hardware tokens or smart cards (Figure 8)

Some of these modes of protection are quite effective but not perfect:

- Like every other program, there may be vulnerabilities in those 'security' programs [23].
- Malware can hide the user interface of the security program, and display a fake user interface.

Hardware tokens are secure enough, if carefully designed. There is one case [34] that a swindler tricked an old man to run a Trojan horse, plug in his USB token, reveal his bank password, and turn off the computer monitor, and then the swindler transferred money out of the account; however that case is out of the scope of phishing.

3.5 Phishing through PDF Documents

Adobe's Portable Document Format is the most popular and trusted document description format. This makes PDF documents more susceptible to phishing threats, owing to their portability and interoperability on multiple platforms. In addition to being a powerful document format, PDF is a comprehensive programming language of its own dedicated to document creation and manipulation with strong execution features. Some critical functions of a PDF language could be misused by an attacker or a hacker to design a PDF document to his/her own advantage and extract the desired information from the victim, thereby creating a new worldwide threat. These potentially dangerous functions include `OpenAction` and `SubmitForm`.

Although Adobe has implemented some security mechanisms in Adobe Reader and Adobe Acrobat in order to alert the user in case of (potentially) malicious attempts, these alert measures are just message boxes, asking the user to allow or block an action. Unfortunately, such message boxes are often neglected by users, and it is possible to bypass these security mechanisms by modifying RdLang32.FRA and AcroRd32.dll files with malware. [11]

4 Additional Preventive Measures

Given the risk of phishing, what are the ways in which individuals and organizations can protect themselves? Though hard to implement but training the end-user is perhaps the best protection mechanism. Sensing the gravity of issue, more non-profit organizations and groups are joining hands to combat phishing scams. Legislation particularly needs attention in this matter to define phishing explicitly and elucidate phishing specific penalties

4.1 User Education

Phishing exploits human vulnerabilities such that technical solutions can only block some of the phishing web sites. It doesn't matter how many firewalls, encryption software, certificates, or two-factor authentication mechanisms an organization has if the person behind the keyboard falls for a phishing attack.

A study on effectiveness of several anti-phishing educational materials suggests that educational materials reduced users' tendency to enter information into phishing webpages by 40%; however, some of the educational materials also slightly decreased participants' tendency to click on legitimate links [30]. This leads to the belief that it is of paramount importance to find a new and efficient way of educating a large proportion of the population[32]. The challenge lies in getting the user's attention to these security tips and advises.

There are few questions that arise: Should we implement all these protection mechanisms which complicates the user interface? Should we provide better user experience at the cost of reduced security or improve security at the cost of user inconvenience? Several recent surveys indicate that lack of security is leading to loss of customer confidence in Internet commerce. That means users want appropriate security controls in place even if it means carrying a password token or getting their passwords on SMS. Today phishing is recognized by users as a real and potentially damaging threat. If appropriate anti-phishing controls are not put in place, chances are high that customers might switch to a more secure party to do business.

4.2 Anti-Phishing Groups

PhishTank, launched in October 2006, is a collaborative clearing house for data and information about phishing on the Internet. PhishTank employs a sophisticated voting system that requires the community to vote "phish" or "not phish", reducing the possibility of false positives and improving the overall breadth and coverage of the phishing data. It also provides an open API for developers and researchers to integrate anti-phishing data into their applications at no charge. PhishTank is backed by OpenDNS, a public DNS resolver; OpenDNS utilizes PhishTank data to prevent phishing attacks for their users.

Formed in 2003, the Anti-Phishing Working Group (APWG) is an international consortium that brings together businesses affected by phishing attacks, security products and services companies, law enforcement agencies, government agencies, trade association, regional international treaty organizations, and communications companies [3].

FraudWatch International, a privately owned Internet security company established in 2003, provides a variety of anti-phishing products and services to protect financial service, e-commerce, and Internet hosting companies from phishing.

4.3 Legal Aspects

Currently little legislation related to phishing exists; this appears to be due to a lack of awareness at the governmental level. In order for technical and educational solutions to be successful, government support is required.

The UK Fraud Act of 2005 covers fraud by false representation, however this does not specifically mention phishing; suggesting that not only the end users but also the governments are unaware of the dangers of phishing [32].

In 2005 a bill named The Anti-Phishing Act of 2005, "A bill to criminalize Internet scams involving fraudulently obtaining personal information, commonly known as phishing", was presented in United States Senate to combat phishing and pharming. The bill proposed a five-year prison sentence and/or fine for individuals who commit identity theft using falsified corporate websites or e-mails. Thus it allows law enforcement officials to fight phishing scams, by creating an opportunity to prosecute before the actual fraud takes place [6].

The Anti-Phishing Act couldn't become a law at federal level, but there are a few states including California, New Mexico, Arizona, and Texas which have strict anti-phishing laws in place [9].

Besides this there is still much work to be done on international basis. Most phishing scams operate overseas and it is exceedingly difficult and time consuming to prosecute an individual residing in a foreign country [5].

References

[1] Identity thieves take advantage of voip. http://www.icbtollfree.com/article_free.cfm?articleId=5926.

[2] Internet explorer 8 features - safer: domain highlighting. http://windows.microsoft.com/en-US/internet-explorer/products/ie-8/feat%ures/safer?tab=ie8dom.

[3] Opendns' phishtank.com and anti-phishing working group to share data. http://www.opendns.com/about/announcements/19/.

[4] Phishing - word spy. http://www.wordspy.com/words/phishing.asp.

[5] Phishing- consumer laws. http://consumerprotection.uslegal.com/phishing/.

[6] Proposed law aims to fight phishing. http://www.pcworld.com/article/119912/proposed_law_aims_to_fight_phishi%ng.html.

[7] Public dns security benefits. https://developers.google.com/speed/public-dns/docs/security.

[8] squid: Optimising web delivery. http://www.squid-cache.org/.

[9] Taking legal action against phishers. http://www.sis.pitt.edu/~nophish/expert/legal.html.

[10] Heather Adkins. An update on attempted man-in-the-middle attacked. http://googleonlinesecurity.blogspot.com/2011/08/update-on-attempted-ma%n-in-middle.html, Aug 2011.

[11] Gundeep Singh Bindra. Masquerading as a trustworthy entity through portable document file (pdf) format. In *Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on and 2011 IEEE Third International Confernece on Social Computing (SocialCom)*, pages 784 –789, Oct 2011.

[12] Rachna Dhamija and J. D. Tygar. The battle against phishing: Dynamic security skins. In *Proceedings of the 2005 symposium on Usable privacy and security*, SOUPS '05, pages 77–88, New York, NY, USA, 2005. ACM.

[13] Rachna Dhamija, J. D. Tygar, and Marti Hearst. Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 581–590, New York, NY, USA, 2006. ACM.

[14] D. Eastlake 3rd. Domain Name System Security Extensions. RFC 2535 (Proposed Standard), March 1999. Obsoleted by RFCs 4033, 4034, 4035, updated by RFCs 2931, 3007, 3008, 3090, 3226, 3445, 3597, 3655, 3658, 3755, 3757, 3845.

[15] Owen Fletcher and Robert McMillan. Baidu: Registrar 'incredibly' changed our e-mail for hacker. http://www.computerworld.com/s/article/9162118/Baidu_Registrar_incredib%ly_changed_our_e_mail_for_hacker, 2010.

[16] Anti Phishing Working Group. Origins of the word "phishing". http://www.antiphishing.org/word_phish.html.

[17] T. Hansen, D. Crocker, and P. Hallam-Baker. DomainKeys Identified Mail (DKIM) Service Overview. RFC 5585 (Informational), July 2009.

[18] Jason Hong. Why have there been so many security breaches recently? http://cacm.acm.org/blogs/blog-cacm/107800-why-have-there-been-so-many-%security-breaches-recently/fulltext.

[19] Markus Jakobsson and Steven Myers. *Phishing and countermeasures: understanding the increasing problem of electronic identity theft*. John Wiley & Sons, Inc., 2007.

[20] Lance James. *Phishing Exposed*. Rockland, MA : Syngress, 2005.

[21] J. Klensin. Simple Mail Transfer Protocol. RFC 5321 (Draft Standard), October 2008.

[22] Eric Lawrence. Fiddler web debugger - a free web debugging tool. http://www.fiddler2.com/fiddler2/.

[23] luoposhusheng. Plaintext disclosure vulnerability of alipay password security control. *Hacker Defense*, pages 6–8, Nov 2011.

[24] John Markoff. Larger prey are targets of phishing. http://www.nytimes.com/2008/04/16/technology/16whale.html.

[25] Julian Mehnle. Sender policy framework - introduction. http://www.openspf.org/Introduction.

[26] Julian Mehnle. Spf vs sender id. http://www.openspf.org/SPF_vs_Sender_ID.

[27] Microsoft. Sender id framework overview. http://www.microsoft.com/mscorp/safety/technologies/senderid/overview.m%spx, Sep 2004.

[28] Federal Bureau of Investigation. Spear phishers. http://www.fbi.gov/news/stories/2009/april/spearphishing_040109.

[29] PhishTank. Phishtank stats-jan 2012. http://www.phishtank.com/stats/2012/01/?y=2012&m=01.

[30] Steve Sheng, Mandy Holbrook, Ponnurangam Kumaraguru, and Lorrie Cranorand Julie Downs1. Who falls for phish? a demographic analysis of phishing susceptibility and effectiveness of interventions. In *28th international conference on Human factors in computing systems*, Apr 2010.

[31] Wikipedia. Phishing — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Phishing&oldid=484977983, 2012. [Online; accessed 2-April-2012].

[32] C. Wilson and D. Argles. The fight against phishing: Technology, the end user and legislation. In *Information Society (i-Society), 2011 International Conference on*, pages 501 –504, Jun 2011.

[33] Guang Xiang, Bryan A. Pendleton, Jason Hong, and Carolyn P. Rose. A hierarchical adaptive probabilistic approach for zero hour phish detection. In *Proceedings of the ESORICS 15th European Symposium on Research in Computer Security*, pages 571–589, 2010.

[34] Yiru Xu. 67-year-old man swindled 700k from online banking account. http://finance.sina.com.cn/money/bank/guangjiao/20110326/16149598471.sh%tml.

[35] Beichuan Zhang. Domain name system (dns).

Honeypots

Mathias Gibbens Harsha vardhan Rajendran

1 Introduction

The primary goal of computer security is to defend computers against attacks launched by malicious users. The are a number of ways in which researchers and developers can work to protect the software that they write. Some are proactive, like code reviews and regression testing, while others are reactive, like the pwn2own contest where new vulnerabilities are used to exploit browsers. Some tools can take on aspects of both; one class of these tools are honeypots.

A honeypot is a computer which has been configured to some extent to seem normal to an attacker, but actually logs and observes what the attacker does. Thanks to these modifications, accurate information about various types of attacks can be recorded. The term *honeypot* was first presented by Lance Spitzner in 1999 in a paper titled *To Build a Honeypot* [20].

Honeypots are unique because they allow a security researcher to see and record what actions a malicious user takes on a compromised computer without necessarily interfering or revealing to the attacker that they are being monitored. Because of this invisibility, valuable intelligence can be gathered about the actual strategies of an attacker. A honeypot can be configured to be either proactive or reactive to attacks, depending on the needs of the person who set it up.

Figure 1 illustrates the role of honeypots in a typical network set up. To an outside attacker (Evan and Eve), the honeypots are indistinguishable from the actual production servers. Thus, they will not behave any differently when attacking them. However, the network security team can monitor the honeypots for recorded attacks and later analyze them. The honeypots can also help to absorb attacks directed against the real servers.

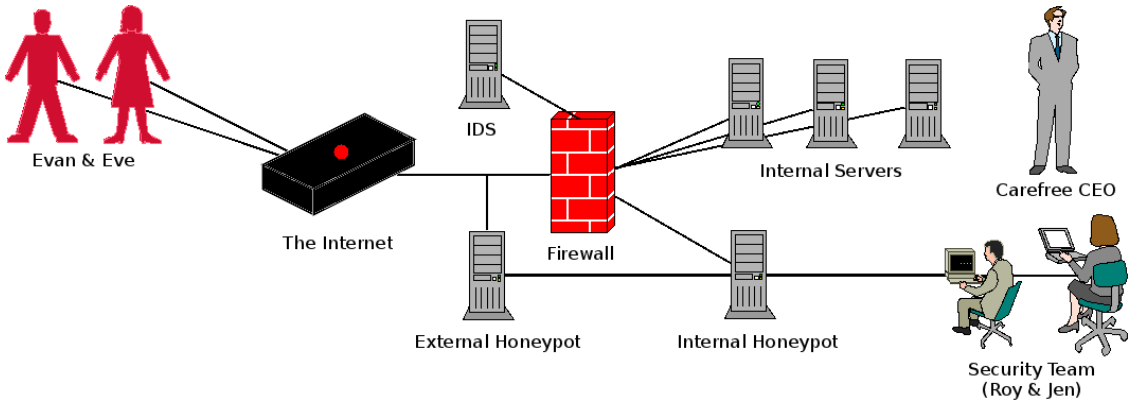


Figure 1: The key characters in our drama.

1.1 History

The term *honeypot* was inspired by actual real-life honeypots. Since such a pot contains something desirable (the honey) to someone (a child or a nest of ants, for example), it could be used to lure them out and then observe them. The same is true for a computer honeypot: a tempting target is presented to an attacker, who then comes out and performs his attacks. Figure 2 shows some of the major honeypot development milestones.

While Spitzner was the first to introduce the term *honeypot*, the concept of trying to figure out how attacks are performed has been around since at least the mid-1980's [19]. An example from the early days of Internet history was recounted by Bill Cheswick regarding his time at AT&T Bell Laboratories in January of 1991 [4]:

> On Sunday evening, January 20, I was riveted to CNN like most people. A CNN bureau chief in Jerusalem was casting about for a gas mask. I was quite annoyed when my terminal announced a security event:
>
> `22:33 finger attempt on berferd`
>
> A couple of minutes later someone used the debug command to submit commands to be executed as root – he wanted our mailer to change our password file!

Cheswick was able to see the commands this remote attacker was issuing, and then fed back modified responses to him. After a day or so, Cheswick and some colleagues worked on setting up a chroot environment for their attacker to play in:

> I wanted to watch the cracker's keystrokes, to trace him, learn his techniques, and warn his victims. The best solution was to lure him to a sacrificial machine and tap the connection. ... We constructed such a chroot "Jail" (or "roach motel") and rigged up logged connections to it through our firewall machine. ... A little later Berferd [the attacker] discovered the Jail and rattled around in it. He looked for a number of programs that we later learned contained his favorite security holes. To us the Jail was not very convincing, but Berferd seemed to shrug it off as part of the strangeness of our gateway. Berferd spent a lot of time in our Jail.

The attacker was observed for several months, until Cheswick finally shut it down. During that same time, this attacker was also active in attacking several other computer networks, and some of the administrators coordinated their information in an attempt to identify the source of the attacks. At the end, all that was known for sure was that the attack was coming from Sweden, and that the attacker had a knack for subverting the system he was on. You most certainly did not want to give an account to this attacker!

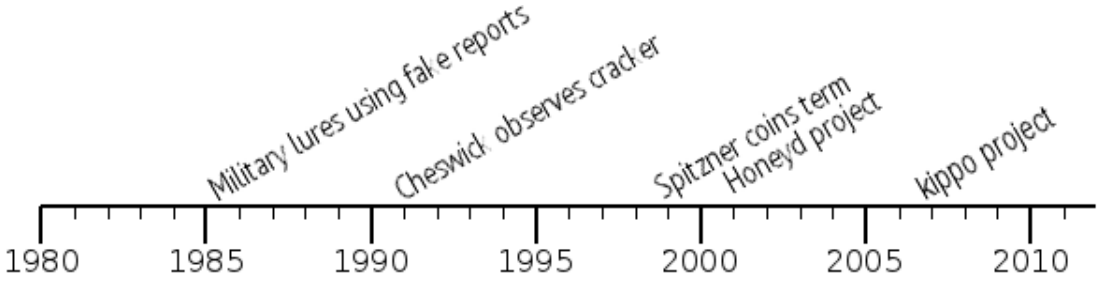


Figure 2: Honeypot development milestones.

1.2 Classification

There are several possible ways to classify honeypots. Some of the more popular are by the level of interaction available to the attacker, the type of data collected, and the type of system configuration [2, 19, 27].

1.2.1 Level of interaction

The most common type of classification is based on the level of interaction which is provided to the malicious user by the honeypot. The more interactive an environment presented, the closer the honeypot becomes to the actual targets of attack, and then potentially more accurate information can be gathered. The downside is that the more realistic honeypots present greater challenges to configure and setup. There are three levels [2]:

1. **Low-interaction** One or more simple services are made available which log all communication attempts to specific services, like a web or SSH server. Usually these are just simple daemons which provide the person who configured them a passive way to monitor attack attempts. The host operating system is not vulnerable to attacks, so low-interaction honeypots are fairly safe to run, but are also unable to be used where a more complex, interactive environment is needed, like a SMTP server.

2. **Medium-interaction** Medium level honeypots begin to emulate collections of software to present a more convincing front to the attacker, but still shield the host operating system. Emulating a collection of software can become quite complex, since the emulated programs should respond the same way as their real counterparts, but must not have the same security issues, otherwise the emulation will break. Further, there are more points of attack for the malicious user, so the chance of system compromise is raised.

3. **High-interaction** Finally there are high-interaction honeypots. The full host operating system is presented to the attacker, along with actual instances of programs, instead of their emulations. The usual goal of high-interaction honeypots is for the attacker to gain root access on the machine, and then see what he does. Because of this goal, this level of honeypot has the highest risk, but also the highest potential for collecting information. Honeypots at this level need constant supervision, since the attacker will actually control it, and could try to use it as a jumping-off point for further attacks.

1.2.2 Type of data collection

A second way of classifying honeypots is by looking at what type of data is collected concerning an attack [19]. A honeypot can be set up to detect and record one or more types of data: events (things that happen which change something in the honeypot), attacks (attempts by a malicious user to exploit a vulnerability), and intrusions (successful attacks that penetrate the honeypot). All three types of data can provide valuable information about the malicious user, and most honeypots can provide some information from each group.

1.2.3 System configuration

Honeypots can work either alone or as a group. A group of honeypots is commonly referred to as a *honeyfarm*. Individual honeypots may be simpler to set up, but they are inherently less powerful and more prone to unintended failure than honeyfarms because they lack the load balancing abilities and redundancy of a group of cooperating servers.

1.3 Basic uses

Honeypots are typically used in one of two main fashions: as part of an organization's computer network monitoring and defense, and by security researchers who are trying to keep up with the activities of blackhats.

Production environment Honeypots deployed in a production environment serve to alert administrators to potential attacks in real time. Because of the advanced level of logging and information that is available on a honeypot, better defenses against the attacks may be able to be devised for implementation on the real servers. Production honeypots tend to be reactive in nature.

Research environment In a research environment, security analysts are trying to figure out what the next generation of attacks by malicious users will be. These honeypots can be quite dynamic, as they are adjusted and tweaked to lure attackers and respond to new attack strategies. Often a research honeypot is actively monitored by a person in real time.

2 Deception techniques using honeypots

Honeypots have shown a lot of promise in the field of digital security. A part from behaving as a system with no production value, honeypots can also be put to other ingenious uses besides just luring attackers. A few novel deception techniques are explored in this section.

2.1 Honeypots as mobile code throttlers

Viruses and worms, also referred to as *mobile code* since they can often spread with little need for manual assistance, are quite prevalent these days in cyberspace. Almost all of the virus scanners are signature based, that is, they scan for particular file signatures and report them. A new approach to prevent virus spread has been proposed, using honeypots. The approach is based on the fact that an infected machine makes more connections than regular ones. Also, it must be noted that this approach in no way prevents a system from getting infected, but it can stop the spread of the virus. In the process, we sacrifice a few machines for the common good. This approach depends on the network behavior of mobile code as opposed to signature based systems. More specifically it depends on the mobile code attempting to leave the system. Having a network of honeypots to throttle mobile code has a performance incentive and offers better security.

Working model Whenever a TCP connection is established, a three-way hand shake is first performed. In a nutshell, system A sends a SYN packet to system B. System B replies back with a SYN/ACK packet. System A again sends an ACK packet to system B. The idea is to count and limit the SYN packets sent during an infection which stops the spread of the virus.

This system works on the observation that under normal usage the majority of connections are made to recently connected destination addresses and no more than one connection per second is made to a target not recently connected. Therefore whenever a request for a connection goes out, it is checked for "newness" by comparing it with a set of recently visited hosts. This set is called the *working set* and can have up to 5 addresses. If the destination address is in the working set, the connection is immediately allowed. If the destination address is new, but the working set is not full, then this address is pushed into the working set and the connection allowed. Otherwise, the destination address is put in the delay queue. Once every second, the delay queue is processed and the SYN packet at the head of the queue is sent out. Also, this destination address is added

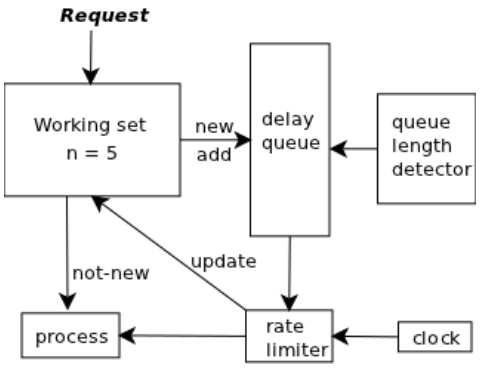


Figure 3: Control flow for mobile code throttler.

To: Chief Financial Officer
From: Security help desk
Subject: Access to financial database
Sir,
The security team has updated your access to the company's financial records. Your new login and password to the system can be found below. If you need any help or assistance, do not hesitate to contact us.
https://finances.ourcompany.com
login: cfo
password: H0n3yt0k3n

Security Help Desk

Figure 4: An example of a Honeytoken.

to the working set. In essence, connections to the five most recently used machines are immediate and new connections are delayed for 1 second. This interaction is diagrammed in Figure 3. Now, during an infection the delay queue length grows to very large values. Thus, the rate delimiter detects this deviation from the normal and throttles the rate of new connections, in turn affecting the spread of viruses. New connections are not processed and the system administrator is alerted.

This idea has it own merits and demerits. Merits include easy deployment, no legal hassles, and no dependence on virus signatures (so no updates, etc.). Demerits include a denial of service (DoS) by the system when new connections are dropped when the delay queue length crosses the threshold value. Also, this system is based on the assumption that all systems connect to recently visited hosts. This may not always be true, as in the case of an email server.

2.2 Honeytokens (cost-effective honeypots)

The greatest misconception about honeypots is that they have to be a computer. To re-iterate the definition of a honeypot is "an information system resource whose value lies in the unauthorized or illicit use of that resource." So, in essence a Honeytoken is a Honeypot which is not a computer, but some digital entity like a credit card number, Microsoft Word file, database entry, bogus login, etc. whose value lies in the unauthorized use of that resource. No one should be interacting with a honeytoken. Anyone who does is, in all probability, an attacker. Consider the following example of a honeytoken.

Assume that there's a hospital with a large patient record database going into tens of thousands of entries. Assume that there exist authorized users numbering in the thousands. Maintaining who is authorized to perform what actions is complex and might end up in giving out many false alarms. To avert such disasters, a bogus record for a celebrity or other popular figure is created. This is a honeytoken and has no value. Hence it must not be accessed by anyone. When a hacker is scanning the database for interesting information, this record will stand out. Therefore the attacker would naturally access it which should trigger an alarm.

Working model Like every other security solution present, this one is useless when used alone. Its value lies in combining it with other solutions. Honeytokens only detect illicit activity and therefore need to be combined with another solution to actually trace the attacker. Also, honeytokens can play a major role in detecting internal attackers in an organization. Consider the following example.

Assume that there exists a software firm: Reynholm Industries. The security team had a

suspicion that the mail interaction among the executives is being illegally monitored. They send out a mail like the one shown in Figure 4. An attacker who comes across this mail will naturally log into that domain with the username and password provided. What the attacker does not know is that the domain is a honeypot. Upon logging in, his movements will be monitored and he might eventually get caught.

In essence honeytokens are lightweight honeypots which are cost effective, simple to deploy and are highly effective.

3 Honeypot Implementations

When a programmer wishes to implement a honeypot, there are two main considerations: the needed complexity to be convincing and how to hide the fact that it is a honeypot from the attacker. Several honeypot projects exist, both commercial and free; [2] describes several in detail. We will highlight two popular open source honeypots: Honeyd and kippo.

3.1 Honeyd

Honeyd [17] is a low-interaction virtual honeypot that simulates virtual computer systems at the network level. It deceives the attacker by simulating the network stack of various operating systems, thus making him believe that he is interacting with a real system over the network.

Honeyd simulates only the network stack rather than the entire operating system. This ensures that even if Honeyd is compromised, the attacker cannot do much damage. Honeyd can be combined with a virtual machine (VM) to simulate multiple operating systems. To add to the realism, Honeyd also simulates arbitrary network topologies, which will further convince the hacker that he is on a real system.

Receiving network data Honeyd in essence replies to the network packets whose destination is any of the simulated honeypots. The network can be configured in several ways. For example by using proxy ARPs (answering the ARP queries of an IP address not in the current network) or network tunnels (a connection across networks). Let A be the IP address of the router and B be the IP address of the Honeyd host. Assume the case in which all IP addresses of the virtual honeypots lie in the local network. When the attacker sends a packet to a specific VM, the router scans its routing table for the forwarding address of the VM. Then it does one of three things: drops packet, forwards packet to another router or sends it directly to the destination. To direct the traffic for a VM to B, the best method is to configure routing entries for all the VMs that point to B. Then the router will directly forward the packets to the Honeyd host. If no route has been configured, then the router uses ARP to determine the MAC address of the honeypot. Normally the ARP requests would fail as all the instances are virtual, but the Honeyd host B has been configured to reply to the ARP requests for the virtual machines, using its own MAC address. Hence the router can directly send the virtual machines' packets to B's MAC address.

Honeyd comprises several components (shown in Figure 5): a configuration database, a central packet dispatcher, protocol handlers and a personality engine. Incoming packets first go through the central packet dispatcher. It is aware of three protocols, TCP, UDP and ICMP. Other packets are discarded. The dispatcher queries the configuration corresponding to the destination address. Then it passes the packet to the protocol-specific handler. On receiving a TCP or UDP packet, the handler helps establish connections to various services. The framework checks if a specific packet is part of an already started service application. If so, all packets are redirected to the

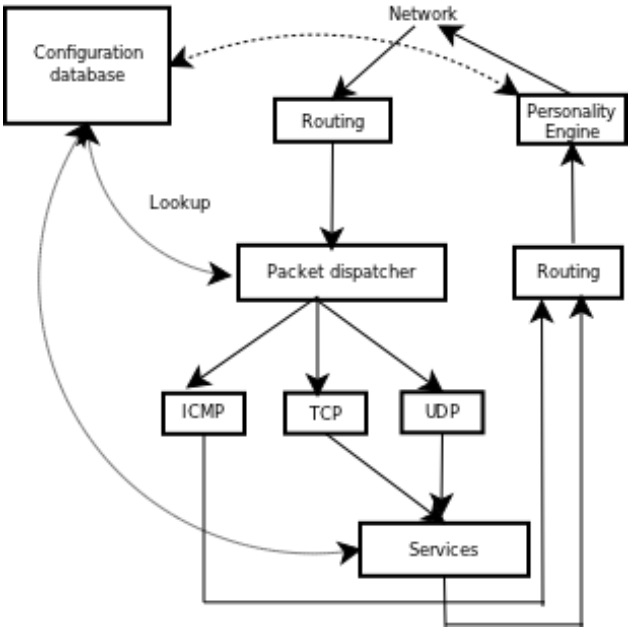


Figure 5: Honeyd architecture.

service, otherwise a new service is started. The handler also helps in redirection of connections; for example, redirecting a DNS request to an actual DNS server. Then the packet is sent to the personality engine which manipulates the packet content to make it look like it has originated from the network stack.

Personality engine The main concern that has to be dealt with while modeling virtual honeypots is that whenever an attacker runs a fingerprinting tool, they shouldn't expose the honeypot as any different from the original system, lest the attacker detects the ruse. This is done by mimicking the personality/behavior of the underlying Operating System. Honeyd simulates the corresponding network stack for various kinds of operating systems. It makes the necessary changes in the headers of every packet that leaves the system so they match the characteristics of the underlying operating system. Honeyd uses N-Map fingerprint information to modify the network stack.

Logging The framework ensures that services report data to be logged using stderr. It then uses syslog to store/maintain that information on the system.

3.2 Service-specific honeypots

In addition to honeypots like Honeyd, there are simpler honeypot options which are easier to set up and monitor. Usually these are low- or medium-interaction stand alone honeypots. A computer running one of these simpler honeypots usually runs it as a service on a specific port and isolates the underlying operating system and file systems. Due to this isolation, more than one honeypot may be running on a single system at the same time, or the computer may be used for additional purposes.

One of the most commonly attacked class of services today is remote administration, like RDP/VNC for Windows servers, and SSH for Linux servers. Because of the access granted by a compromised remote connection, they present a tempting target for an attacker. Due to the

popularity of attack, it is common to see SSH honeypots. One particular SSH honeypot implementation is called kippo [23], which mimics an actual SSH server and virtual file system using Python. It can log both connection attempts, and if the attacker is allowed access, all commands and outputs of the emulated shell are saved for playback and analysis at a later date. Because of its limited focus, kippo is easy to setup and use, but does have some limitations to what it can mimic.

Several other services can be honeypotted, such as a web server, FTP server, or DNS server. However, the idea for all these simpler honeypots remains the same: observe and capture information about the attacker.

4 Honeypot deployment strategies

After implementation and testing, the honeypot needs to be deployed into an existing network. Depending on the need, honeypots can be deployed using various strategies and in various positions on a network (Figures 1 & 6). Some honeypot deployment techniques may involve a variety of legal issues.

4.1 Sacrificial lamb

This is the simplest of all deployment techniques. As the name suggests, the honeypot just sits on the network, waiting to be pounced upon by an attacker. That is, you give the attacker what he wants and keep him busy, all the while trying to track him down. The honeypot as always has no production value, but helps buy time for the administrators to act upon. Examples of honeypots which can be deployed using the Sacrificial lamb strategy include Deception tool kit (DTK) and Specter.

4.2 Deception ports on production systems

This deployment strategy involves mimicking various services on the different ports of the system. For example HTTP would be mimicked on port 80, SMTP on port 25, etc. The idea here is quite similar to the one in the Sacrificial lamb technique. That is, you deceive the attacker in such a way that he is stuck-up trying to solve the deception, while measures like trace-back, forensics, etc. can be taken. This kind of deployment is the most common and has the least liability. Examples of honeypots which can be deployed using this strategy include Honeyd and Specter.

4.3 Proximity decoys

The main idea behind this deployment strategy is to maintain the honeypot close to the production server. If you ensure that the honeypot is in the same subnet that the main server is in, then it would technically imply that it is still on the same network. Therefore, there will not be any legal hassles, as you are entitled to use any techniques to protect your assets as long as it is within your network. Also, it will be easier to re-route any attacks aimed at the production servers or to trap them. Examples include deployment of honeypots using VMware and Usermode Linux (UML).

4.4 Redirection shield

Honeypots deployed using this strategy mainly use port re-direction or traffic re-routing. It is believed that this strategy is the most promising one of all, as it has the most commercial value. Just like Software as a Service, honeypots can be used as a service and not just a device. The

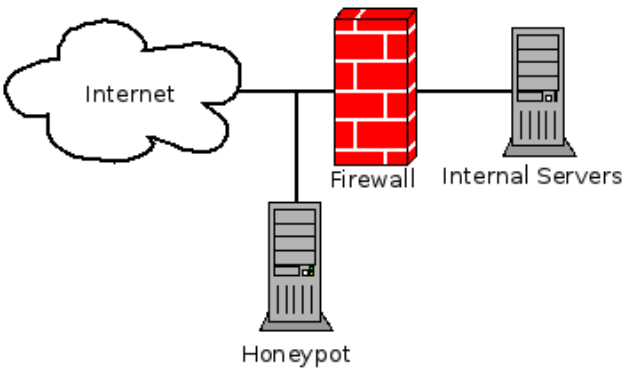


Figure 6: A example of an exposed honeypot.

strategy mainly involves installing re-routing switches at the client sites. Now the honeypots can sit anywhere in the world, and receive all the traffic flowing to the servers and track hackers.

4.5 Minefield

Honeypots deployed using the Minefield strategy are placed at the perimeter. Just as the name suggests, they explode upon contact (usually by an attacker). To ensure complete security, the intrusion detection systems (IDS) and vulnerability scanners are also placed in the network. But rather than monitoring the production servers, these IDSs and vulnerability scanners use the contents of the honeypots to sound alarms. So, the probability of producing false alarms is highly reduced. In essence these Honeypots act as a third layer of defense and they trap, deceive, trace or tarpit attackers. Examples include LaBrea and Honeyd both when used in Stealth mode and Mantrap.

5 Pros/Cons

Honeypots are a powerful tool that can be used to monitor attacks. However, they are not without fault.

Pros Chief of the advantages of honeypots is that they shield actual production servers from direct attack by emulating or visualizing services. If an attack would have penetrated a weakness, only the decoy was impacted, instead of the real server. Additionally, since honeypots can maintain logs of attacks, information can be gathered about the tactics and software used by an attacker once the system has been compromised. This lets security researchers devise defenses to be implemented on the real servers to foil hacking attempts. Finally, honeypots do not have to contain any actual data of value. If data is leaked during an attack, the administrator of the honeypot can be confident that no sensitive information was compromised.

Cons In spite of their usefulness, honeypots do have disadvantages. First, they are at best a copy of the real target. If the copy is not good enough, the information recorded or means of compromise may not closely match an actual server's response. Second, the emulation done by a honeypot must not have the same weaknesses, known or unknown, otherwise the attacker will be able to compromise the honeypot itself. Thirdly, using and deploying a honeypot takes time in addition to the time needed for setting up and maintaining the real server. Administrative staff

may not have the needed time to do this, which would prevent them from using honeypots as part of their security defenses.

6 Real life results

Honeypots are used every day as part of comprehensive security configurations to detect, deflect, and prevent security breaches. The type of deployment strategy used directs what role the honeypot plays in the system. An example of honeypots used in a minefield configuration was described by a person in Utah State University's IT department [12]. In addition to their normal SSH servers, several SSH honeypots are deployed on servers which do not have any entries in DNS pointing to them. Thus, any SSH attempts will most likely be coming from port scanners. Credentials from the attacker are collected and the offending IP address is banned. Additionally, since only automated attacks should be seen on these honeypots, USU is able to work with ISPs who own the IP address block containing the attacker's source IP to help remove malware from infected computer systems.

7 Improvements

Like any piece of software designed to target and reveal information about malicious users, there is a constant battle between the researchers trying to outsmart the attackers and vice versa. If a certain identifying characteristic of a honeypot is discovered, attackers can look for that, and if it is found, they know they are on a fake system. Developers then need to find a way to mask the identifier to make the honeypot even more similar to the targeted system or service. Additionally, honeypot implementations may contain undesirable bugs or security flaws which could open up the honeypot for use maliciously itself. Lastly, as new versions of servers and software are published, naturally hackers will target them. Honeypots need to be updated regularly to be able to continue providing a tempting target.

Most of the honeypots which are in use today concentrate on emulating parts of a server. However, there are other places in a networked environment where the idea of using a honeypot could give security professionals an advantage in watching attacks. One example is managed networking equipment (switches, routers, etc). If these devices are not properly configured or are running outdated firmware, an attacker may be able to intercept or disrupt network traffic. A honeypot representing itself as a network device could help detect how such an attack was performed.

8 Conclusion

Hopefully through this short introduction to honeypots you have been able to see the potential that they bring to the field of computer security. Using them allows researchers and security professionals to monitor malicious attackers without their knowledge, and hopefully learn about their techniques and exploits in real time. Also a few honeypot deployment schemes have been discussed, which we hope has given the reader a fair idea of how honeypots are actually used in the real world. But one important point to be noted is that usage of only honeypots to secure a system is not recommended. You have to have multiple security tiers to successfully thwart attacks. Therefore combining honeypots with other security solutions is necessary.

References

[1] Cláudia J. Barenco Abbas, L. Javier García Villalba, and Victoria López López. Implementation and attacks analysis of a honeypot. In *Proceedings of the 2007 international conference on Computational science and Its applications - Volume Part II*, ICCSA'07, pages 489 – 502, Berlin, Heidelberg, 2007. Springer-Verlag.

[2] Reto Baumann and Christian Plattner. Honeypots, 2002.

[3] Jeremy Briffaut, Jean-Francois Lalande, and Christian Toinard. Security and results of a large-scale high-interaction honeypot. *Journal of Computers*, 4(5):395 – 404, 2009.

[4] Bill Cheswick. An evening with berferd in which a cracker is lured, endured, and studied. In *In Proc. Winter USENIX Conference*, pages 163–174, 1992.

[5] Marc Dacier, Fabien Pouget, and Hervé Debar. Honeypots: practical means to validate malicious fault assumptions. In *Dependable Computing, 2004. Proceedings. 10th IEEE Pacific Rim International Symposium on*, pages 383 – 388, march 2004.

[6] David Dagon, Xinzhou Qin, Guofei Gu, Wenke Lee, Julian Grizzard, John Levine, and Henry Owen. Honeystat: Local worm detection using honeypots.

[7] Maximillian Dornseif, Thorsten Holz, and Sven Müller. Honeypots and limitations of deception.

[8] Anand Gupta, S. K. Gupta, Isha Manu Ganesh, Pankhuri Gupta, Vikram Goyal, and Sangeeta Sabharwal. Opaqueness characteristic of a context honeypot system. *Information Security Journal: A Global Perspective*, 19(3):142 – 152, 2010.

[9] Amit Lakhani. Deception techniques using honeypots. Master's thesis, University of London, UK.

[10] John Levine, Richard LaBella, Henry Owen, Didier Contis, and Brian Culver. The use of honeynets to detect exploited systems across large enterprise networks. *IEEE Systems Man and Cybernetics SocietyInformation Assurance Workshop 2003*, (June):92–99, 2003.

[11] Mario Marchese, Roberto Surlinelli, and Sandro Zappatore. Monitoring unauthorized internet accesses through a 'honeypot' system. *International Journal of Communication Systems*, 24(1):75 – 93, 2011.

[12] Miles. Ssh honeypots have many benefits. http://ask.slashdot.org/comments.pl?sid=2170514&cid=36188168, Mar 2011.

[13] Iyatiti Mokube and Michele Adams. Honeypots: concepts, approaches, and challenges. In *Proceedings of the 45th annual southeast regional conference*, ACM-SE 45, pages 321–326. ACM, 2007.

[14] Anh-Quynh Nguyen and Yoshiyasu Takefuji. Towards an invisible honeypot monitoring system. In *ACISP*, volume 4058 of *Lecture Notes in Computer Science*, pages 111–122, Melbourne, Australia, 2006. Springer.

[15] Laurent Oudot. Fighting internet worms with honeypots.

[16] Van-Hau Pham and Marc Dacier. Honeypot trace forensics: The observation viewpoint matters. *Future Generation Computer Systems*, 27(5):539 – 546, 2011.

[17] Niels Provos. A virtual honeypot framework. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 1–14, Berkeley, CA, USA, 2004. USENIX Association.

[18] Niels Provos. Development of the honeyd honeypot. http://honeyd.org/, Feb 2012.

[19] Christian Seifert, Ian Welch, and Peter Komisarczuk. Taxonomy of honeypots, 2006.

[20] Lance Spitzner. To build a honeypot. http://www.spitzner.net/honeypot.html, Aug 1999.

[21] Lance Spitzner. Honeypots: catching the insider threat. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 170–179, 2003.

[22] Lance Spitzner. *Honeypots: Tracking hackers*. Pearson Education, Inc., 2003.

[23] Upi Tamminen. kippo - ssh honeypot. https://code.google.com/p/kippo/, Feb 2012.

[24] Jamie Twycross and Matthew Williamson. Implementing and testing a virus throttle.

[25] Steve Webb, James Caverlee, and Calton Pu. Social honeypots: Making friends with a spammer near you.

[26] Zaiyao Yi, Liuqing Pan, Xinmei Wang, Chen Huang, and Benxiong Huang. Ip traceback using digital watermark and honeypot. In *Ubiquitous Intelligence and Computing*, volume 5061 of *Lecture Notes in Computer Science*, pages 426–438. Springer Berlin / Heidelberg, 2008.

[27] Feng Zhang, Shijie Zhou, Zhiguang Qin, and Jinde Liu. Honeypot: A supplemented active defense system for network security.

Botnets — Secret Puppetry with Computers

Balaji Prasad T.K (bpt@email.arizona.edu)
Nupur Maheshwari (nupurm@email.arizona.edu)
Department of Computer Science,
University of Arizona

Abstract

Botnets have traditionally been and are still considered one of the top threats to Internet security. The scope of botnets transcends the boundaries of Internet security, leveraging a variety of technologies and strategies. They are capable of launching attacks at a massive scale that are difficult to defend. In this paper we describe anatomy of botnets and malware that recruits new bots, variants of botnets and strategies of defense against botnets.

1 Introduction

Malware is a computer program that compramises the security of a computing system. These malware include *viruses* that replicates and mutates itself to avoid detection in host, *worms* that spreads itself across the network, *Trojan Horses* that pretend to serve a useful purpose but rather undermining the system, or a *Rootkit* that enjoys unauthorized root access in the system to evade detection and continue its malicious activities. A botnet is a network of compromised hosts called"Zombies" which have malware installed in them - called the "bot". The bot exploits vulnerability in the users system and opens a back door, which enables the attacker (BotMaster or BotHerder) to remotely control the zombie [2]. The compromised host listens to commands from the remote master and can report confidential details to the botmaster. Over a period the botmaster and the zombies actively scan for newer victims or set traps so that the zombie army grows in size and strength.
Botnets are the single biggest menace to Internet security. A recent report suggest that 83% of global spam in 2011 was sent by botnets, marking a 6% increase from the previous year. Reports claim that close to 80% of all email messages are spam. Recent statistics suggest that the botnet market is growing and consolidating. 2010 reported a 600% growth in the number of victims and 2011 maintains the trend but only 3 of the top ten botnets reported in 2010 are now active, indicating that newer botnets keep emerging as threat. With the increase in the number of smart phone users, the number of mobile botnets is also on the rise.

2 The Botnet Threat Landscape

Botnets threats are multifaceted and can launch a denial of service attack and several other attacks on a critical government website or other crucial systems and are posing a threat to the security of a nation. The hackers managed to temporarily bring down sites such as cia.gov (the US Central Intelligence Agency) and www.soca.gov.uk (the British Serious Organized Crime Agency.)

2.1 What Can a "Zombie-Army" Do?

Botnet creation begins with the spreading of the malware called "Bot" along with an tembedded payload, The portion of malware that actually exploits a vulnerability, which spreads by email attachments or clicking links that cause drive-by download of the malware. Once the malware manages to penetrate the victim's machine, it creates a backdoor. Now the infected machine further recruits new machines, building the "Bot-army" and when the size of the army is sufficiently large, it can used for a variety of attacks described below.

Distributed denial of service attack : With thousands of zombies under control, the botmaster can launch a denial of service attack on any high profile web site, governmental service, DNS servers thus making them unavailable for legitimate users. Attack usually floods them with fake requests, usually TCP SYN packets, or brute forcing passwords.

Spyware and Malware : The zombies can monitor the activities of the user and constantly report it to the master. It can sniff into the data of the user, log keystrokes, survey vulnerabilities of the port it to a third party.

Identity theft : Botnets are capable of stealing personal information relating to a person's identity, credit card details, SSN, passwords, spending patterns etc.

Adware : Zombies may automatically download, install, and display ad pop-ups and make the user's browser to periodically visit certain websites.

E-mail spam : The master can organize zombie computers to send large scale emails to thousands of users. More than 80% of the spam mails today are sent by zombies.

Click fraud : A malware installed in a zombie can have the capability of imitating a legitimate browser increasing the click counts on pay-per-click advertising networks, thus generating illegitimate revenue [3].

Phishing : The zombies can identify servers with vulnerabilities that can be exploited to host fake websites that resemble legitimate ones so as to trick the users into giving out his personal information.

Cyber Extortion : The botnets have been reported to extort money from famous companies by launching a DDOS attack and demanding money to withdraw the attack. Companies are left with little means of saving their image and stopping the launched attack.

Anonymous Internet connections : The attacker can use one of his zombies for nefarious attacks on other systems and uses it as a means of concealing his identity.

Illegal file transfers : botnets can be used as a site for hosting illegal material for monetary gain. The host's identity is again concealed as he hides behind the zombie army that he had built.

Brute forcing : botnets can be used to send series of request and brute force passwords to gain illegitimate access to bank accounts. This can be easily defended these days by restricting the number of password attempts.

2.2 Some Famous Botnets

Some botnets took the world by storm. A few notorious ones are:

- *The Storm bot* is a P2P botnet which peaked during 2007. Number of victims was estimated between 50,000 and 10 million. Storm bot was famous for its "fighting-back" capabilities—it was designed to launch a DDoS attack on any system that tried to scan and delete the malware that created Storm bot. Malware behind this bot is called *"Storm-Worm"* which exploits vulnerabilities in Windows. It began infecting computers using an e-mail message with a subject line about a recent weather disaster, "230 dead as storm batters Europe."
- *Conflicker* is another popular p2pbotnet built with the malware also called *Conflicker*. The attacker sends a carefully crafted RPC request, which causes buffer overflow in the target machine. The target machine is forced to execute a shellcode, which in turn communicates with the attackers machine to download a Win32 DLL. The DLL attaches itself with svchost.exe and hence starts every time the computer is rebooted. It managed to penetrate the French Navy, United Kingdom Ministry of Defense, Manchester City council's system and police network, and the German army systems.
- *Zbots* is a notorious botnet that is still active and it is built with a Trojan called Zeus. Zeus mainly spreads through drive-by-downloads and Phishing scams where the unsuspecting user is tricked into clicking a link. A 2009 finding suggests that Zeus had compromised over 74,000 FTP accounts on websites of companies such as the Bank of America, NASA, Monster.com, ABC, Oracle, Play.com, Cisco, Amazon, and Business Week. Upon execution the Trojan automatically gathers any Internet Explorer, FTP, or POP3 passwords that are contained within Protected Storage (PStore).

3 Technical Overview and Botnet Architecture

The four main components to building and sustaining a botnet is to create it, propagate the malware to recruit newer victims, obfuscate itself from detection and take-over, and finally establish a means of controlling the victims.

3.1 Building a Botnet

The botmaster exploits the vulnerabilities in a victim's machine to open a backdoor through which he can take control over the machine. A simple Google search can yield a number of botnet source codes and sophisticated tools which are used for various criminal activities based on owning a botnet. Operating system and browser vulnerabilities are also available for sale. The ones that are new and still unpatched by the OS vendor (also called zero-day exploits) are the most wanted by attackers. This attack procedure can be described in a five step process [4]:

1. *Probe* - As a first step the master probes a range of IP addresses and network services and waits for a response. Once it receives a response from one of the machines, a victim has been identified for exploitation
2. *Exploit* - Now the master can attempt a well known exploit in the vulnerable machine. For instance it can try a buffer overflow attack on port 135 (Windows RPC). Now a back door channel has been established to breach the victim's machine (In this case, through Windows RPC port 135)
3. *Force Binary Download* - Now the attacker forces the victim to execute script or *shellcode* that downloads the entire binaries from the master site and attaches itself to a well known service at the victim's machine (Like the svchost, which starts up every time the machine is rebooted).
4. *Communicate With Dictator* - The infected victim now establishes itself as a zombie by establishing connection to a botnet C&C server established over a variety of channels like IRC or HTTP (explained in more detail below.)

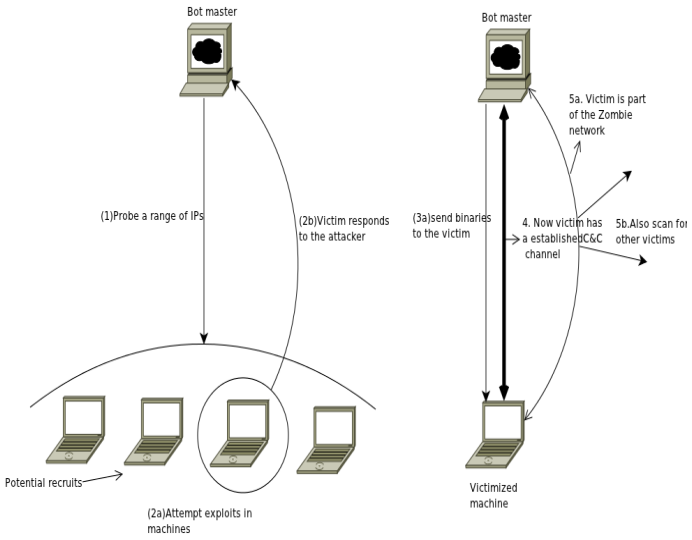


Figure 1: How a Victim is Recruited

5. *Establish and Scan* - As a final step in consolidating itself to the bot network, the victim finds a way to establish a dedicated port to communicate with C&C server and receive regular updates and modules to add to its malicious capabilities and then starts its outbound scan.These sequence of steps are shown in *Figure 1*.

3.2 Propagation and Obfuscation

The master and zombies have to actively scan for victims. One common way of spreading is to send spam emails with intriguing links which lead to websites where the Trojans are embedded. The attackers generally trick people into installing malware by attaching them to a crack tool for games etc. Most common way to is to exploit vulnerabilities in operating systems, instant messaging systems, browsers, and email clients. The members of a botnet generally use encryption and obfuscation while propagating or while communicating with the botmaster. The malware generally consist of an encryption routine and a key that encrypts the Trojan to defend detection through signatures. It also comes with an associated mutation engine that mutates the virus to produce different versions of same virus. The bots also use encoding techniques for maintaining their resources and also to communicate with the master. Storm worm, that we discussed earlier, maintains a list of infected P2P hosts that are encoded in hexadecimal. Other bots generally use various means of obfuscation in their communication - use of base64 encoding in GET requests is a common example.

3.3 Command and Control

Once the victim is exploited to be a part of the botnet, a command and control channel is established between the zombie and the master. The C&C architecture generally falls in the following categories;

IRC-Based C&C: One of the obvious means of communication is to directly establish a connection with the host, but that would compromise the identity of the botmaster. So a public message drop point can be used but it is involves a lot of latency. IRC channels turn out be the obvious choice which enables instant exchange point that delivers messages instantly [4].IRC's protocol is widely deployed in the Internet, has a text based command syntax and there are a large number of publicly available IRC channels. The IRC network consist of a number of IRC servers. Master and each of the bots connect to one of the IRC servers

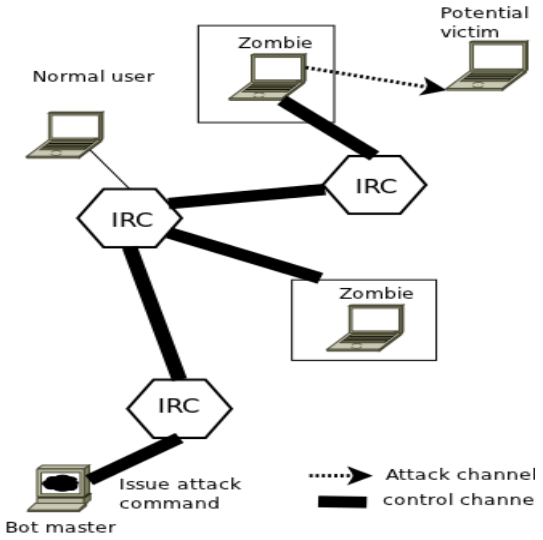


Figure 2: IRC C&C

as shown in *Figure 2*.

P2P C&C: IRC-based botnets described above have centralized master which is single point of failure for the entire botnet. Recent botnets have moved to P2P based architecture for C&C in which the botmaster can use any of the nodes to pass commands or collect information from other nodes in the botnet [9].This makes a sturdy architecture as even if a large portion of the botnet has been taken down, still the remaining portion of the network survives and manages to remain in communication with the master, but they are more difficult to modify and manage than the IRC based ones.

Web-based C&C: Botnets have now evolved to use HTTP and HTTPS protocols for C&C. The bots can now talk to a web server acting as their master. Using Web based protocols gives distinct advantage to the adversary as HTTP ports are always enabled and this C&C merges well with the normal traffic to provide obscurity. To avoid a single point of failure (which is the web server here) the adversary usually uses more than one server to act as master or use random domains which are updated from time to time.

4 A Closer Look at The Anatomy of Well Known Bots

In this section we describe the anatomy and mechanisms employed by well known malwares to open a back door for C&C, mode of control over the bots and also to obscure their presence to escape detection [6].

4.1 Architecture

- *AgoBot*: AgoBot also known as Phatbot is one of the oldest known bots. It has close to 20,000 lines of code in C/C++. It is an IRC based bot with a huge arsenal of exploits with the ability to launch DDoS attacks and harvest passwords through key logging and traffic sniffing.
- *SDBot*: SDBot is also an old bot known since 2002.Today this bot has hundreds of variants providing a wide range of capabilities but the core code is very compact when compared to AgoBot with just 2000 lines of C code. The design is in such a way that extension of code to add a newer capability is very straightforward and also diffuses accountability of the creator.

4.2 Botnet Control Mechanism

- *AgoBot*: AgoBot uses IRC for C&C. It uses set of IRC commands as wells as specific commands developed for this bot. The bot command set includes commands that make the bot to perform specific tasks like some of the example task commands shown in the following table:

Command	Function
bot.execute	Makes the bot execute a specific .exe
bot.sysinfo	Echo the bots system information
bot.status	Echo bot status information
bot.nick	Changes the nickname of the bot
bot.open	Opens a specified file
bot.remove	Removes the bot from the host

  In addition to commands, AgoBot also uses a lot of control variables to decide on the capabilities of the bot. For instance DDoS_max_threads variable, if set, will direct the bot to flood the specified host with SYN packets.

- *SDBot*: SDBot uses a relatively straight forward set of IRC commands to exercise control. The steps that a bot follows to contact a server are *(i)* Send the nick name and user name to server *(ii)* Respond to a PING command from master with a PONG and expect for a return code from the master *(iii)* Send a JOIN request to the host and when the connection is established, expect for commands from the server.

4.3 Host Control Mechanism

- *AgoBot:* The set of host control mechanism possessed by AgoBot is quite comprehensive. It can be widely categorized into *(i)* Secure the system, i.e. take countermeasures to prevent other attacks like patching the vulnerabilities.*(ii)* A set of *harvest* commands to harvest sensitive information *(iii)* A list of *pctrl* commands that lists and kills processes running on victim machine *(iv)* A list of *inst* commands to add or delete auto start entries

Type of command	Functions
harvest	capable of returning a list of CD keys, emails, AOL specific information, windows registry information etc.
pctrl	capable of returning set of all processes, kill a process, info about running services, stop specific services etc.
inst	capable of adding and deleting an auto start entry

- *SDBot:* The host control capabilities provided SDBot is somewhat limited to basic remote execution of commands and gathering some information from the victim site. Some of the commands include *download-* download a file from specified URL and execute it, *killthread-* kill a specified thread, *sysinfo-* list host system information, *execute-*run a specified program, *Update-* If the version of bot is different, download and upgrade the bot to the latest version.

4.4 Attack Mechanism

- *AgoBot:* AgoBot contains an elaborate set of exploits and attacks and they are well maintained and updated with every variant of AgoBot. It includes a number of scanners for worm backdoors. It also includes brute forcing capabilities for opening NetBIOS shares and brute forcing the passwords of open SQL servers. It also comes with capabilities of performing DDoS attacks on specified host.

- *SDBot :* SDBot's core capabilities are relatively benign (to give its creator a chance to disown any responsibilities) but it was easy to add newer modules to increase its capabilities. SDBot includes modules for sending UDP and ICMP packets to specified host which can be controlled using a simple command of the following type:
  udp/ping <host to attack> <port no. of packets> <packet size>

5 Detection and Prevention

Detection and prevention of Botnets is a challenging task. Let's walk through some of the techniques.

5.1 Anomaly Based Detection

The Botnet C&C master generally performs active scanning of the network to evaluate vulnerabilities and recruit potential victims. This scanning involves sending TCP SYN and other control packets to find open ports. This fact can used to identify anomalous behavior in network by forming what is called a *TCP work weight*.

$$w = (\mathrm{SYN_n} + \mathrm{ACK_n} + \mathrm{FIN_n})/\mathrm{TCP_n}$$

Where,
w = TCP work weight, $\mathrm{SYN_n}$ = number of SYN packets, $\mathrm{ACK_n}$ = number of ACK packets, $\mathrm{FIN_n}$ = number of FIN packets, $\mathrm{TCP_n}$ = Total number of TCP packets in the sample period.

A large value of TCP work weight means the packets are dominated by control packets which are typical to Botnet master scanning for victims. If such scan packets can be traced to a IRC host, most probably a Botmaster has been identified. But recent Botnets use what is called **"Idle scanning"**. Imagine a master M wants to scan potential victim P, he spoofs his IP address to B - a bot (who will eventually take the blame of scanning) and sends a SYN request to P, the potential. P thinks B has sent a SYN request to it and either responds with a SYN+ACK or RST depending on whether the port was open or not repectively. Now if B gets a SYN+ACK, it'll respond with a RST as it had no idea about the connection. Now the master can easily find the IPID of the reset packet of the bot B(Note that IPID is incremented with every packet sent out the system) which will show what P responded to B and hence whether P's port was open. Now the detection becomes tricky.
In such situations, instead of tracking the host with anomalous packet behavior, we can form a *Host Exposure Map* which captures the host-port combinations of the connections in which the host generally involves. This data should be obtained by initially training the system and capturing the pattern. Now any activity on the host which doesn't fall in the Exposure Map can be reported and attacker tracked down.

5.2 Detection by Dialogue Co-relation

The victimized host goes into specific states during attack and initial establishment of C&C channel. So a correlation engine which models these specific states can be used in the network to detect malicious communications [4]. To be more specific, the packet scanner will look for initial inbound scanning, using an exploit, binary download, coordination with the master and outbound attack propagation. The model is described in *Figure 3*

The dialogue co-relation engine sits at the perimeter of the network and make use of the services of *Intrusion Detection Systems (IDS)* to analyse the contents of payload against malicious signatures and also to determine inbound and outbound scan patterns. It keeps track of the various stages in the infection trail, and reports an alert when the interaction model fits into malicious pattern.

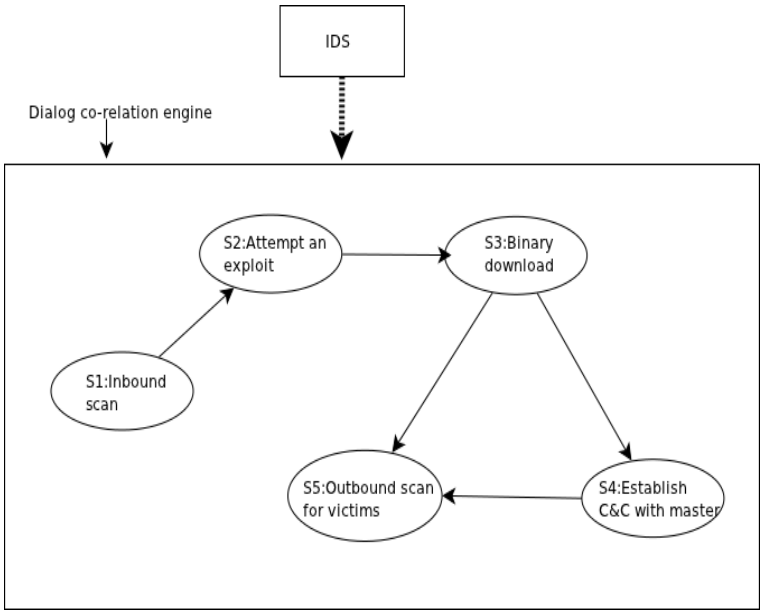


Figure 3: Dialog co-relation for Botnet detection

5.3 P2P Botnet Detection

We have seen how P2P botnets differ from regular botnets in earlier section. One of the effective ways of P2P botnet detection is to monitor the target P2P network and form statistical fingerprints which profiles the different P2P applications including the legitimate ones (like Skype) and cluster based analysis which profiles the hosts. This can be devised as a two step process-

1. First process involves detection of hosts in the network that involve in P2P communication. This can be done by filtering the packets of each host and extracting a number of statistical values which will isolate the P2P hosts.

2. Now the second phase involves separation of legitimate P2P hosts from the malicious ones [11].

This can be done by the following parameters:

- *persistence pattern :* There is usually a difference in the amount of time the legitimate P2P user and a botnet host remain active in the P2P network. For instance a file sharing application may be closed as soon as the job is done, but a bot remains active as long as the system is up.

- *interaction pattern :* The botnet peers also follow a pattern in interaction including using the same protocol, and the large overlap in the number of peers contacted by two different bots.

The P2P detection system is intuitively described in *Figure 4*.

6 Evolution of Botnets

The botnet architecture has evolved over time with the attacker's modus operandi being well known to the security community and also with the sophistication of tools and computing resources available to the attacker community. Initially the attacker relied preliminarily on security through obscurity as the existence

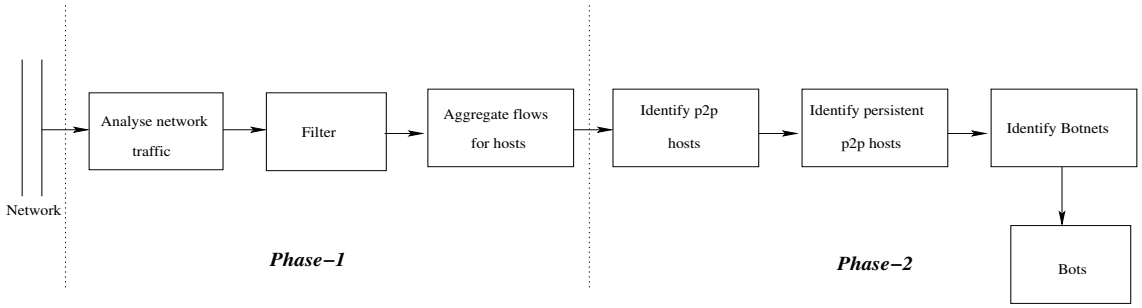


Figure 4: P2P botnet detection

of botnets was not widely known. Attackers used *IRC channels* to command and control the zombies. This was very insecure as it was possible to bust them easily and also for other adversaries from the attacker community to take control of their botnets [1]. Despite its weaknesses, the classic architecture has its strength in ease of setup and maintenance, widely and freely available infrastructure for botnet setup, support from the expert hacker community, and the ability to fully control and track his bot army.

6.1 From Chaotic to Professional

Today's botnets have moved from public IRC networks to *privately hosted servers*. To prevent adversaries from take-over [8], the bots are designed to ignore any commands that do not contain a secret pass-phrase as the prefix of the commands. Now the botmaster can control the victims with *HTTP* rather than IRC. The victim now loses the option of shutting down the IRC port to prevent contact with command and control center as shutting the HTTP port down is practically infeasible. They also have fancy GUIs with mining and querying capabilities that the botmaster can choose to send spam mails from zombies in a particular country. It also has ability to send pre-formatted SQL queries, sending data and receiving commands over HTTP. Most of them are so sophisticated and work at such low levels that they could record the HTTP payloads before they are actually encrypted.

A notorious hack kit called *Mpack* provides a classic example of second generation botnets [1] [5]. *The Mpack kit* has an arsenal of scripts that could detect the recruit's OS, browser information, location, other software installed in the system and from the library of exploits available with the kit, could then decide the best means to bring down the recruit. Thus the newer generation bots don't actively scan for recruits but instead they lure recruits and turn them into honeypots for further multiplication. Also with the explosive growth of smart phone users, mobiles and smart phones are turning out to be one of the major platforms for botnets. Such devices are almost always online and have more and more functionalities and processing power built in them. Such devices are connected to computers to synchronize contents and this could be exploited to spread and update malware [10] [7]. The Figure 5 depicts the evolution in the architecture of the modern botnets.

7 Conclusion

Having evolved with technology and time, the botnets still remain the greatest threat to the security of networked computing resources. The botnet kits of today use every conceivable form of attack with a score of malwares, worms, Trojans, root kits, spam engines that analyze the vulnerabilities of computers before

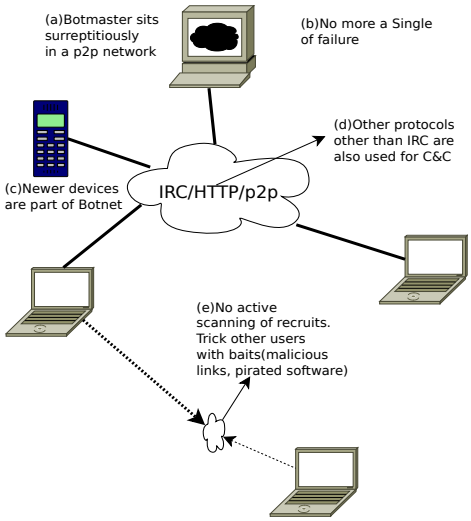


Figure 5: Botnet evolution

attacking them. The fight against botnets starts from a very rudimentary level at the user's general awareness of threats to his system posed by Malware and botnets. Majority of the attacks are due to ignorance and indifference to the actual threat. The following can be considered as general guideline for basic safety against malware:

- Stay aware and updated about the threats and spread the *awareness*
- Install *security* updates for OS, browser etc. promptly
- Don't visit untrusted links or visit unknown websites and get infected.
- Avoid using peer-to-peer software as much as possible.
- *Block JavaScript*-Change your settings to enable browser prompt before executing any script.
- *Watch your ports* for unexpected inbound and outbound traffic. Disable all unused, unnecessary ports.

With these general personal awareness and discipline the spread of botnets can be controlled to a large extent. A sufficiently large botnet can pose serious threat to the security of an entire nation. In the era where computers run everything from businesses to governments, it's time to take computer security against botnets seriously and carry out sufficient research in this area and not letting the attacker dictate terms.

References

[1] Zheng Bu, Pedro Bueno, and Rahul Kashyap. The new era of botnets. *White paper from McAfee*.

[2] Evan Cooke, Farnam Jahanian, and Danny McPherson. The zombie roundup:understanding, detecting, and disrupting botnets. *SRUTI 05, USENIX association*, 2005.

[3] N. Daswani, M. Stoppelman, the Google Click Quality, and Security Teams. The anatomy of clickbot. *USENIX Hotbots07*, 2007.

[4] Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. Bothunter: Detecting malware infection through ids-driven dialog correlation. *In Proceedings of the 16th USENIX Security Symposium (Security'07)*, Aug 2007.

[5] Corey Nachreiner and Scott Pinzon. Understanding and blocking the new botnets. *white paper from LiveSecurity*, April 2008.

[6] P.Barford and V.Yegneswaran. An inside look at botnets. *Special Workshop on Malware Detection,Advances in Information Security, Springer Verlag*, 2006.

[7] Kapil Singh, Samrit Sangal, Nehil Jain, Patrick Traynor, and Wenke Lee. Evaluating bluetooth as a medium for botnet command and control. *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, July 2010.

[8] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. Your botnet is my botnet: Analysis of a botnet takeover. *CCS '09 Proceedings of the 16th ACM conference on Computer and communications security*, 2009.

[9] Ping Wang, Sherri Sparks, and Cliff C. Zou. An advanced hybrid peer-to-peer botnet. *In Proceedings of the First Workshop on Hot Topics in Understanding Botnets*, 2007.

[10] Cui Xiang, Fang Binxing, Yin Lihua Liu, Xiaoyi, and Zang Tianning. Andbot: Towards advanced mobile botnets. *In Proceedings of the 4th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, March 2011.

[11] Junjie Zhang, Roberto Perdisci, Wenke Lee, Unum Sarfraz, and Xiapu Luo. Detecting stealthy p2p botnets using statistical traffic fingerprints. *IEEE/IFIP 41st International Conference on Dependable Systems and Networks*, page June, 2011.

The Stuxnet Worm

Paul Mueller and Babak Yadegari

1 Overview of Stuxnet

Stuxnet is a sophisticated worm designed to target only specific Siemens SCADA (industrial control) systems.

It makes use of an unprecedented four 0-day vulnerabilities- attacks that make use of a security vulnerability in an application, before the vulnerability is known to the application's developers.

It also uses rootkits - advanced techniques to hide itself from users and anti-malware software - on both Windows and the control computers it targets. It employs two stolen digital certificates to sign its drivers, and its creators needed a large amount of knowledge of its targeted systems. See Figure 1 for an overview.

It was discovered in June 2010, but an early version appeared a year earlier. It is widely suspected of targeting Iran's uranium enrichment program, since it is rather specific about what it attacks, and this matches the Iranian Natanz enrichment plant.

One indication that Stuxnet targeted Iran's nuclear program is that it only targets facilities that have a certain specific physical layout. The layout of the centrifuges in a facility such as Natanz is called a cascade, and describes how the centrifuges are piped together; this is done in stages, and the centrifuges within one stage are piped in parallel.

Iranian President Ahmadinejad visited Natanz on April 8, 2008, and photos of the visit were published on his website as a photo-op. In one of the photos (Figure 2) is what appears to be a monitor showing the structure of a so-called IR-1 (for Iran-1 centrifuge) cascade. This structure, giving the number of centrifuges in each stage, matches the Stuxnet code exactly.

Iran's nuclear program launched in the 1950s with the Shah of Iran obtaining non weapons-related assistance from the United States' "Atoms for Peace" program. The program's inception was delayed because of the 1979 revolution and after that because of the Iran-Iraq war. However, Iran's new leaders were interested in continuing the nuclear program and started getting help from other countries to further it.

In 2002, an Iranian opposition group publicly revealed two undeclared nuclear facilities, resulting in Iran admitting to having constructed facilities for fuel enrichment and heavy water production, ostensibly for use in research reactors. Iran suspended its plans in 2003 but resumed them in 2006 and insists that it has the right to establish its own uranium enrichment program.

Iran maintains that its nuclear program is entirely peaceful. However, the IAEA (International Atomic Energy Agency) insists that Iran does not comply with the safeguard program it has agreed to, resulting in various sanctions against Iran by the UN Security Council [9]. It is widely believed that Iran is in fact working toward producing nuclear weapons.

Assuming that Stuxnet was intended to damage this suspected nuclear weapons program, it was somewhat effective: it may have destroyed 1,000 centrifuges at Natanz, about 11% of the total number installed at the time. Also, Iran doesn't have an unlimited number of centrifuges, and the ones they do have tend to fail relatively often, so such a decrease is significant, albeit not

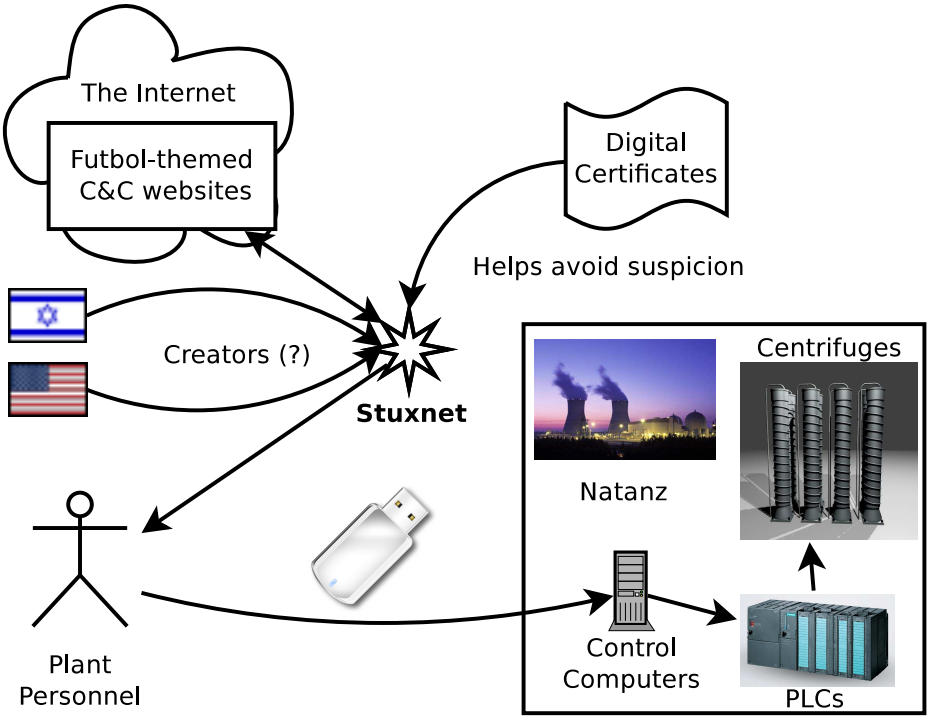


Figure 1: A high-level overview of Stuxnet.

Figure 2: Iran's president revealed the cascade structure at Natanz: The dark lines at the bottom probably denote the division between stages; yellow lines have been added above them to make the division clearer. from right to left- 4, 8, 12, 16, 20, 24, 20, 16. (Photo: Office of the Presidency of the Islamic Republic of Iran)

immediately fatal to the program. In addition, Stuxnet decreased production of enriched uranium and likely sowed chaos within the Iranian nuclear program.

Israel and the U.S. are the leading suspects as Stuxnet's creators. Neither are friends of Iran's current leadership (to put it mildly) and both, especially Israel, fear a nuclear-armed Iran.

Israel has said that cyberwarfare is an important part of its defense strategy, and has a military intelligence unit dedicated to it, according to Wikipedia. Furthermore, Israeli officials are said to have responded with "wide smiles" when they were asked in 2010 whether Israel created Stuxnet [5].

Furthermore, a video celebrating the operational successes of the head of the Israeli Defence Forces' Lieutenant General Gabi Ashkenazi, shown at his retirement party, showed Stuxnet as being among them, according to the Daily Telegraph [15].

Before Stuxnet had been discovered, according to Wikipedia, "John Bumgarner, a former intelligence officer and member of the United States Cyber-Consequences Unit (US-CCU)" had published an article describing a Stuxnet-like attack on centrifuges, and claimed that such attacks against nations enriching uranium in violation of international treaties are legitimate. This, combined with some US officials' not-quite-denials of involvement, raise suspicions that the US participated in Stuxnet's creation, despite official denials.

2 Operation of Stuxnet

One thing that differentiates Stuxnet from more run-of-the-mill malicious software is that its creators have incorporated lots of capabilities into it. These range from exploiting multiple zero-day vulnerabilities, modifying system libraries, attacking Step7 installations (Siemens' SCADA control software) and running an RPC server, to installing signed drivers on Windows operating systems. Figure 3 shows an overview of multiple components which are present in the malware. This section describes these components and their various purposes.

Stuxnet spreads via several vectors, no doubt selected to ultimately allow it to infect the PLCs it targets. It is capable of auto-updating, so that it can update old versions of itself to newer versions available on a local network. It communicates with command and control servers to provide information on its spread to its creators and also provide another way for it to be updated. It conceals its presence and the source of its destructive effects from plant personnel, who may be totally unaware that it is the cause of unexplained problems.

2.1 How Stuxnet Spreads

Stuxnet spreads readily, but it also contains safeguards to limit its spread. It only infects three computers from a given infected flash drive and is hardcoded to stop spreading itself after June 24, 2012.

As illustrated in Figure 4, Stuxnet employs several methods to spread itself:

Via USB flash drives The ultimate destination of Stuxnet is the computers that control the centrifuges. These are called PLCs (Programmable Logic Controllers), and are special-purpose computers, used for controlling electronic devices or systems, such as industrial systems.

The PLCs are connected to computers that control and monitor them, and typically, neither are connected to the Internet. Therefore, Stuxnet needs some other vector to reach those computers, and so it is capable of propagating via USB flash drives.

In the case of Natanz, the infected flash drives may have been introduced to the control computers via outside contractors working at the plant.

Different versions of Stuxnet use different ways to do this: recent versions use an Windows LNK vulnerability and older versions use an `autorun.inf` file vulnerability.

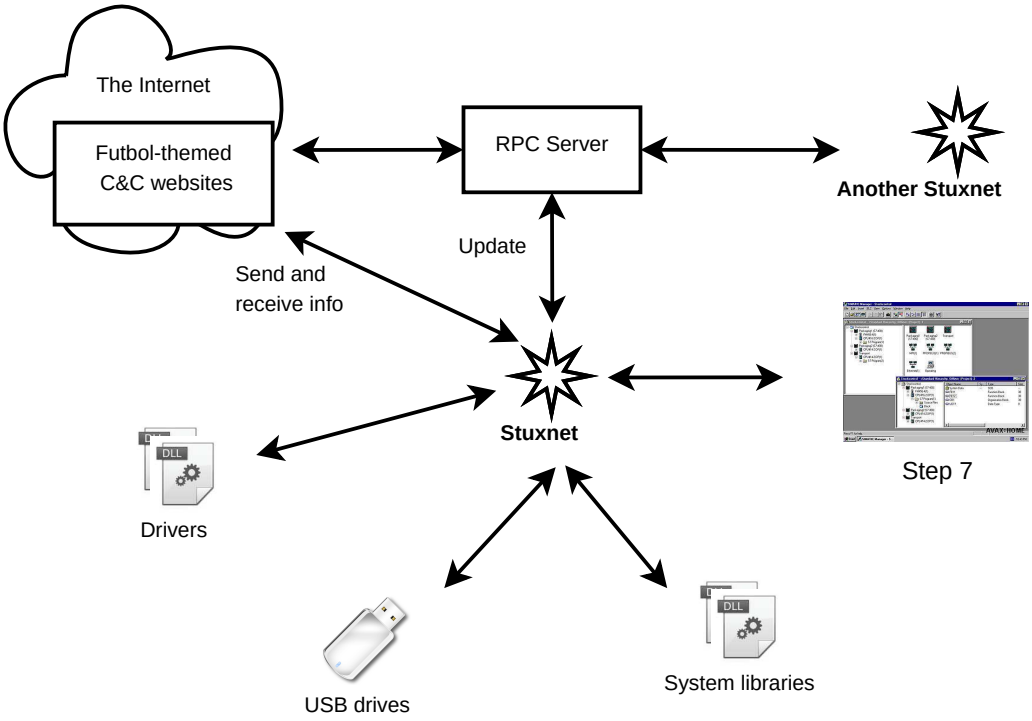


Figure 3: Stuxnet's various components

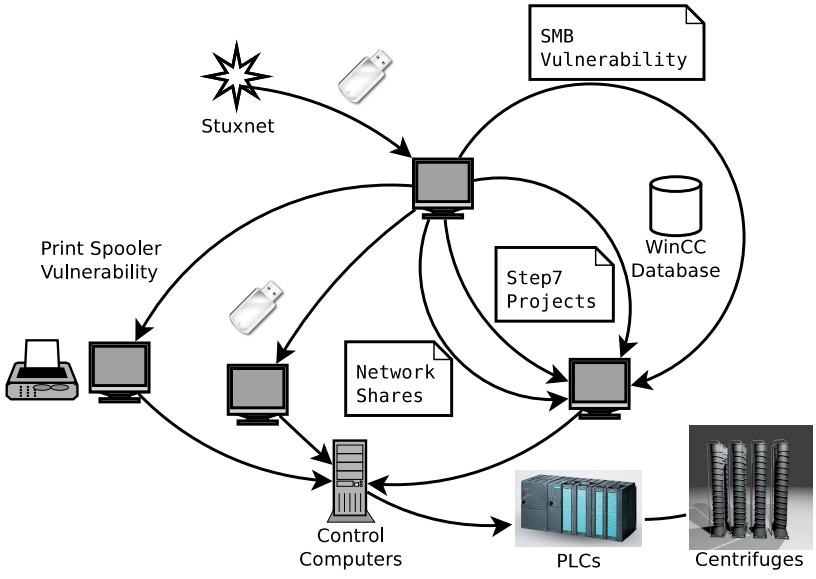


Figure 4: Stuxnet employs many ways to reach its target PLCs.

- LNK vulnerability (CVE-2010-2568)
  Stuxnet registers code to an infected Windows computer that, upon a USB drive being inserted, copies Stuxnet to the drive. Interestingly, an existing copy of Stuxnet on the external drive will be removed if that drive has already infected three computers.

  In addition to the Stuxnet DLL and a loader for it, the malware creates four `.lnk` files on the removable drive. These are used to execute the loader when a user views the drive; four are needed in order to target different versions of Windows.

- `autorun.inf` file
  An `autorun.inf` file is a file that causes Windows to automatically run a file on removable media when that media is inserted into the computer. Older versions of Stuxnet place an `autorun.inf` file on flash drives that are inserted into an infected computer.

  However, instead of using a separate file, it inserts the code for itself directly into the `autorun` file, along with valid commands to infect the computer using this code. Windows ignores the Stuxnet data portion, since it ignores invalid commands in an `autorun.inf` file.

Via WinCC Stuxnet searches for computers running Siemens WinCC, an interface to their SCADA systems. It connects using a password hardcoded into WinCC, and attacks its database using SQL commands to upload and start a copy of itself on the WinCC computer.

Via network shares Stuxnet can use Windows shared folders to propagate itself over a local network. It places a dropper file on any shares on remote computers, and schedules a task to execute it. ESET [11] says the task is scheduled to run the next day, whereas Symantec [7] claims it is scheduled for two minutes after the file is shared.

Via the MS10-061 print spooler 0-day vulnerability Stuxnet copies itself, places the copy on remote computers via this vulnerability, and then executes the copy, thereby infecting the remote machines. In brief, Stuxnet "prints" itself to two files in the `%system%` directory on each target machine, using the 0-day privilege escalation. It then executes the dropper file to infect the computer.

Via the MS08-067 SMB vulnerability If a remote computer has this vulnerability, Stuxnet can send a malformed path over SMB (a protocol for sharing files and other resources between computers); this allows it to execute arbitrary code on the remote machine, thereby propagating itself to it.

Via Step7 Projects Stuxnet will infect Siemens SIMATIC Step7 industrial control projects that are opened on an infected computer. It does this by modifying DLLs (Windows Dynamic Link Library; a library of shared objects: code, data, and resources) and an `.exe` file in the WinCC Simatic manager, so that they execute Stuxnet code as well. The additional code will insert Stuxnet into Step7 project directories.

2.2 Auto-updating

Stuxnet can update itself from infected Step7 projects. If an infected project is opened, and its version of Stuxnet is newer than the one already on the computer, the one on the computer will be updated.

Additionally, Stuxnet uses a built-in peer-to-peer network to update old instances of itself to the latest version present on a local network. Each copy starts an RPC (Remote Procedure Call) server, and listens for connections. Other instances, connecting via their RPC clients, are able to update themselves if their version is older, or update the server if it is older.

2.3 Command and Control servers

After Stuxnet establishes itself on a computer, it tries to contact one of two servers via HTTP:

- www.mypremierfutbol.com
- www.todaysfutbol.com

It sends its IP address, some unknown data, and a payload consisting of, in part, information on the host OS, the host computer name and domain name, and a flag indicating if Siemens Step7 or WinCC, which Stuxnet targets, is installed [7].

The server may respond by sending a Windows executable, which it can specify to be loaded into the current process or a different one via RPC. This allows Stuxnet's authors to update it remotely, or to run entirely new malware on the infected computers.

Interestingly, the data sent to and from the server is encrypted, each way with a different 31 byte key (of seemingly random bytes) that is XORed with the data. However, both keys are static, and so don't provide much protection against someone who has intercepted multiple such communications.

The command and control servers have since been rendered impotent by redirecting their DNS entries (initially to nowhere and later to fake servers, set up by Symantec, to gather information on the worm.)

According to Symantec, Stuxnet is able to update itself to communicate with new command and control servers, but this hasn't been observed in the field.

2.4 Infection

The malware's main module consists of both user-mode and kernel-mode components. The user-mode functions are mainly designed to do several things: 1) inject it into a chosen process - add its own code into a running process, which results in the execution of that code in the target process's address space; 2) check for the appropriate platform; 3) escalate privileges; and 4) install two kernel-mode drivers, one for running Stuxnet after reboot and the other as a rootkit to hide its files.

2.4.1 User-Mode

The main module DLL exports 21 functions (Figure 5). Stuxnet starts by calling export 15. In this function, it first checks if it is running on an appropriate Windows version. Assuming it is and the machine is not infected already, it uses one of the zero-day exploits, depending on the version of Windows, to elevate its privileges.

Export 16 is then called to continue installation: it injects code into the `services.exe` process to infect removable drives and infects any Step7 projects it finds. It then checks a registry value and aborts the infection if it matches. It also checks to ensure the current date is not later than June 24, 2012, and checks if it is running the latest version. If all these checks pass, it will drop two driver files to install the driver files, `Mrxnet.sys` and `Mrxcls.sys` [7].

It is unclear if there is any special significance to the cutoff date of June 24th; it may be arbitrary, used to stop Stuxnet's spread after it had (presumably) already done "enough" damage.

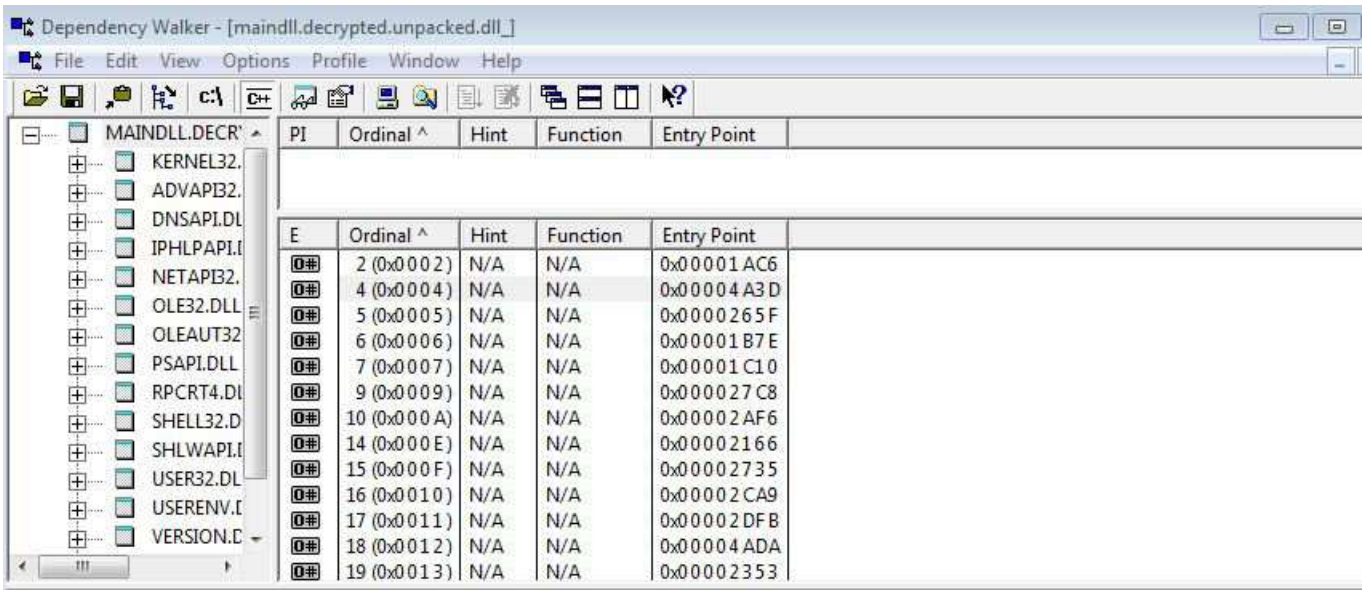


Figure 5: Stuxnet's main library export functions as shown by Dependency Walker

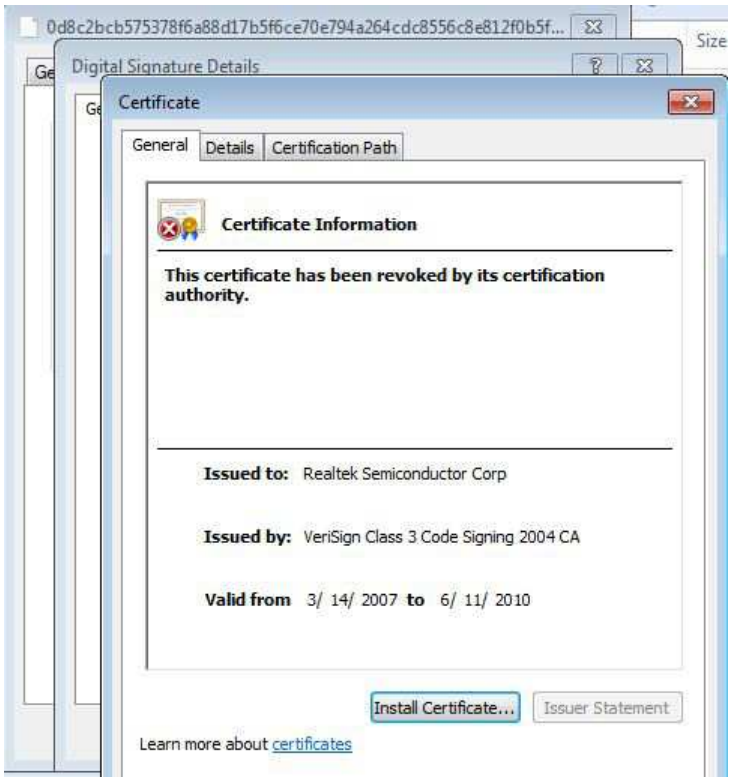


Figure 6: Stuxnet installs two drivers signed with stolen signatures

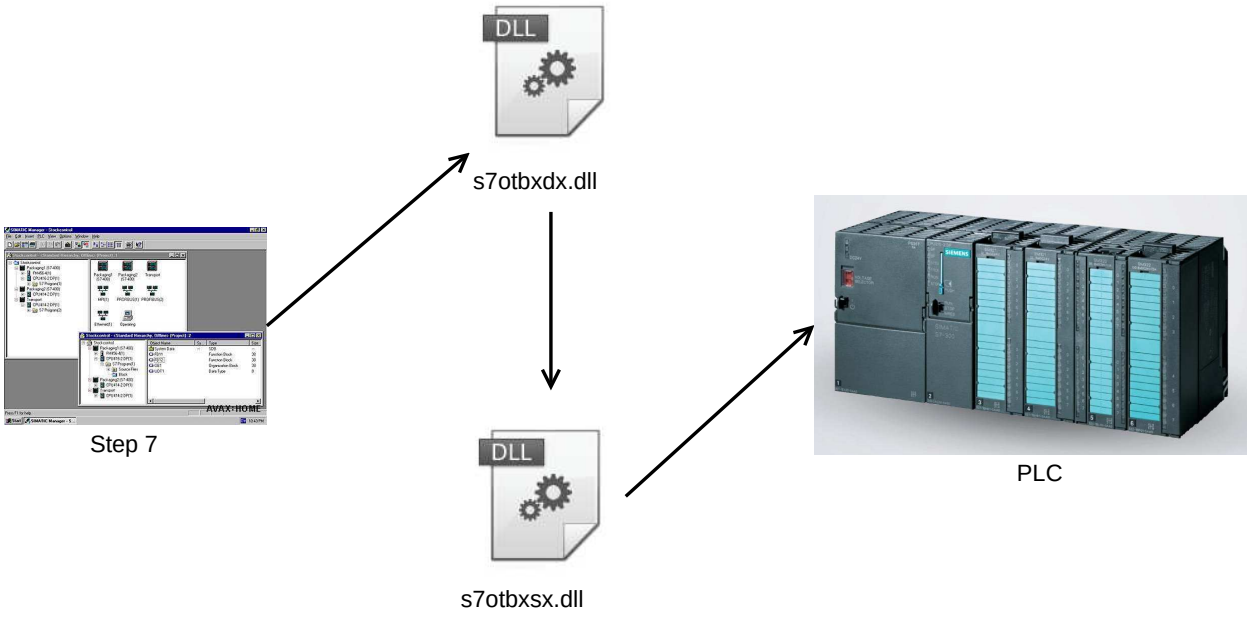


Figure 7: Stuxnet wraps the library used to communicate with the PLCs

2.4.2 Kernel-Mode

Stuxnet installs two kernel-mode drivers. `Mrxcls.sys` is a driver signed by a Realtek certificate as shown in Figure 6. When Stuxnet wants to install it onto the system, it marks it as a boot startup so it starts in the early stages of Windows boot. This driver first reads a registry key which has been written in the installation step and contains the information for injecting Stuxnet images into certain processes.

The other driver, `Mrxnet.sys`, is actually the rootkit and is also digitally signed by a Realtek certificate. It creates a device object and attaches it to the system's device objects so that it can monitor all requests sent to these objects. The purpose of this job is to hide files which meet certain criteria from users.

2.5 Attack phase

Stuxnet is not harmful to ordinary users since its aim is to modify Simatic PLCs manufactured by Siemens [14]. Step7 programs, which control and monitor these PLCs, use a library named `s7otbxdx.dll` to communicate with the actual PLC to read or modify its contents (codes). Stuxnet gets control over all requests sent to the PLC by wrapping this library.

In addition, Stuxnet writes its own malicious code to target certain specific PLCs. It hides its malicious code from users by returning original code blocks instead of the modified blocks upon a read request, as illustrated in Figure 7 [7].

The code Stuxnet infects PLCs with contains three attack sequences, named A, B, and C in Symantec's report. Sequences A and B are similar, with only slight differences, and have the same effects. Sequence C is more advanced but is incomplete and is never executed.

Sequences A and B perform their attacks by running the centrifuge rotors at too-low and too-high frequencies (as low and high as 2 and 1410 Hz, respectively). Interestingly, the periods in which they command the centrifuges to spin at these inappropriate speeds are quite short (50 and 15 minutes, respectively), and are separated by about 27 days between attacks, possibly indicating

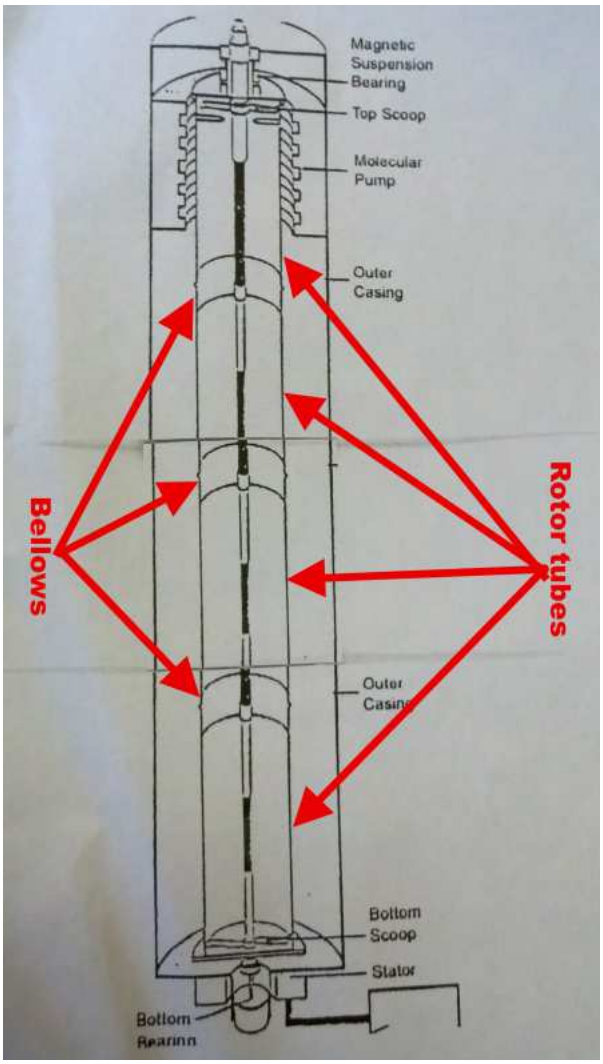


Figure 8: Diagram of a P-1 centrifuge. The Natanz centrifuges are based on the P-1. (Diagram: Institute for Science and International Security)

the designers wanted Stuxnet to operate very stealthily over long periods of time. See Figure 8 for a diagram of a centrifuge similar to the ones used at Natanz.

Although the time spans during which the centrifuges are slowed down or sped up are probably too short for them to actually reach the minimum and maximum values, they still result in significant slow-downs and speed-ups. The slow speeds are sufficient to result in inefficiently processed uranium, and the high speeds are probably sufficient to result in centrifuges actually being destroyed, as they are at the centrifuges' maximum speed limit.

2.6 Concealment of attack activities

Stuxnet hides itself from plant personnel by installing rootkits on infected Windows computers and on infected PLCs, in order to hide its files. By installing a driver on Windows computers, it hides itself by manipulating requests sent to devices.

By intercepting calls to `s7otbxdx.dll`, Stuxnet hides the malicious code it writes to PLCs to

sabotage centrifuges, and also prevents those malicious codes from being overwritten.

Before the malware runs an attack routine, it records the centrifuges' normal operating frequencies and feeds this recorded data to the WinCC monitor program during the attack. The result is that the system shows normal operation instead of alerting personnel to the anomalous frequencies the centrifuges are actually running at.

Stuxnet also modifies some routines on the PLCs, preventing a safe shutdown even if the operator finds out that the system is not operating normally [7].

3 Conclusion

Stuxnet is a malware of such sophistication that it is most likely the work of one or more nations, with Israel and/or the U.S. being the presumed author(s). It spread around the globe, but most of its infections were in Iran, and it seems that the only place it activated its payload was Natanz.

Now, almost two years after Stuxnet's discovery, Iran appears to have purged it from their Natanz equipment, according to a Reuters article [8]. The article was published in February 2012, but states that it is unknown exactly when the Iranians succeeded in clearing their systems of the infection, so they may have been clear for some time now.

It is likely that Stuxnet would have had a greater impact had it not been noticed by security researchers, who subsequently published detailed reports on it (See [7] and [11] for two good examples). It probably destroyed about 1,000 centrifuges and delayed Iran's nuclear weapon program, but likely didn't have as much impact as its creators had hoped for.

Now that the Iranians are on the alert for Stuxnet in specific and such threats in general, it will likely be harder to pull off another such attack against them. Thus, the best chance to derail or at least significantly delay Iran's nuclear weapons program has probably passed, which leaves us with the worrying prospect of Israel, with or without U.S. aid, militarily attacking Iran.

Stuxnet has also increased awareness of the vulnerability of industrial control systems, which haven't been the target of many attacks. This should result in them becoming more hardened against attack as time goes on, but this is balanced against the increased risk of such attacks.

Stuxnet has increased the likelihood that malware authors, be they nation-states or smaller entities, will perpetrate similar attacks in the future. It has proven such attacks possible, raised awareness of them and perhaps interest in them among malicious entities, and provided a sophisticated code base for malware authors to study and modify.

Presumably, assuming the U.S. and/or Israel were Stuxnet's creators, they did a cost-benefit analysis taking into account its potential to deter Iran's nuclear ambitions against the increased risk of similar attack against themselves and others. They must have concluded that the risk was worth it. Hopefully they won't be proven wrong in the coming years.

4 Overview of References

Symantec has produced a detailed, highly-recommended examination of Stuxnet [7]. ESET has produced a similarly detailed report, although with a somewhat different focus [11]. The Langner company, headed by Ralph Langner, was instrumental in deciphering Stuxnet. There is a lot of interesting information on Stuxnet and other topics on their blog [6]. Wikipedia has a good page on Stuxnet [5]. ISIS' website [1] has a wealth of information on Stuxnet, Iran's nuclear program, and other interesting topics. Finally, for some lighter reading, Wired.com has an interesting story about how Symantec, Langner, and others figured out what Stuxnet was [16].

References

[1] The website of isis. Technical report, World Wide Web, http://www.isis-online.org.

[2] David Albright, Paul Brannan, and Christina Walrond. Did stuxnet take out 1,000 centrifuges at the natanz enrichment plant? Technical report, World Wide Web, http://isis-online.org/uploads/isis-reports/documents/stuxnet_FEP_22Dec%2010.pdf, December 2010.

[3] David Albright, Paul Brannan, and Christina Walrond. Stuxnet malware and natanz: Update of isis december 22, 2010 report. Technical report, World Wide Web, http://isis-online.org/uploads/isis-reports/documents/stuxnet_update_15%Feb2011.pdf, February 2011.

[4] Mark Clayton. Stuxnet cyberweapon looks to be one on a production line, researchers say. Technical report, World Wide Web, http://www.csmonitor.com/USA/2012/0106/Stuxnet-cyberweapon-looks-to-be-%one-on-a-production-line-researchers-say, January 2012.

[5] Contributors. Stuxnet. Technical report, World Wide Web, http://en.wikipedia.org/wiki/Stuxnet.

[6] Ralph Langner et. al. The blog of langner.com. Technical report, World Wide Web, http://www.langner.com/en/blog/.

[7] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32.stuxnet dossier (version 1.4). Technical report, World Wide Web, http://www.symantec.com/content/en/us/enterprise/media/security_respons%e/whitepapers/w32_stuxnet_dossier.pdf, February 2011.

[8] Mark Hosenball. Experts say iran has "neutralized" stuxnet virus. Technical report, World Wide Web, http://www.reuters.com/article/2012/02/14/us-iran-usa-stuxnet-idUSTRE81%D24Q20120214, February 2012.

[9] Nuclear Threat Initiative. Iran's profile. Technical report, World Wide Web, http://www.nti.org/country-profiles/iran/nuclear/, March 2012.

[10] Ralph Langner and associates. The prez shows his cascade shape. Technical report, World Wide Web, http://www.langner.com/en/2011/12/07/the-prez-shows-his-cascade-shape/, December 2011.

[11] Aleksandr Matrosov, Eugene Rodionov, David Harley, and Juraj Malcho. Stuxnet under the microscope (revision 1.31). Technical report, World Wide Web, go.eset.com/us/resources/white-papers/Stuxnet_Under_the_Microscope.pdf.

[12] Symantec Security Response. W32.duqu - the precursor to the next stuxnet. Technical report, World Wide Web, http://www.symantec.com/connect/w32_duqu_precursor_next_stuxnet, October 2011.

[13] Mark Russinovich. Analyzing a stuxnet infection with the sysinternals tools, part 1. Technical report, World Wide Web, http://blogs.technet.com/b/markrussinovich/archive/2011/03/30/3416253.a%spx, March 2011.

[14] Wikipedia. Simatic s5 plc. Technical report, World Wide Web, http://en.wikipedia.org/wiki/Simatic_S5_PLC/, February 2012.

[15] Christopher Williams. Israeli security chief celebrates stuxnet cyber attack. Technical report, World Wide Web, http://www.telegraph.co.uk/technology/news/8326274/Israeli-security-chi%ef-celebrates-Stuxnet-cyber-attack.html, February 2011.

[16] Kim Zetter. How digital detectives deciphered stuxnet, the most menacing malware in history. Technical report, World Wide Web, http://www.wired.com/threatlevel/2011/07/how-digital-detectives-deciphe%red-stuxnet/all/1, 2011.

5 Credits for Images Used in the Figures

- The US and Israel flag images come from the game Freeciv, and are licensed under the GPL (version 2).
- The nuclear power plant image is from MSN.com, via http://www.msnbc.msn.com/id/45524978/ns/technology_and_science-space/t/could-natural-nuclear-reactors-have-boosted-life/ (And yes, we know Natanz isn't actually a nuclear power plant).
- The PLC image is from alibaba.com
- The USB flash drive image is from psdgraphics.com
- The centrifuge image is from http://www.turbosquid.com/3d-models/blender-nuclear-centrifuge/663104

Hacking Networked Games

Matthew Ward and Paul Jennas II

Abstract

With the explosion of the game of poker came an explosion of on-line poker. The popularity of on-line poker boosted the popularity of on-line gambling in general. Even with the passage of the *Unlawful Internet Gambling Enforcement Act of 2006*, which prohibited banks and other financial institutions from allowing transactions with on-line gambling sites, on-line gambling worldwide remains a $30 billion dollar per year industry [2]. That kind of popularity and money often attract people who are willing to take the money by immoral or even illegal means. This paper will explore the various methods of cheating that threaten on-line gambling games.

1 Introduction

The on-line gaming industry as a whole has taken the world by storm with games such as World of Warcraft, Second Life, and Final Fantasy each amassing millions of monthly subscribers. Along with their success has also come a variety of techniques contrived by cheaters to gain an unfair advantage. In non-gambling games, new markets have been created where gamers are able to purchase virtual items from one another for real money. These virtual items might be virtual money, weapons, clothes, etc. The involvement of real money has led to increased focus on preventing cheating. On the other hand, one could argue that because these markets are outside of the realm of the game and even in many cases against the will of the game companies themselves, these markets should not have an impact on a gaming company's motivation to prevent cheating. This implies that it is the company's desire to provide an enjoyable experience for the gamer that should be the sole motivation. According to Ralph Koster, creative designer for *Star Wars Galaxies*, "any behaviour that hurts business is bad behaviour." [13] However in the world of on-line gambling, the exchange of money is at the heart of the game. It is therefore clear that the involvement of money in on-line gambling provides the driving force for protecting the integrity of the game.

Today many of the techniques employed in non-gambling games are now a threat to these on-line casino games. Through the process of generating the attack tree in Figure 1, which consist of these various methods, a somewhat natural grouping of the various methods resulted. While others would likely have variances in their attack tree upon completing the same exercise, it is clear that the different cheats have a definite relation and similarities to other cheats such that they can be grouped based on these similarities. In this paper we will discuss the major categories of cheating in on-line gambling games. In particular we will show each type of cheat, including information about known countermeasures and historical examples. Section 3 details the controversial use of bots. The different forms of DoS attacks are discussed in Section 4. Sections 5 and 6 cover collusion and software exploits. The paper then concludes in Section 7.

1.1 Poker Basics

This report focuses on exploiting on-line poker. The techniques described can be applied to any on-line game, but the financial importance of on-line poker make its an excellent candidate to study.

With this is mind a short tutorial on the basics of poker is required. Poker is a card game where card ranks (e.g Ace, King, Queen) and forming "hands" is used to determine a winner. A hand is a pre-defined combination of cards. For example, "three of a kind" is a hand where a player has three cards of the same rank. The rarer the odds of obtaining a hand the higher the rank of that hand. This makes poker a game of statistics. However, there is also a human element, as players can pretend to have certain hands (called a bluff) and good players must be able to tell what hand a player really has. There are many variations of poker (Texas Hold'em, Omaha, Stud, etc.), but the hand definitions are fairly standard across most variations. The differences lie in how the hands are formed. In Texas Hold'em each player is dealt two hole cards which only that player can see. Then there are 5 community cards that all players can see and use to form their hands. Typically community cards are revealed one at a time and after each card is dealt players take turn performing game actions. Typical actions include Bet, Check, Fold, Call, Raise. Players often make these decisions based on a concept called pot odds.

Pot odds essentially determine whether or not making a particular bet is profitable. To make this determination, a player will first calculate the ratio w of their chances of losing to their chances of winning. They will then compare this to the ratio p of the size of the pot to the size of the bet. If p is greater than w then this is a profitable bet, otherwise it is a losing bet. For example, let us imagine a situation where a player has a 20% chance of getting a card that will give them the winning hand but they are faced with needing to make a \$10 bet to see the next card. In this case, w is 4:1 (i.e. 80%:20%). Thus the pot size needs to be greater than \$40 in order for p to be greater than 4:1 (i.e. \$40/\$10). On average, for every 5 times the player is in this situation with a large enough p they will lose \$10 four times but win more than \$40 one time so they are said to be "getting odds" to place the bet as the law of averages will work in their favor over time.

2 Bots

One controversy that has long existed in the game of on-line poker is the use of bots. Bots are pieces of software that are typically used by on-line gamers to automate the process of providing some form of input for he user. In on-line poker, bots can range from simple hotkey scripts that employ a simple and standard strategy to much more sophisticated programs that employ an intelligent AI. A simple strategy based solely on hand probabilities and pot sizes may only result in a small amount of money earned per hour. However, multiple bots can be run at different tables around the clock. Unlike its human counterpart, a bot is also not subject to its play degrading due to fatigue nor emotions. Thus the average amount that it is capable of winning per hour will be fairly consistent.

2.1 Artificial Intelligence

Skilled players will rely heavily on their memory to collect information on how each player at the table plays. Keeping track of each players' habits to learn for example how often each player bluffs in a given situation can be used as input to factor into calculating pot odds. However, a bot can far exceed the limits of a human player in both the amount of data that can be readily retrieved and processed within a set amount of time. Therefore, a more sophisticated bot with a complex AI can be far more profitable than a bot playing a simple strategy. This is evidenced by the 2008 Man vs. Machine Poker Championship in which a bot named the Polaris Bot defeated its human competitors and won the entire event [6]. The fact that developing the Polaris Bot required a team of researchers at the University of Alberta over a 17-year period might suggest that players will

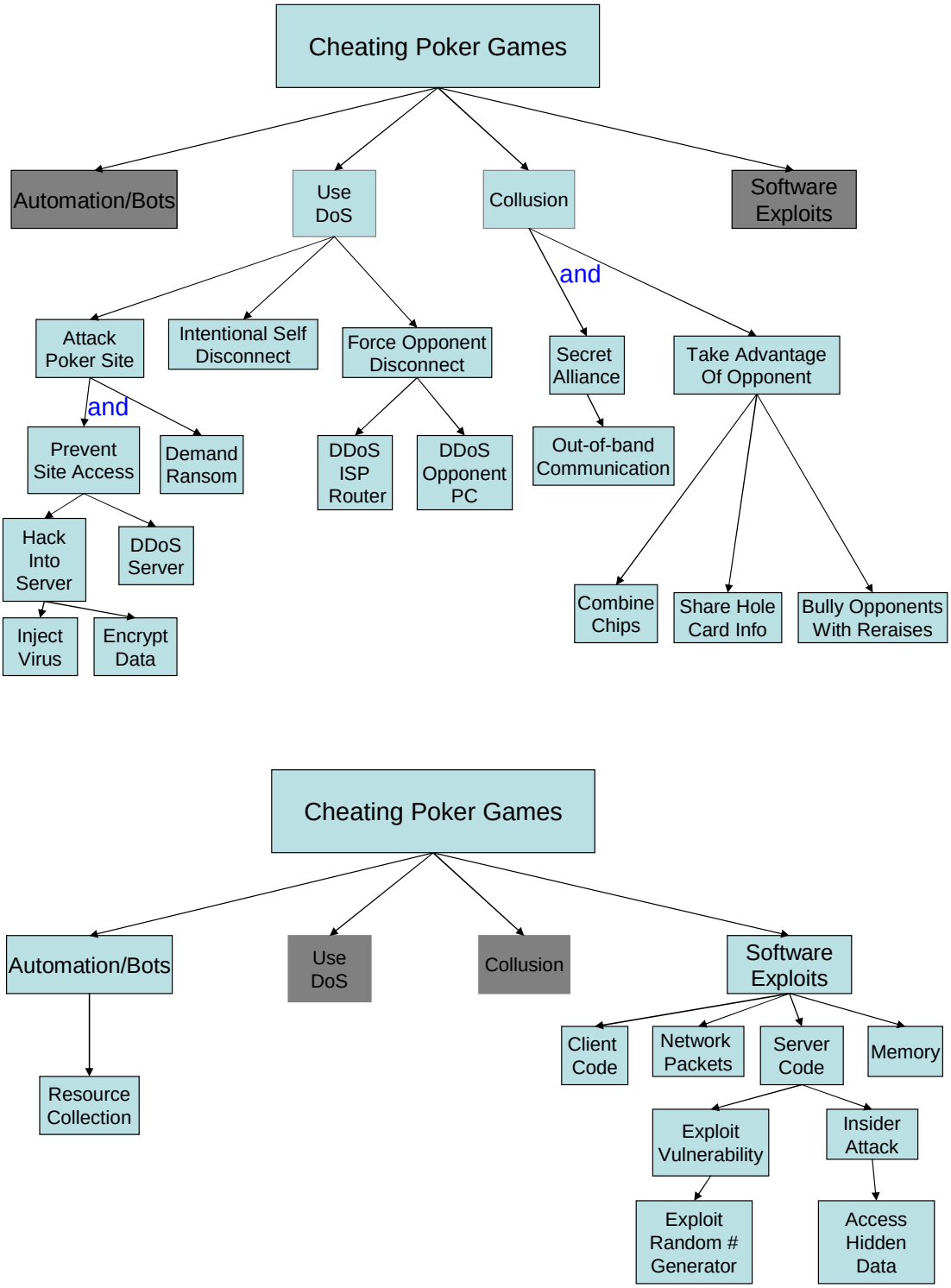


Figure 1: Attack tree for cheating on-line gambling

Figure 2: Example of a CAPTCHA

likely not need to fear the presence of any bot in the typical on-line poker room with this level of sophistication. However, Polaris was playing against poker professionals so the potential for less sophisticated poker bots with enough complexity to play a winning game against average players cannot be ruled out.

2.2 Bot Countermeasures

Some players welcome the use of bots because of their view that they are essentially playing against their human designers. However, many poker sites still consider the use of bots to constitute cheating. To combat the use of bots, some sites will periodically prompt their players with CAPTCHAs. This mechanism prevents bots from interfacing with their clients. Figure 2 shows an example of a CAPTCHA.

Still others have even tried to use detection schemes to identify common bots on their players' systems. A similar method was used by Blizzard Entertainment when they introduced the "Warden" [10]. The Warden software monitors and scans the computers of World of Warcraft players to search for bots or other software enabling the player to cheat. Given the controversy that the Warden caused, employing this countermeasure in poker may be even more controversial than the original problem that it is intended to defend against.

Players themselves will often also try to detect bots by using the in-band chat feature of the poker software since a bot is essentially unable to respond. However, in a game where deception is fundamental to the strategy, some players will refuse to respond as they may be perfectly content to have opponents believe that they are bots.

3 Denial of Service Attacks

Denial of service (DoS) attacks are another form of cheating in on-line gambling. There are two traditional forms of DoS attacks that have been used to exploit online gambling sites: a malicious player can attack the site itself to stop its business or they can attack an opponent to prevent communication between their computer and the server. In addition, there is a non-traditional DoS exploit that malicious players can use in certain situations: an intentional disconnect.

3.1 DoS Against Poker Sites

On-line gambling company Grafix Softech suffered an attack from hackers in Russia [15]. The hackers were able to launch a virus within the production servers used for on-line games. As a result, the data on the servers was encrypted by the virus. The hackers demanded a ransom, the amount of which remains undisclosed to the public, in exchange for the decryption key.

Upon receiving the decryption key, the company was able to decrypt data on all but 1 of its servers. However, on that system the problem became worse once the decryption key was used. The system lost all of its data which was key to their operations. It is unknown if the loss of data was due to some other mechanism put in place by the hackers or human error on the part of Grafix Softech's IT team in trying to decrypt the data. Grafix ultimately turned to CBL Data Recover Technologies Inc. to recover the data. Only the metadata pointing to the locations of real data was removed during the deletion as is typical for most operating systems during a deletion operation. They were able to manually reconstruct the data and get Grafix back up and running.

3.2 DoS Against Opponents

In on-line poker, players are given a set amount of time to make their decision when the "action" gets to them (i.e. when it is their turn to bet). At many sites, if the player does not make their choice within that time then the server automatically folds their hand. Depending upon the information provided by the server about the other players, a given player may have his or her opponents' IP addresses. Alternatively, a user could determine the ISP to which a poker site is connected (by running traceroute for example). In either case, when a cheater is involved in a large hand, he or she can launch a DoS attack either against the opponent when it is his or her time to bet or launch a brief attack against the ISP thereby preventing communication between the opponent and the server.

The challenge here is that the attack has to last just long enough for the opponent to be auto-folded but short enough that communication can resume with the cheater's system so that they can win the pot. Since there are likely a number of variables involved that are outside the control of the cheater, they may not be able to always successfully execute the attack. Therefore, any given attempt could actually cost the cheater money. However, through trial-and-error the cheater can determine the probability of successfully executing the attack. This can be useful in a situation where they are heads up against an opponent for a large pot. "Heads up" means that there are only two players left in a hand. They can simply compare the probability of a successful DoS attack on the opponent against their probability of winning the pot outright. If the chances of a successful attack is higher then executing the attack will still prove profitable in the long run. The cheater may want to limit their use of this cheat so as not to have other players discover their intent which is why they may choose to use this attack only when a large pot is at stake.

3.3 Intentional Disconnects

An alternative policy that some on-line poker sites employ for players who get disconnected is to place them all-in with their current bet and remove the remaining chips in their stack from play. In an all-in scenario, the player will remain in the hand until the end. Whatever money is in the pot at that point is placed to the side as the "main pot". A side pot is created for future bets between the remaining players at the table. The disconnected player is still eligible to win the main pot if their hand beats all of the remaining players at the end.

With this policy, if a player is in a situation where a large pot is at stake and they feel they only have a small chance to win but do not want to pay the remaining bets to get to the end, they can simply disconnect their client by killing the application or causing a network disconnect. The site should take countermeasures by tracking the frequency of disconnects by each player during a hand and the number of times that the disconnects were advantageous to the player i.e. when the disconnect came during a large pot with continued aggressive betting. Suspected players can then be warned or more severe actions can be taken.

4 Collusion

Collusion is any secret collaboration by two or more players at a table. This form of cheating has always been a problem in the game of poker and is difficult to detect even at a live table on a casino floor. Collusion requires some means of hidden communication being available to players. At live tables, players have developed very advanced means of communication with subtle signals and speaking in code that appears to be legitimate conversation about another subject. However, with the action being moved to on-line virtual tables where players sit privately in a location where they can no longer be monitored, solving this problem becomes many times more difficult. The requirement for secret communication is easily met in an on-line environment and once players have established their communication vehicle, there are a number of actions they can take to gain an advantage.

4.1 Combining Chip Stacks

Players may choose to combine chip stacks by having one player purposely lose all of his or her chips to another player with which he or she is collaborating in order to gain an advantage. This technique is also known as "chip dumping". In poker tournaments, the difference between the sizes of two opposing players' chip stacks can in some cases have an even bigger impact on the outcome of the game than the difference between their playing abilities. Two or more players using this technique will then split up the winnings after the game.

4.2 Sharing Hold Card Information

Players can share what the values of their hole cards are with one another to gain a significant advantage in decision-making as the hand plays out. Experienced players are often calculating pot odds with each bet that they place. Knowing the values of two additional cards has a significant impact on the result of these calculations.

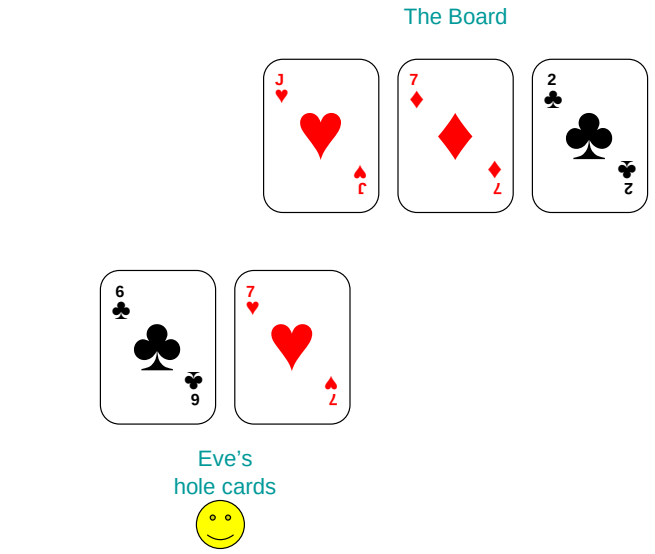
Figure 3 demonstrates this concept. Without knowing Bob's hold cards, there are 5 cards remaining in the deck that can improve Eve's hand. Thus with 2 more cards to be dealt, she has a 20% chance of obtaining what is very likely a winning hand. Calling a bet will only be profitable, long-term, if she is getting at least 4:1 pot odds. However, having Bob's hole card information allows Eve to improve the accuracy of the pot odds. Since Bob has 2 of the cards that would improve her hand, there are only 3 cards remaining in the deck that can improve Eve's hand giving her a 12.5% chance of winning. Now she needs at least 7:1 pot odds. If the pot is giving her 6:1 for example, that appears to be a very good bet until she sees Bob's hole cards. Now she is able to make a better decision and save money.

4.3 Whipsawing

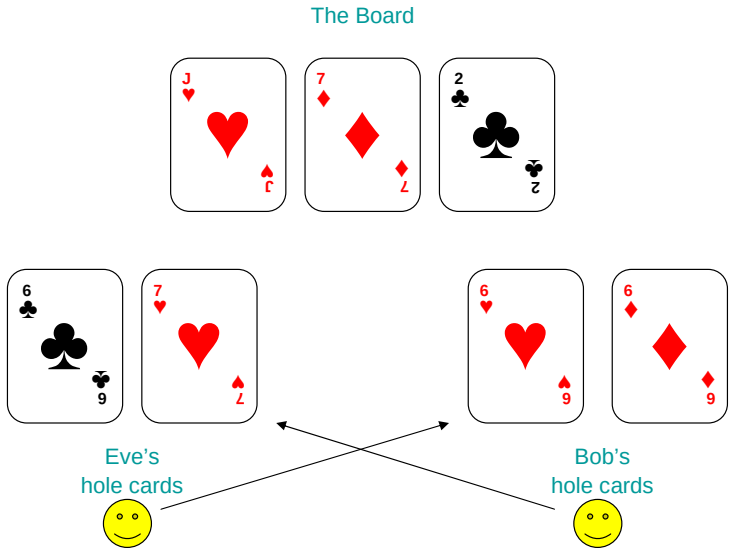
Whipsawing refers to a scenario where two or more players with a small number of players in between them raise and re-raise one another in a coordinated effort to force the players in the middle to fold. This can be extremely frustrating for players who fall victim to this cheat. However, it is also fairly easy to spot when it occurs with any kind of repetition.

4.4 Countermeasures

Most sites have two common mechanisms for preventing collusion: IP checking and collusion-detection algorithms.



- 5 cards left that could improve Eve's hand
  - three 6's, two 7's
- Eve needs at least 4:1 pot odds



- 3 cards left that could improve Eve's hand
  - one 6, two 7's
- Eve now needs over **7:1** pot odds
- Bob also gains information
- This information saves both Eve and Bob money

Figure 3: Eve and Bob are sharing hole card information which gives them an advantage in calculating pot odds

4.4.1 IP Checking

A site can easily detect two players with similar IP addresses sitting at the same table together. Similar IP addresses usually implies a similar location. However, with the numerous means for connecting to the internet nowadays this may not be as effective as it once was. For example, one player could connect via wifi while another player in the same physical location could connect via another close-by wifi network or tethering with their cell phone. With the number of methods for connecting on-line increasing (for example, new vehicular networking technology is likely to be deployed during the year of this writing), IP checking will lose further effectiveness as time goes on. Even when this method is effective in keeping physical distance between players at a table, nothing can prevent the players from communicating via phone, instant message, walkie talkie, etc.

4.4.2 Collusion-Detection Algorithms

Most sites have algorithms for detecting collusion. For example, in the case of whipsawing one can determine algorithmically whether or not this form of cheating is present with a high degree of probability. If two players are regularly raising and re-raising, thereby causing players in the middle to fold then these two players are usually collaborating. However in the case of sharing hole card information, detecting players committing this form of cheating can be extremely difficult. Another example of cheating will later be presented regarding a case where a player had the ability to see the hole cards of all of his opponents yet the effect this had on his decision-making was never detected by the site. Therefore, this form of collusion where a player has information about only two additional cards is even far less likely to be detected even though it creates a significant advantage.

The most effective defense against collusion is for on-line gambling sites to track player statistics and conduct regular investigations of players whose statistics lie too far outside the standard deviation [9]. This can help to prevent huge advantages that a cheater might gain over an extended period of time.

5 Software Exploits

In any type of software, hacking can come in many forms and take place in various components of its overall functionality. On-line gaming is no different. The cheater can look for software exploits that can be performed on his or her own computer by investigating and/or modifying the client code of the game, the network packets being communicated back and forth, or the memory of the system. In order to reduce the opportunity for vulnerabilities, modern day game design dictates that the client should not perform any critical decisions nor calculations that will have an impact on the outcome of the game. Otherwise any of these client-side attacks can be used to artificially produce an outcome that favours the cheater. The problem with many on-line games is that there are CPU-intensive computations for which performance can be greatly improved when they are offloaded to the client. This is usually the case in a graphics-intensive or high-action environment with a large number of operations so it is not uncommon to find a vulnerability in a game's client.

However, running an on-line gambling game generally requires little processing power so these types of games are generally not victims of this tradeoff. As a result, it is not likely that the client will be designed in such a way that it will control any of the critical decisions. There are a number of what-if scenarios that could be dreamt up for hacking the client if it had control over some critical decision but that just is not likely to happen for on-line gambling with the absence of the

security/performance trade-off. Therefore, this section will focus only on attacks that are actually likely to happen or have already happened.

5.1 Exploit Vulnerability

Many vulnerabilities exist with any software. One potential area of vulnerability that exists in all computers is the inability to generate truly random data. While there are many flavors of casino games, they all have one thing in common. At their foundation is randomness and players bet on this randomness. With casino games moving into a virtual environment, the random number generator provides a natural point of attack.

5.1.1 Computer Randomness - Shuffling

How computers generate random numbers is another method that hackers can utilize to exploit on-line poker. ASF Software started an on-line poker site and displayed their shuffling algorithm on-line to show how fair this algorithm was. This is following the design principle that security should not be through obscurity. However, Cigital Software (an on-line security company) was quickly able to break the shuffler in real time [8]. This allowed them to predict what cards players held in their hands and what cards would come down for the shared cards.

Basically they knew how the deck had been shuffled. A seed number is generally used to start a "random" sequence in a computer. If the seed and algorithm are known, then the next number in the "random" sequence can be predicted. ASF started with a 32 bit seed, which right away is not good. This is because a real deck has 52! possible ways to be shuffled. A 32 bit seed can only shuffle a deck in four billion different combinations. Though four billion is a large number, this is still significantly less than 52!. Next, once a day, AFS initialized the seed with the number of milliseconds since midnight. There are only 86 million milliseconds in one day, so this implies even fewer possible shuffles.

Cigital Software was able to estimate when the seed was generated and what the server clock was at that time. With this information Cigital Software reduced the number of possible shuffles to less than 200,000 (which a computer can quickly figure out). Once five cards were known by Cigital Software (hole cards and shared cards), they immediately knew what cards their opponents had and what cards would be dealt in the future. This is of course an overwhelming advantage in the game of poker. Figure 4 shows the Cigital GUI used in the shuffling attack.

5.2 Insider Attack

Another major area of concern for many on-line gambling sites is insider attacks. With all of the protections and security measures available to companies trying to guard against attacks, external cheaters are limited in the frequency and impact of their attacks. A cheater on a company's payroll, however, essentially has the keys to the city, particularly if they are an administrator or someone else in a role with increased privileges.

5.2.1 AbsolutePoker Attack

One well publicized case of an insider attack took place at AbsolutePoker [14]. A handful of accounts playing at their high stakes tables were winning at an alarming rate. Officials for the company claimed that they did not discover any cheating when the issue was raised by other players.

This prompted an investigation by a group of players who lost large amounts of money to the suspect accounts - one player estimated his losses at $400,000 to $700,000. The group was able to

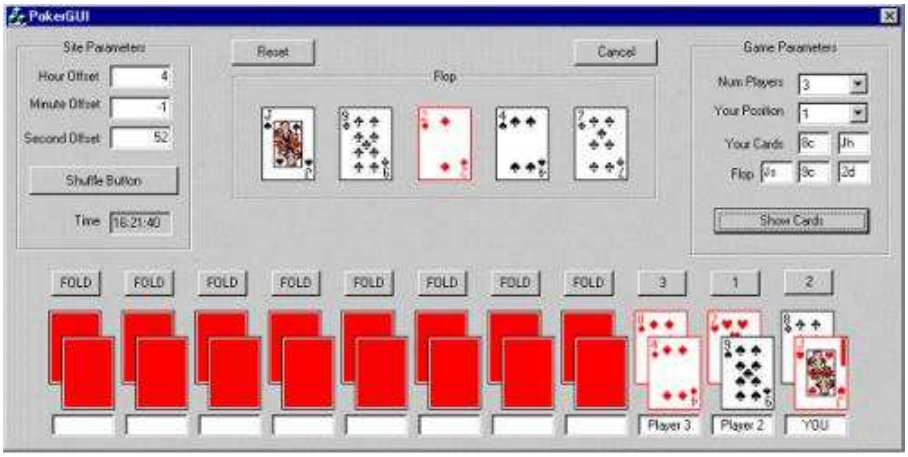


Figure 4: Cigital GUI

obtain the hand histories of their play from AbsolutePoker, including the hole cards of each player. Analysis of the play by these accounts showed that if their play was simply due to "luck" it would be statistically equivalent to winning the jackpot of a lottery with 1 in a million odds 6 times in a row.

The hand history data sent by AbsolutePoker also contained IP address information which they were able to use to discover that the cheating was taking place on a computer within AbsolutePoker's own network - a possible insider attack! AbsolutePoker was able to identify the employee but agreed to allow him to remain anonymous in exchange for details on how the attack was executed. He exploited a vulnerability in their system to gain the ability to see the hole cards of each of the other players at the table. AbsolutePoker never revealed what that vulnerability was, only that they have since patched it. The money was paid back with interest to the victims as AbsolutePoker officials indicated that it had never been withdrawn by the employee.

6 Conclusion

Computer security has been likened to other forms of security in that there is no silver bullet to completely securing a system. Effectively securing one's home or vehicle to prevent theft typically requires that multiple security measures be put in place, including placing locks on the windows and doors and perhaps even installing an alarm system. For houses, a surveillance system may also be appropriate in certain situations, while steering wheel and tire locks may be used for vehicles. As criminals thwart these different forms of security, new security devices are designed and employed to protect people's property. Likewise, securing software, online gambling software included, requires multiple security methods and continued effort to maintain security. Attackers will continue to discover new ways to cheat online games so security needs to be thought of as an ongoing battle to continue to make it more difficult for attackers to exploit a game [9].

Given that security turns into a game of cat-and-mouse, attackers will occasionally have success. Extremely important for gaming companies is to employ a good disaster recovery scheme. If an attack is successful, the environment may need to be reverted to a previous state or have certain attributes reverted. In addition, any attack that results in loss of availability can cost a company large amounts of money by the hour so it is important that they are able to quickly failover to a

secondary site to maintain business continuity.

6.1 Gaming in General

This report has focused on exploiting on-line gambling games such as Texas Hold'em Poker, however the techniques described can be applied to any genre of game (and often with software in general). For example, in World of Warcraft bots are often used to farm resources. Farming is the process of performing a repetitive task automatically to accumulate game resources (for example World of Warcraft gold). Collusion is used to gain advantages in all types of games (Chess, FPS, MMPORGs, etc.). Denial of service attacks against opponents also have obvious advantages in almost any game. DLL Injections are used in all types of software to monitor and modify target processes memory and are important tools in creating sophisticated attacks like wall hacks (seeing through walls in a FPS) and aim bots (which can also be created with bots).

Something that became clear during the exploration of on-line cheating in gambling is that money certainly adds a motivating factor in the protection of any game regardless of whether the transfer of money is in-band as it is for on-line gambling or out-of-band as it is for many other MMOs (massively multi-player on-line games). Regardless of whether a game company itself supports the outside market for virtual items, it adds importance from the players' perspective. This adds pressure on game designers to protect the integrity of their product in order to keep customer satisfaction high. Another excellent point about added motivation to prevent cheating that the exchange of money can generate is found in Steven Davis' book [6]. In it he reminds the user "And, of course, there is one kind of help you usually don't want: the government. Game violence, addiction, privacy, obscenity, pedophiles, gambling, marketing, terrorists, hackers, criminals–all sorts of issues can get you on the government's radar.". Most companies would like to avoid this at all costs.

6.2 Final Thoughts

The study of exploiting on-line games is both technically interesting and of great importance. The importance lies in both the financial power of the video game market and the relation to computer security in general. Game exploiters will always exist and continue to innovate and find new ways to cheat their favorite games. Game companies have to decide how much effort they wish to put into anti-cheat capabilities.

7 Overview of References

Noa Bar-Yosef provides an overview of the various methods for cheating online gambling sites [3]. Greg Hoglund and Gary McGraw have a very popular book detailing various aspects of exploiting online games [8]. They also wrote a precursor to this book which is less exhuastive but gives a brief yet sufficient overview of online game cheats [10]. Stephen Davis provides another book which goes into depth on cheating online games and countermeasures [6]. Adam Lake's book provides information on general principles for protecting games [9]. CBS News provides their investigation of the most popular attack in online poker [14].

References

[1] Cheating in online games. Wikipedia, February 2012. `http://en.wikipedia.org/wiki/Cheating_in_online_games`.

[2] Online gambling - american gaming association, 2012. http://www.americangaming.org/government-affairs/key-issues/online-gamb%ling.

[3] Noa Bar-Yosef. Hacking the house: How cybercriminals attack online casinos. Security Week, August 2011. http://www.securityweek.com/hacking-house-how-cybercriminals-attack-onl%ine-casinos.

[4] Simon Carlass. *Gaming Hacks*. O'Reilly Media, Inc., 2004.

[5] Darawk. Dll injection. Blizz Hackers, March 2006. http://www.blizzhackers.cc/viewtopic.php?p=2483118.

[6] Stephen Davis. *Protecting Games: A Security Handbook for Game Developers and Publishers*. Course Technology PTR, 2009.

[7] Jack M. Germain. Global extortion: Online gambling and organized hacking. TechNewsWorld, March 2004. http://www.technewsworld.com/story/33171.html.

[8] Greg Hoglund and Gary McGraw. *Exploiting Online Games: Cheating Massively Distributed Systems*. Addison-Wesley Professional, 2007.

[9] Adam Lake. *Game Programming Gems 8*. Course Technology PTR, 2010.

[10] Gary McGraw and Greg Hoglund. *Cheating Online Games*. Addison-Wesley Professional, 2006.

[11] Matthew Pritchard. How to hurt the hackers: The scoop on internet cheating and how you can combat it. Gamasutra, July 2000. http://www.gamasutra.com/view/feature/3149/how_to_hurt_the_hackers_the_%scoop_.php.

[12] Shahen Ramezany. Hacking / exploiting / cheating in online games. Abysssec, March 2011. http://www.abysssec.com/blog/wp-content/uploads/2011/03/Exploiting-Onli%ne-Games.pdf.

[13] Andrew Rollins and Ernest Adams. *Andrew Rollings and Ernest Adams on Game Design*. New Riders, 2003.

[14] Ira Rosen. How online gamblers unmaksed cheaters. CBS News, June 2009. http://www.cbsnews.com/2100-18560_162-4633254.html?tag=contentMain.

[15] Nikola Strahija. Russian hackers raid largest online gaming operation and destroy data in blackma. Xatrix Security, February 2003. http://www.xatrix.org/article/russian-hackers-raid-largest-online-gamin%g-operation-and-destroy-data-in-blackma/2726/.

[16] Daniel Terdiman. Hacking online games a widespread problem. CNET, April 2009. http://news.cnet.com/8301-10797_3-10226485-235.html.

Network Security Visualization

Keith Fligg and Genevieve Max

1 Introduction

Network security is the art and science of detecting, stopping, and defending against current and future network intrusion incidents. As attack tools get more sophisticated and more freely available, it becomes easy for even a novice to launch attacks. Attacks can have many different motives, including bringing network services down, stealing confidential information, and misusing computing resources.

All of the activity generated by an attacker leaves traces, which can be seen by inspecting network traffic. Log files are created in response to both malicious and innocuous activity, depending on the logging level. These logs and traces are what network security professionals have to work with in order to detect, defend against, and prevent security breaches.

The problem is the vast amount of network traffic data to be sifted through. Security tools have been developed to help security analysts process all this data. The aim of any security tool is to find patterns, determine if these patterns are anomalies and communicate the severity of the attack.

One of the most useful tools for network security workers are visualizations. Just as graphs and charts are useful to convey ideas about the economy, experimental results and your spending habits, they can also help make sense of the mass of data related to network operations. Humans are better at processing visual information since their sense of sight is more developed than any other sense. It is the same logic as the common saying "a picture is worth a thousand words." The aim of network security visualization is not to create new information, but to present the available information in an easy to digest manner.

Figure 1 shows a typical attack and how visualizations help the security analyst fix vulnerabilities much faster than reading through raw network data. However, creating just any image of data will not achieve the goals of network security visualization. There is a need to understand visualization theory just as much as network security in order to design and build an effective network security visualization tool.

2 The Psychology of Visualization

Network security visualization allows security analysts to take the multitude of text logs produced by a network and generate an image indicating what information these logs contain. Since such a large portion of our brain is dedicated to gathering information by looking at it, it makes sense that we would want to use this natural strength to our advantage when designing network security tools. A properly structured visualization allows us to take large amounts of information and process its meaning quickly.

However, simply transforming text logs into a picture will not achieve the goal of easily understanding network activity. The visualizations constructed must allow important information to be more visible than less important information. This concept is called *pre-attentive*, which is the

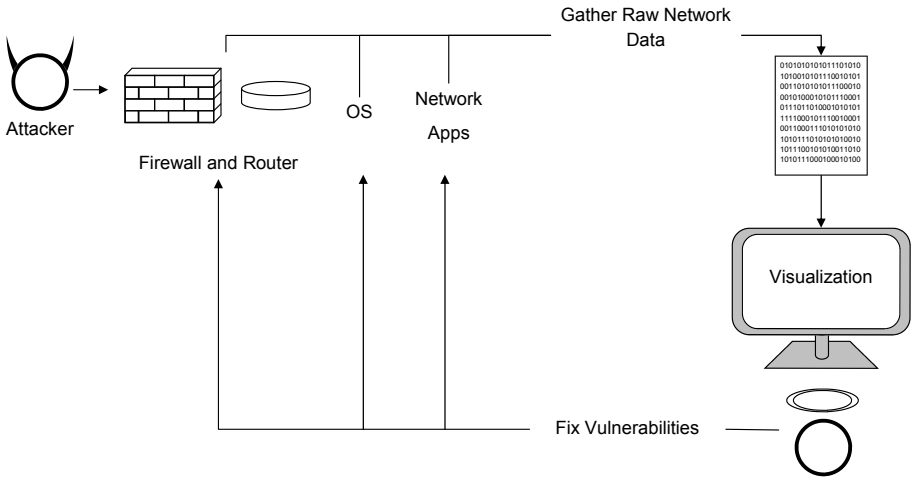


Figure 1: A typical attack on a network must be detected as quickly as possible. This speed is achieved using network security visualization.

characteristic seen when an image stands out from other images. Commonly, the circles, squares, lines and other shapes that make up an image are called *objects*. Objects that are pre-attentive are effortlessly distinguishable from other objects. Additionally, pre-attentive objects are easy to spot no matter how many irrelevant objects, *distractors*, there are in the same image. This concept is particularly important to network security visualization as we want an event, such as a security breach, to be immediately visible no matter how much normal activity is occurring in the background.

The amount of time it takes for the human brain to process pre-attentive objects versus non-pre-attentive objects is a factor in why pre-attentive objects have their stand out effect. A pre-attentive object will be processed in 10msec or less. A non-pre-attentive object will be processed in 40msec or more. This means pre-attentive objects are perceived by the brain before the human is even aware of what they are looking at.

2.1 Pre-Attentive Objects

Pre-attentive objects can be categorized by their:

- Color
- Position
- Form
- Motion

Color is pre-attentive as seen in Figure 2(a). The dark circle is quickly seen, even though there are several lighter circles in the image. Position refers to the physical location of objects in an image. Figure 2(b) illustrates how position is pre-attentive, in that the circle that is offset from the others can be quickly spotted. Form describes a physical difference between an object of interest and all other objects. Some examples are shape, size, orientation and enclosure. Figure 2(c-f) shows how these can be used. Motion refers to an object moving or blinking on the screen, as opposed to being stationary.

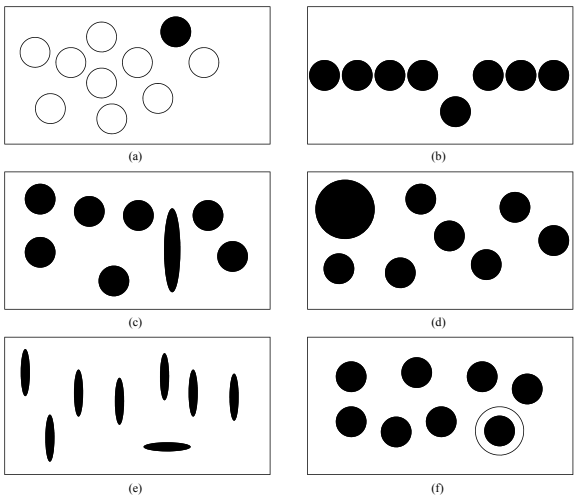


Figure 2: Pre-attentive objects are distinguishable from the objects surrounding them. (a) Color (b) Position (c) Form – Shape (d) Form – Size (e) Form – Orientation (f) Form – Enclosure

2.2 Visualization Techniques

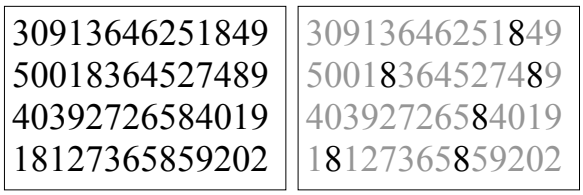
A challenge in network security visualization is that there is no single right way to represent network logs because these logs are abstract data. As an example, other science disciplines can use visualizations to represent molecules or geographic features. Such visualizations could be based on physical measurements and data. In network security, data is abstract and does not correspond to physical data. Despite this challenge, there are techniques that can be used to develop an effective visualization of network security data.

Some of the features sought after in visualizations displaying large amounts of data, described in subsequent paragraphs, are:

- Remove Serial Parsing
- Minimize the Number of Types of Objects
- Minimize Non-data Ink/Pixels
- Determine Root Cause
- Provide Interactive Display
- Allow Data Comparisons

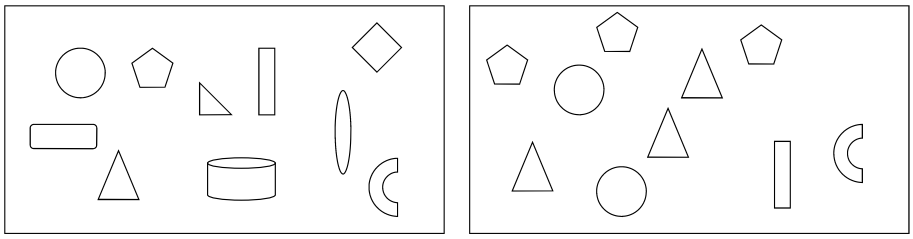
A visualization must allow the user to understand the image without using serial parsing, or visual scanning. This follows the idea of pre-attentive objects. The user of the system should not have to visually scan an image to find what is important, or there would be no improvement over visually scanning log files. Instead, a visualization must make the important information stand out. A popular way to illustrate this is shown in Figure 3. In Figure 3(a) it is difficult to find all the 8s in the table. In Figure 3(b), the pre-attentive category of color is used to easily detect where the 8s are.

When designing a visualization, attention must be paid to keep the number of different types of objects to a minimum. Although color and shape are pre-attentive, overuse of these can have the



(a) (b)

Figure 3: Reduce serial parsing by highlighting information the user needs to see immediately, such as the 8s



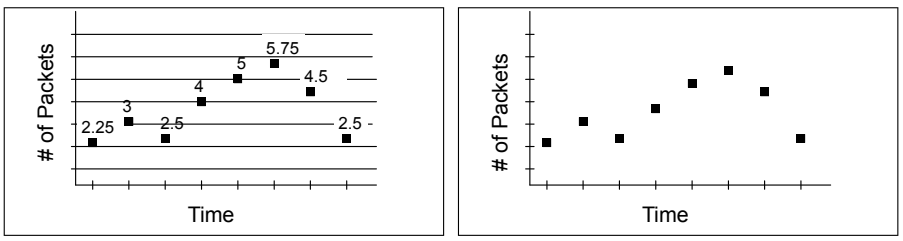
(a) (b)

Figure 4: Minimize the number of objects, such as the number of different shapes

opposite effect. The number of shapes should be kept to a minimum to avoid important information from being overlooked. A general good number to use is five to eight shapes per visualization. In Figure 4(a), the image shows how too many shapes detract from what is important. Figure 4(b) highlights two possible anomalies, the rectangle and the arc.

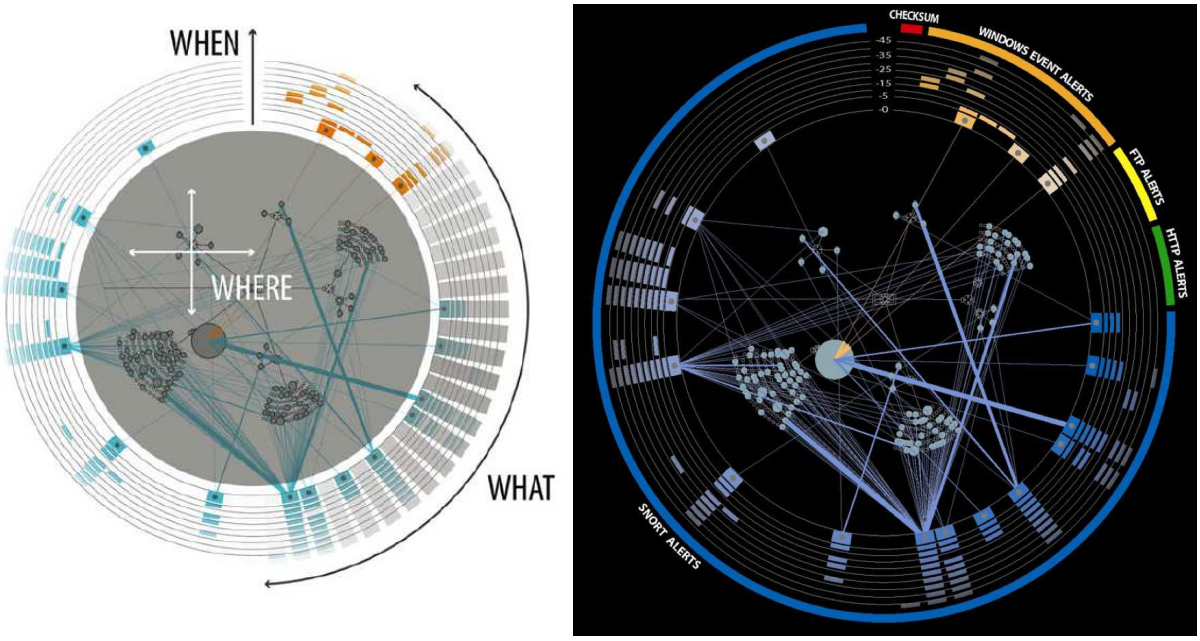
There are only so many pixels on any screen, and whatever visualization is produced must make the best use of what resources are available. For this reason, the amount on non-data ink, or pixels, must be minimized. An example of non-data ink on a chart are the two lines drawing the x and y axes. While these are necessary, we may not need the tick marks at each interval, so they should be removed. As seen in Figure 5(a), the numeric labels on each data point and the horizontal grid lines on the plot are examples of non-data ink that can be removed without affecting the main message the graph is conveying. Figure 5(b) shows how this graph can be made easier to read.

While a visualization may show that a security breach has occurred, it is even more important that the root cause, or why the breach happened, can be determined from the visualization. This



(a) (b)

Figure 5: Minimize the amount of non-data ink, such as excess lines in a graph



(a) **(b)**

Figure 6: Example of a network security visualization tool applying visualization concepts. (a) This visualization shows how an image can anwser the questions *where*, *when* and *what* about an attack. (b) This is the same visualization, but showing the actual tool usage, including what types of alerts are being analyzed. Image(a) from: [6], image (b) from: [7]

may require a visualization to be interactive, or allow the ability to overlay multiple data sources.

A network security visualization must not simply be an image, it must allow the user to further explore and understand the data they are looking at. Creating an interactive display achieves this goal.

It is not enough to know that an anomaly has been detected in a network. It is also important to show the amount of deviation from normal system behavior. To achieve this, a visualization should allow comparisons between different sets of data.

2.3 Application to Network Security Visualization Tools

A network security tool must answer three major questions:

- *Where* in the network is the attack happening?
- *When* is the attack happening?
- *What* type of attack is happening?

Figure 6(a) illustrates how visualization techniques discussed in the previous sections can be applied to answer these three questions. The center portion answers the question *where*. The circles represent different nodes in the network, and the network topology is shown through the links between the circles. The size of the circle increases when a node experiences more alerts. The pre-attentive idea of form based on size is used to indicate security issues.

Each column in the outer circle corresponds to a different type of attack, and answers the question *what*. Figure 6(b) shows actual usage of the tool with the types of alerts being looked for.

The circles moving outward represent time, with the circles closer to the network topology being more current, and past activity moving outward. This part of the visualization answers the question *when*.

3 Data Sources

In order to generate visualizations, data is needed. One source of data are log files generated by firewalls, web servers, routers and other tools that utilize the network. Other sources of log files may be virus/malware detection systems, email systems, and the operating system itself. In addition to log files, network packets themselves may be inspected and used to create visualizations. Finally, Intrusion Detection Systems can aggregate log files and network packets and perform actions according to user-defined rules. All of the above are fodder for visualization generation.

3.1 System Log Files

Almost every daemon or service that runs on a computer is capable of generating information that pertains to its operation. Some simply write log files directly to their own logging subdirectory, like Tomcat, while others utilize system-wide logging services, such as syslog, which stores logs in a dedicated operating system directory, e.g., /var/log. Additionally, the log level of most software is configurable and generally range from extremely verbose debugging output to very terse logs of exceptional conditions and critical errors.

In almost all cases, the log entries are time stamped so that there is a chronological order to the log entries. This is important because without a clear ordering, subsequent entires may not make sense, because of the lack of context.

Similarly, most log files on UNIX and UNIX-like systems are text-based. This means that log files can become quite large – especially at greater levels of verbosity. There are automatic mechanisms to rollover and compress log files, but this is only a stopgap measure, since most systems do not have infinite disk space. Therefore, there is usually a relatively small window of time during which log inspection may occur.

3.2 Packet Data

The protocol of the Internet, and thus what nearly all network security is concerned with, is IP, or Internet Protocol. Another is ICMP, or Internet Control Message Protocol, upon which the 'ping' utility is built. IP routes datagrams between machines on the Internet and each machine is assigned an IP address.

The datagrams that are transported with IP include TCP (Transport Control Protocol) and UDP (User Datagram Protocol) among others. These are part of what is considered the transport layer. Applications like DHCP, HTTP, FTP, DNS, etc. run on top of this layer and are part of the application layer.

Of primary concern to network security on the packet level are TCP and UDP. Application layer protocols are generally dealt with as log files, whereas the TCP and UDP packets can be taken apart and analyzed separately. The primary advantage of looking at packets is that they contain the 'ground truth'. Log files may be manipulated by intruders and malicious users, so called insider threats, but the packet data can be retrieved directly from the networking stack that runs in kernel space, and are therefore much harder to manipulate.

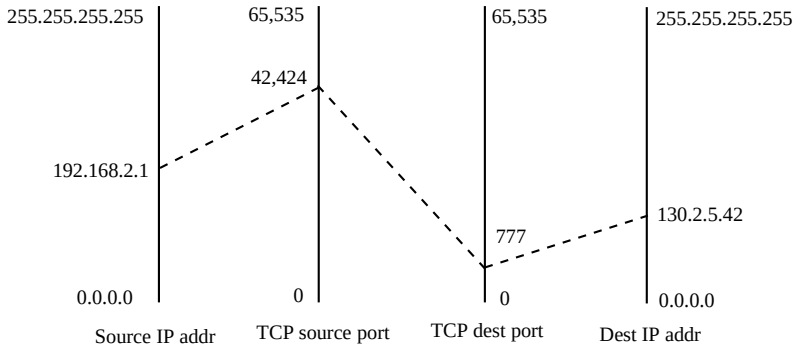


Figure 7: Parallel coordinate plot for a TCP packet from 192.168.1.1:42424 to 130.2.5.42:777.

3.3 Intrusion Detection Systems

Another source of visualization data are Intrusion Detection Systems, or IDSs. These are policy-based security systems that monitor log files and/or network packets looking for conditions that may indicate a security breach. They themselves may write log files, send alerts to system and network administrators, and/or write to the console. Some IDS systems, known as Intrusion Prevention Systems may even modify firewall rules in response to events.

One of the issues with IDSs are the rules that they use. On one hand, rules written too strictly may generate false positives. One the other hand, rules can be too loose and allow attacks to occur undetected, a so called false negative. Balancing the strictness of rules can be difficult in itself, and may be made worse by network users that are doing 'interesting' things on the network that cause rule relaxation to reduce false positives.

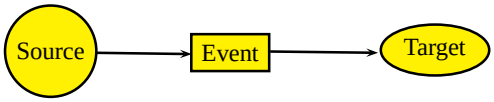
4 Example Visualizations

Visualizations come in many shapes, sizes and colors, and most are based on traditional methods of data visualization. Different types of data may be more suitable for a different type of visualization. Additionally, as we try to cram more dimensions (dimensions of data, not necessarily spatial) into the visualization, new and different techniques are necessary to make the resultant visualization useful and not just a cluttered mess of colored pixels.

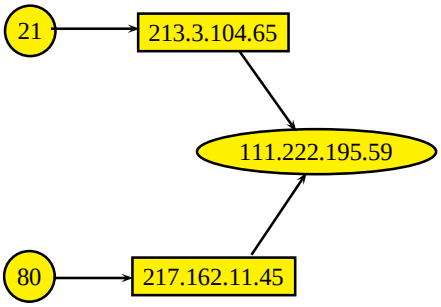
One of the most useful visualizations to investigate data packets is the *parallel coordinate plot* illustrated in Figure 7. This type of plot makes it easy to see how traffic is flowing between machines. The number of axes used is only limited by screen width, but a practical limit is around 20. Figure 9 shows an example with 19 axes, showing one evening of Internet traffic to a single computer placed directly on the Internet, and not responding to any traffic. While the graph is busy, it shows overall trends in packet characteristics, and serves as a good starting point from which deeper evaluation may take place. From here packet length, protocols, payloads, and traffic patterns may be investigated by filtering the input, changing axis assignments or zooming in on certain spans, for example ports between 0 and 1024.

Another useful visualization is the *link graph*, depicted in Figure 8. A link graph is a directed graph who's basic structure is shown in Figure 8(a). The circular 'source' vertex points to the square 'event' vertex, which in turn points to the 'target' vertex. The link graph is another way to visualize traffic in a network, and is usually used to display firewall logs. It is discussed in more detail in Section 5.

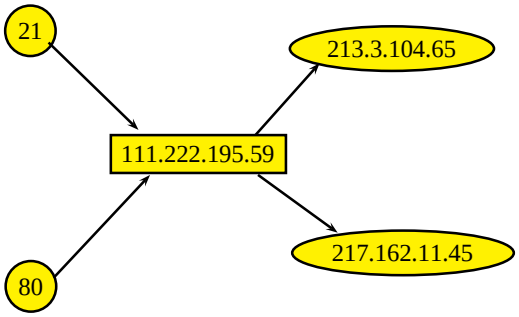
In many cases, different views into the same data are needed. One solution to this problem is to



(a) Link graph nomenclature.



(b) Destination port, source address, and destination address.



(c) Destination port, destination address, and source address.

Figure 8: Example link graphs.

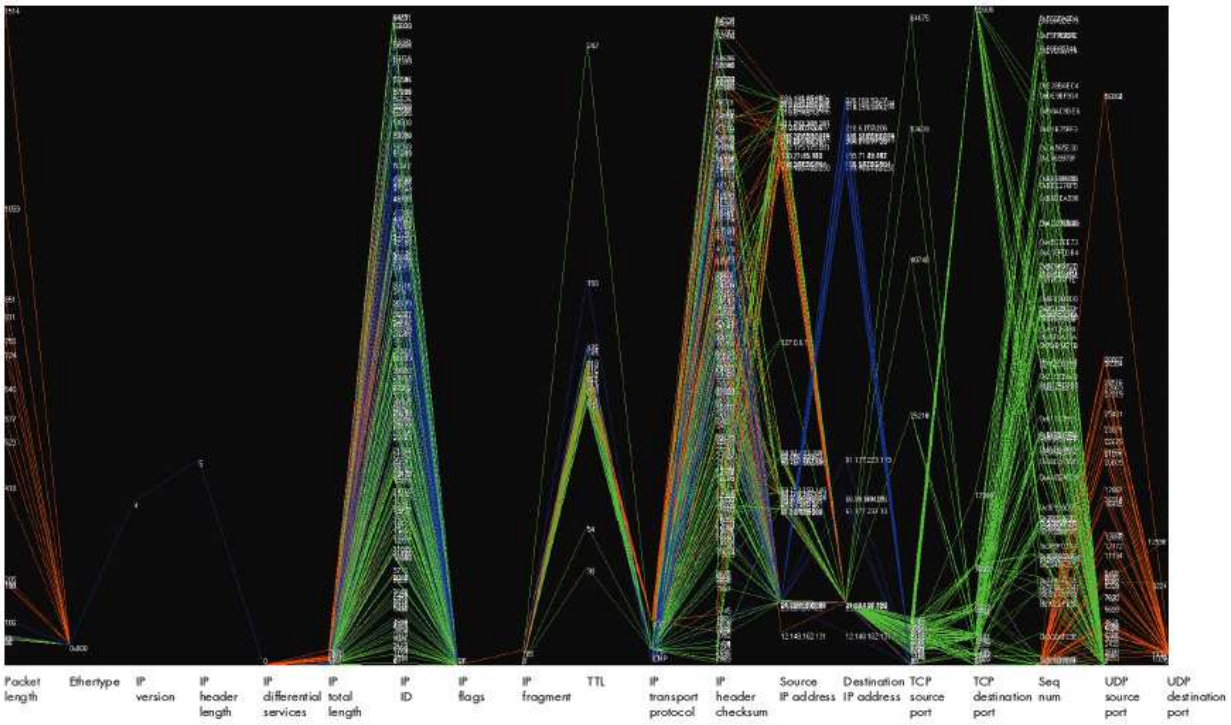


Figure 9: An example parallel coordinate plot from [4].

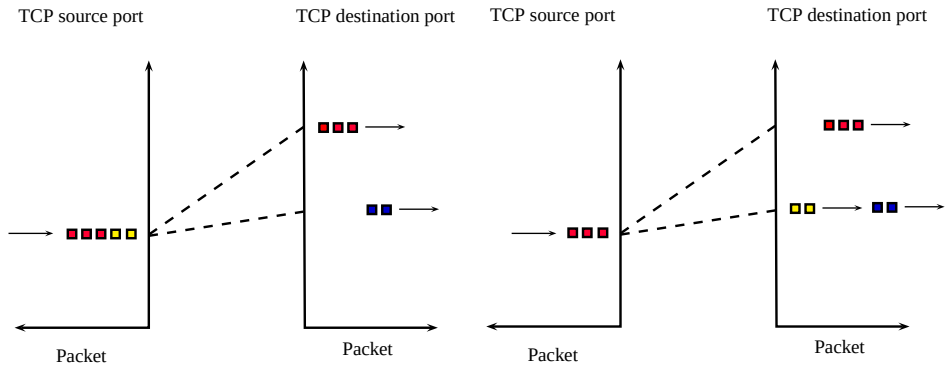


Figure 10: Packets are animated in this parallel coordinate plot. The left side shows five packets arriving, with five having already arrived. The right side is after two more packets have arrived.

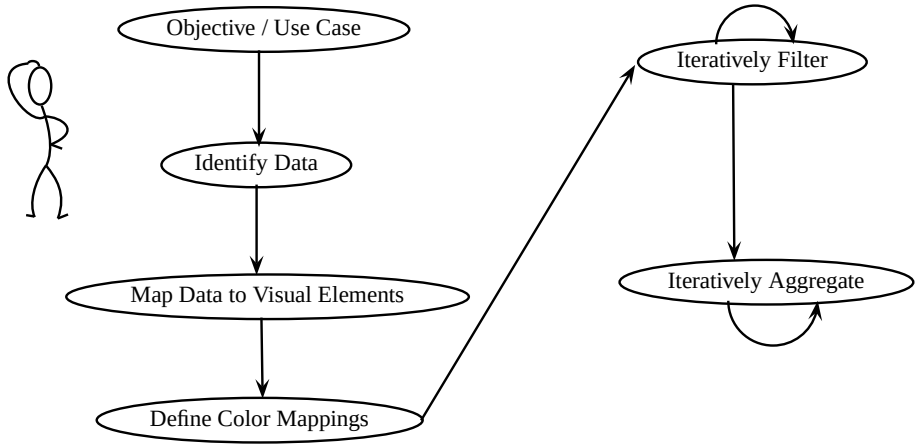


Figure 11: Typical Visualization Creation Workflow

provide several different windows with different views. For example, a bar chart showing the packet count per target address; a parallel coordinate plot showing attacker IP address and destination port in another window; and a pie chart showing the business role of target machines in yet a third window. All of these may be controlled by a sliding window on a timeline showing packet count per unit of time. Using multiple views, especially when they all automatically update to reflect the current dataset, can help to gain insight into the data that may not be available by just looking at one type of graph.

Animation can be useful in network security. For example, adding packet animation to the parallel coordinate plot can be very useful. This alleviates a serious problem with the parallel coordinate plot: if many packets have the same attributes, there will be only one plot for all of the packets. As shown in Figure 10, animating the packets by arrival time significantly clarifies the graph.

5 Workflow

The typical workflow used to create visualizations can be seen in Figure 11. The first step is to clearly state the purpose of the visualization. This important, often ignored step is crucial to ensuring that subsequent steps are properly aligned. Skipping this step may lead to some interesting discoveries, but most of us are very much event driven, and unless you start with a clear use case,

time will be wasted. Finally, it is at this stage that the type of visualization to use will be decided.

The next step is to identify the data fields that will be visualized. This may be source and destination IP addresses or ports if the data are network packets. Netflows may simply be the IP addresses. Timestamp data may be necessary if looking at data over time, or the change of a quantity over time. Log files may include user information, filesystem information, etc. in addition to networking data.

Once the data has been identified, one needs to map the data to the individual aspects of the visualization being created. This step is important, since choosing the incorrect mappings can obfuscate what is happening with the data, as shown in Figure 8(b) and (c). Figures 8(b) uses the structure of destination port → source address → destination address, and clearly shows two connections to 111.222.195.59. These are the connections to port 21 from 213.3.104.65 and another to port 80 from 217.162.11.45.

The same information visualized as destination port → destination address → source address is shown in Figure 8(c). The problem with this graph is that it is impossible to tell which host initiated the connection to port 21 and which to port 80. Perhaps both hosts made connections to each port? It's not possible to tell. This is an excellent example of how choosing the correct visual element can make or break a visualization.

After the data is mapped to the elements of the visualization, color mappings may be chosen. The use of color should enhance the immediate understanding of the data. Sometimes color is not necessary, and can be harmful if it is abused, as discussed in Section 2.2.

Once data is mapped to colors and visual elements it's possible to draw the visualization. However it may not immediately show what's important, but hopefully some patterns appear. The next step is to filter the data to make the interesting patterns clearer. Filtering may include removing events that might be firewall or server misconfigurations, like DNS not being setup correctly. This is an iterative process because as data are removed, the picture will become clearer (or not) and you will need to either filter more data, or put some previously filtered data back.

Finally, another data reduction method that is similar to filtering is aggregating. An example of aggregating data is to lump together all incoming traffic by the first IP octet, e.g., 194.xxx.xxx.xxx. A variation is to only identify internal hosts by the last octet. This clears up the graph, while still leaving the pertinent information in the visualization. The difference between filtering and aggregation is that in the former case, the filtered data is completely removed from the visualization, whereas in the latter case, the data still exists; however it's combined with similar data, thus reducing the pixel count of the visualization.

6 Conclusion

Network security visualization combines both the fields of visualization theory and network security practices to help ease finding attacks on network systems. These tools, when properly designed, allow details about the attack's progress to be determined quickly. By gathering the right data and visualizing it logically, situational awareness is increased, and attacks can be easily communicated to the people who need this information and the attack can be addressed accordingly.

References

[1] Robert Ball, Glenn A. Fink, and Chris North. Home-centric visualization of network traffic for security administration. In *In VizSEC/DMSEC 04: Proceedings of the 2004 ACM workshop on Visualization and*, pages 55–64. ACM Press, 2004.

[2] Ryan Blue, Cody Dunne, Adam Fuchs, Kyle King, and Aaron Schulman. Visualizing real-time network resource usage. In *Proceedings of the 5th international workshop on Visualization for Computer Security*, VizSec '08, pages 119–135, Berlin, Heidelberg, 2008. Springer-Verlag.

[3] Bill Cheswick, Hal Burch, and Steve Branigan. Mapping and visualizing the internet. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '00, pages 1–1, Berkeley, CA, USA, 2000. USENIX Association.

[4] Greg Conti. *Security Data Visualization: Graphical Techniques for Network Analysis*. No Starch Press, 2007.

[5] Anita D. D'Amico and K. Whitley. The real work of computer network defense analysts. In Goodall et al. [8], pages 19–37.

[6] Stefano Foresti, Jim Agutter, Yarden Livnat, Shaun Moon, and Robert Erbacher. Visual correlation of network alerts. In *IEEE Computer Graphics and Applications*, pages 48–59. IEEE, 2006.

[7] J. R. Goodall. Introduction to visualization for computer security. In John R. Goodall, Gregory Conti, and Kwan-Liu Ma, editors, *VizSEC 2007*, Mathematics and Visualization, pages 1–17. Springer Berlin Heidelberg, 2008. 10.1007/978-3-540-78243-8_1.

[8] John R. Goodall, Gregory J. Conti, and Kwan-Liu Ma, editors. *VizSEC 2007, Proceedings of the Workshop on Visualization for Computer Security, Sacramento, California, USA, October 29, 2007*, Mathematics and Visualization. Springer, 2008.

[9] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6:24–43, January 2000.

[10] Noah Iliinsky Julie Steele. *Beautiful Visualization*. O'Reilly Media, Inc., 2010.

[11] Noah Iliinsky Julie Steele. *Designing Data Visualizations*. O'Reilly Media, Inc., 2011.

[12] A. Komlodi, P. Rheingans, Utkarsha Ayachit, J.R. Goodall, and Amit Joshi. A user-centered look at glyph-based security visualization. In *Visualization for Computer Security, 2005. (VizSEC 05). IEEE Workshop on*, pages 21 – 28, oct. 2005.

[13] Kiran Lakkaraju, William Yurcik, and Adam J. Lee. Nvisionip: netflow visualizations of system state for security situational awareness. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, VizSEC/DMSEC '04, pages 65–72, New York, NY, USA, 2004. ACM.

[14] C.P. Lee, J. Trost, N. Gibbs, Raheem Beyah, and J.A. Copeland. Visual firewall: real-time network security monitor. In *Visualization for Computer Security, 2005. (VizSEC 05). IEEE Workshop on*, pages 129 – 136, oct. 2005.

[15] Yarden Livnat, Jim Agutter, Shaun Moon, Robert F. Erbacher, and Stefano Foresti. A visualization paradigm for network intrusion detection. In *In Proceedings of the 2005 IEEE Workshop on Information Assurance And Security*, pages 92–99. IEEE, 2005.

[16] Raffael Marty. *Applied Security Visualization*. Addison-Wesley Professional, 2008.

[17] Jonathan McPherson, Kwan-Liu Ma, Paul Krystosk, Tony Bartoletti, and Marvin Christensen. Portvis: a tool for port-based detection of security events. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, VizSEC/DMSEC '04, pages 73–81, New York, NY, USA, 2004. ACM.

[18] Toby Segaran. *Programming Collective Intelligence*. O'Reilly Media, Inc., 2007.

[19] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers, 2004.

[20] Christopher D. Wickens, Diane L. Sandry, and Michael Vidulich. Compatibility and resource competition between modalities of input, central processing, and output. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 25(2):227–248, 1983.

Password Cracking

Sam Martin and Mark Tokutomi

1 Introduction

Passwords are a system designed to provide authentication. There are many different ways to authenticate users of a system: a user can present a physical object like a key card, prove identity using a personal characteristic like a fingerprint, or use something that only the user knows. In contrast to the other approaches listed, a primary benefit of using authentication through a password is that in the event that your password becomes compromised it can be easily changed. This paper will discuss what password cracking is, techniques for password cracking when an attacker has the ability to attempt to log in to the system using a user name and password pair, techniques for when an attacker has access to however passwords are stored on the system, attacks involve observing password entry in some way and finally how graphical passwords and graphical password cracks work.

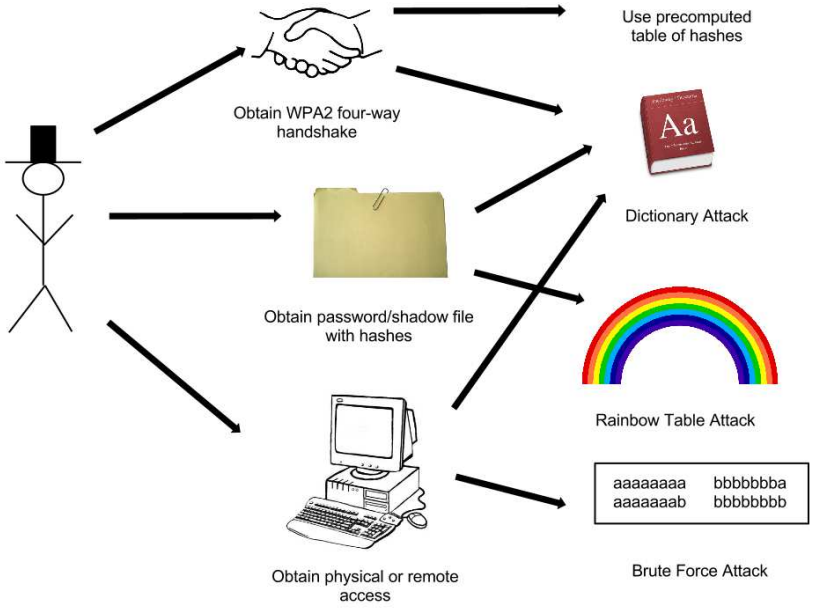


Figure 1: The flow of password attacking possibilities.

Figure 1 shows some scenarios attempts at password cracking can occur. The attacker can gain access to a machine through physical or remote access. The user could attempt to try each possible password or likely password (a form of dictionary attack). If the attack can gain access to hashes of the passwords it is possible to use software like OphCrack which utilizes Rainbow Tables to crack passwords[1]. A spammer may use dictionary attacks to gain access to bank accounts or other

web services as well. Wireless protocols are vulnerable to some password cracking techniques when packet sniffers are able to gain initialization packets.

2 How Passwords are Stored

In order to understand how to compromise passwords, it is first necessary to understand how passwords are stored on typical systems. Storing user names and corresponding passwords in clear text is an unacceptable solution. Attempting to hide passwords stored as clear text (such as putting the password file deep in a convoluted directory hierarchy) would amount to "Security Through Obscurity" which would also be unacceptable. The Unix system of file management provides a better solution though: one of permissions. Initial versions of Multics (the precursor to Unix) stored the password file in clear text, but only viewable with superuser permissions. This solution also failed when a bug switched some temporary files around and the password file (in clear text) was printed for every user upon login.

Unix, instead, stores the hashed value of passwords in the password file instead of the actual passwords. Then when a user inputs their password, the system can simply take the hash of the input and compare it to the stored hash value. To get an idea of how a password file is stored, let's look at the Unix password file scheme.

2.1 Unix Password File

On most Unix-based file systems the password file is located at /etc/passwd[8]. Each line in this file contains information about one account on the system. The file itself is readable by all users, but is only writable with superuser privileges. Each entry in this password file has seven fields, like so:

Figure 2: An excerpt from a Unix password file

The first field, "ender", is the account or user name. The second field contains the letter 'x', explained below. The third field is the user number, an integer identifier for the user. The fourth field is the group identifier, which shows the primary group of this user. The fifth field is a comma separated value list known as the Gecos field, information regarding the user's full name and contact information. In this particular case most of the fields are blank in the Gecos field. The sixth field is the home directory for the user. The final field is the program which should be executed when the user logs into the system; in this case the bash shell is used when ender logs in.

In early versions of Unix the second field for entries into the password file (the letter 'x') contained the one-way hash values for the passwords on each account. Over time those were moved to a shadow file (located in /etc/shadow), only readable by those with superuser privileges. This provides an additional layer of defense.

2.2 Windows Password File

The Windows system for storing the password file is similar to the Unix strategy. The password file for Windows, known as the Security Accounts Manager (SAM) file, is located in

`C:\windows\system32\config\sam.`

An entry in the SAM file contains seven colon deliminated fields: the user name, user number, encrypted password, hashed password, hashed password under a different algorithm, full name of user, and finally home directory. In contrast to the Unix password file, the Windows SAM file is not readable once the operating system has booted. The password file for Unix needs to remain readable for all users so that certain programs can access user information. Because of this, in order to read the SAM file, it is necessary to get access to it before the system has booted. Alternatively, people have installed secondary operating systems to read the SAM file from there.

2.3 Online Password Storage

Many websites and online services require users to log in with a typical password scheme. This necessitates the storage of password information. However, online services typically store passwords for their system in a non-standardized way, and these systems are not always designed by engineers with backgrounds in privacy or security. For instance, Sony had a security breach in which it was discovered that passwords were being stored in cleartext in a SQL database[4]. Password cracking for a system such as this only involves gaining access to the password storage system.

2.4 Password Salts

Storing the hashed or encrypted values for passwords is certainly much more secure than storing their plain text in a password file, but there is a common additional measure of security that can be implemented. This tactic involves adding additional randomness to passwords and is known as a password or cryptographic salt. When a password is created, the system will add some sort of randomness to the password. This randomness (or salt) should be different every time a password is made. Because of this, the exact same password will be stored as a different hash on different machines or different accounts on the same machine.

	Password	Hashed Value
No Salt	this1sAg00dPASSword!!	a5a5baa0c16166260e9ef8a48dbde112
Salted	6789o3uigtbgeat7this1sAg00dPASSword!!	53cffc58904a10b9dcc40345433862dc
Salted	v8734ihv6!nre432this1sAg00dPASSword!!	28b8f782262a890b4d730f8001d23bd5
No Salt	love	b5c0b187fe309af0f4d35982fd961d7e
Salted	12bg55tygsdf4gvi9yrdslove	65c96e15930d34dd9a9ce916b81fb044
Salted	879rughq2ebt5dfxcasedlove	a35436c0e0f2821db2703c1983a641ab

Figure 3: A table showing the md5 hash values for unsalted and salted passwords.

Figure 3 shows the effects of password salts on a strong password and a weak password. Without password salts, even a strong password will hash to the same value. This is a vulnerability in the event that an attacker gains access to the shadow file. An attacker could simply store what a few common passwords' md5 hashed values are and then be able to check if any passwords on a system match that hash. Salting causes this sort of attack to become much more difficult; the randomness added to the passwords causes a large amount of entropy in the hashed value, even for weak passwords.

3 Brute Force Attacks

The feasibility of brute force depends on the domain of input characters for the password and the length of the password[5]. Figure 4 shows the number of possible passwords for a given password length and character set, as well as how long it would take to crack passwords of that type.

	lower case	lower/upper	lower/upper/digits	lower/upper/digits/symbols
1	26	52	62	95
2	676	2704	3844	9025
4	456,976	7,311,616	14,766,336	81,450,625
8	$2.09x10^{11}$	$5.35x10^{13}$	$2.18x10^{14}$	$6.63x10^{15}$
16	$4.36x10^{22}$	$2.86x10^{27}$	$4.77x10^{28}$	$4.40x10^{31}$

(a) Password search spaces

	lower case	lower/upper	lower/upper/digits	lower/upper/digits/symbols
1	26 microseconds	52 microseconds	62 microseconds	95 microseconds
2	676 microseconds	2.704 milliseconds	3.844 milliseconds	9.025 milliseconds
4	≈.5 seconds	≈7 seconds	≈14 seconds	≈81 seconds
8	≈2.42 days	≈1.7 years	≈6.9 years	≈210 years
16	≈1.38 billion years	≈91 trillion years	≈1.5 quadrillion years	≈1.4 quintillion years

(b) Desktop cracking times

	lower case	lower/upper	lower/upper/digits	lower/upper/digits/symbols
1	9 nanoseconds	19 nanoseconds	22 nanoseconds	34 nanoseconds
2	241 nanoseconds	966 nanoseconds	1.373 microseconds	3.223 microseconds
4	≈163 microseconds	≈2.61 milliseconds	≈5.28 milliseconds	≈29.1 milliseconds
8	≈74.6 seconds	≈5.307 hours	≈21.6 hours	≈27.4 days
16	≈.5 million years	≈32 billion years	≈.5 trillion years	≈.5 quadrillion years

(c) ElcomSoft Co. cracking times

Figure 4: Password search spaces and how long it would take to brute force them

A desktop computer could attempt one million passwords per second when trying to brute force a password. Figure 4b shows how long it would take in the worst case for each of the input domains and password lengths. ElcomSoft Co. claims the ability to test 2.8 billion passwords per second using a high end graphics processor on a single machine. Figure 4c shows how long it would take to do the same tasks with a powerful graphics card and cutting age technology.

3.1 Dictionary Attacks

Brute force password cracking won't work for sufficently long passwords. Furthermore, passwords are very rarely actually random. Many passwords will be English words, perhaps with the first letter capitalized and a digit appended or prepended[9]. The search space for common or reasonable passwords is much smaller than $2.18x10^{14}$ for passwords of length 8 or less (even for passwords with upper case, lower case and digits available).

From the password data leaked from Sony, over fifty percent of passwords were less than eight characters long[4]. In addition, only four percent of passwords actually had lower case, upper case

and digits and only one percent of the leaked passwords contained a non-alphanumeric character. Some of the most common occuring passwords were seinfeld, password, 123456, abc123 and purple. Creating a dictionary that contains commonly occuring passwords like these would help an attacker guess correct passwords more often than brute forcing. For instance, a publicly available dictionary matched over 35 percent of the passwords leaked[2].

An attacker who gains access to the password or shadow file may not even need to actually try all the passwords in the dictionary (list) of common passwords. An attacker could precompute the hash value of each of those passwords and then simply match those to the hashed values stored in the password file.

This precomputation can potentially be a very serious threat to password security. However, as mentioned above, many systems implement a salt value that adds randomness to passwords for that machine. Now if an attacker would like to precompute hashes of common passwords, he or she would actually need to precompute those hashes for each password coupled with each possible salt. If the salt is sufficiently large, this can be a significant deterrence for the attacker, even for relatively short passwords.

4 Rainbow Tables

The idea of a time-memory trade-off in computer science is very common. The principle is straightforward: by precomputing some part of the problem (commonly either by solving subproblems or by finding common solutions), the time cost of solving the problem in general is decreased while the space requirements are substantially less than what would be needed to fully precompute the solution. This section details how a similar approach can be applied to the problem of password cracking, beginning with an innovation first described decades ago, and from there describing incremental improvements contributed which have led to the development of Rainbow Tables.

4.1 A Time-Memory Trade-Off

The idea of applying a *time-memory trade-off* to the problem of cryptanalysis was first proposed by Martin Hellman in 1980. Though his approach attacks a cipher text encrypted using the Data Encryption Standard (DES), it requires only minor modifications to instead attack a password hashing scheme. Given a precomputed table smaller than what would be employed in an attack

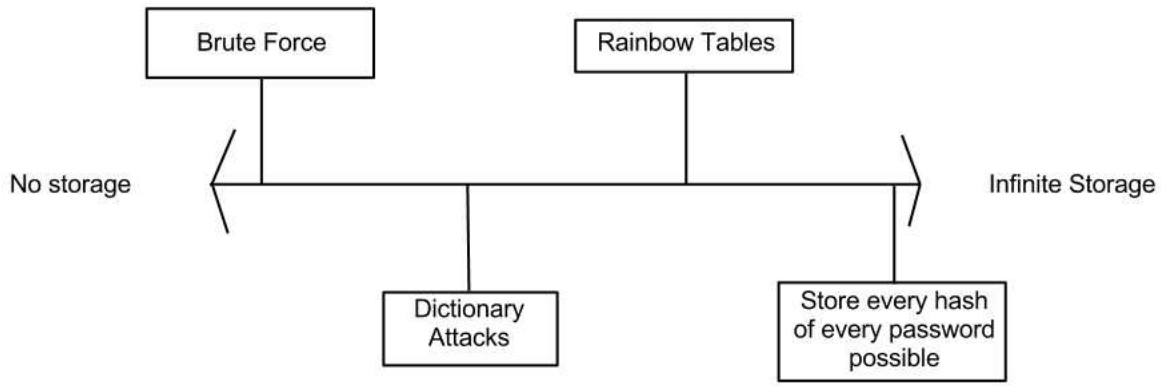
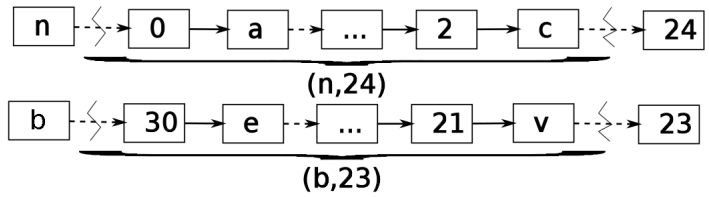
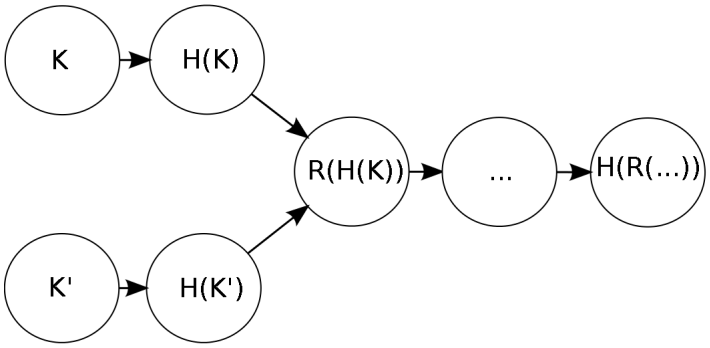


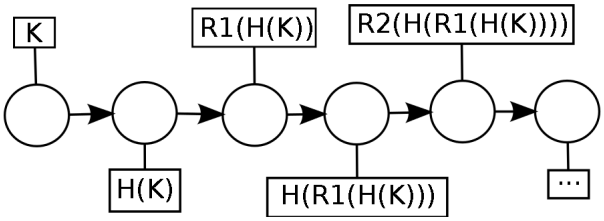
Figure 5: The spectrum of possibilities for password cracking attacks.



(a) Building a Hash Chain



(b) Merged Chains



(c) A Rainbow Chain

Figure 6: Building hash chains according to Hellman's original algorithm, two chains which have merged (storing almost entirely duplicate data), and a Rainbow Chain

where the hash for every candidate password was precomputed, Hellman's algorithm significantly reduces the search cost compared to a purely brute force attack [3]. The main innovation introduced by Hellman is the idea of using a *reduction function* which maps hashed output back into key space to produce chains of hashes and keys (the relationship between these functions is illustrated in figure 7). The full table is constructed by building a series of rows (the construction of two rows is illustrated in figure 6a). Each row begins with some random value (in this case n), which is hashed (producing the value 0 here). The hash of the starting value is then reduced (in this case producing the value a), and these operations are repeatedly chained together in this fashion until the row is of the length desired, terminating with some final hash value (in this case 24).

Note that the attack employed after computing the table (detailed below) requires only the start and end point for each chain, which means during precomputation all intermediate points can be discarded as they are used.

In order to attack a hash value, the hash and reduction functions are once again chained together; now, however, each hash value must be checked to see whether it is in the list of the table's row end points (see figure 8). In this case the hash being attacked is 13, and after several iterations of reduction and hashing, the hash value 26 is calculated, which is one of the table's end points. This end point's corresponding start point (which in the figure is s) is then looked up, and

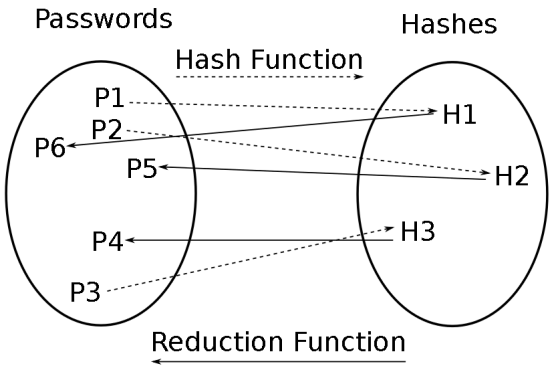


Figure 7: An illustration of the the hash and reduction functions used to generate tables for Hellman's approach

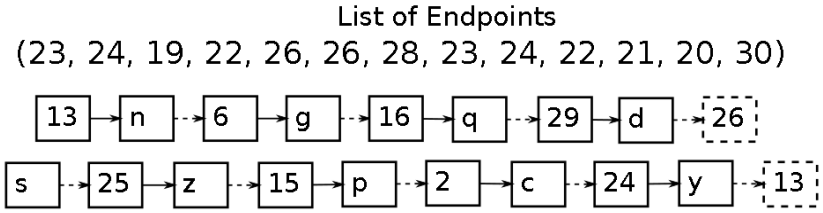


Figure 8: An illustration of the the hash and reduction functions used to generate tables for Hellman's approach

that row of the table is reconstructed (exactly as before) until either the end point's value (26 in this example) or the target hash (13 in this example) is produced. In this case, several iterations of hashing and reduction produce the value 13, and by examining the value immediately preceding it, the attacker can determine that the user's password (or at least a collision for it) is y. Had the end point's value been reached instead of the target hash value, the attacker would have continued building the original chain (which began with 13) until another end point was generated [3].

Hellman's paper uses tables of size $N^{2/3}$ (where N is the size of the search space for passwords) as a value which produces a table which will minimize the number of keys which miss the table completely, while still providing an appreciable increase in computation speed. Although this is a significant improvement, it is still insufficient when attacking all but the weakest passwords: the 16-character lowercase-only password from Table 4c could be cracked in around 123 hours (using the same value of 2.8 billion passwords per second from that table), but a password of the same length using both upper- and lowercase letters as well as numbers and symbols would still take over three hundred thousand years, and even at a severely reduced size, the lookup table would require nearly thirty petabytes of storage. Clearly, more improvement is needed to this approach if we plan to use it successfully.

4.2 Further Improvement

The innovation *Rainbow Tables* offer as compared to previous approaches is the use of a series of reduction functions (as opposed to a series of tables, each generated with a single reduction function). When the chains are generated (as in figure 6a), the position in the chain determines which reduction function is applied to each hash, whereas before the same reduction function was applied at every position [6].

This provides a significant advantage in terms of precomputation time: if a collision occurs, it degenerates into merged chains *only* if the collision occurs at the same position within the chain (since otherwise the next reduction function applied would be different and the chains would likely split again). While a Rainbow Table chain could conceivably contain a series of repeating values, the probability of this event occurring is incredibly low, as is the probability that two chains will contain a series of identical values of any appreciable length [6].

5 LanManager Hashes

An important point to make about Rainbow Tables is that, despite their greatly improved speed compared to previous such attacks, they are still rendered largely ineffective by the inclusion of a salt in the password hashing scheme. Why, then, are there many widely-available tools which use Rainbow Tables to crack the hashes stored in the Windows Security Accounts Manager (SAM) file? This widespread vulnerability is attributable in large part to the Lan Manager (LM) hash system used in older versions of Windows.

5.1 Usage

The LM Hash was the only password hashing protocol used on Windows systems until the launch of Windows NT; however, even after the introduction of newer hash schemes in later versions of Windows, it continued to be stored by default (for backwards compatibility) until the release of Windows Vista (which is why, as described in Section 2.2, the SAM file contains multiple hashes of

each password). Unless a sufficiently long password is used or this behavior is turned off, Windows XP installations still store this hash in the SAM file.

5.2 Weaknesses

There are multiple odd behaviors in the implementation of the LM Hash which render it more vulnerable to attack; one of the most significant is the fact that the hashes are not salted. This means that the same password will always produce the same hash on any Windows installation, which is what allows for the widespread availability of tools that use Rainbow Tables to crack passwords. Additionally, however, the protocol splits the password into halves and hashes each separately; this allows the two hashes to be attacked in parallel, in addition to enormously reducing the search space for an attacker (for reference, when dealing with 16-character passwords, this corresponds to moving from approximately .5 quadrillion years to just over twenty-seven days). The hash also converts all alphabetic characters to uppercase before hashing the password, which, while less significant than the fact that it hashes halves separately, still substantially reduces the search space (from about 2^{46} to about 2^{43} for a 14-character ASCII password).

5.3 Avoidance

The LM scheme cannot generate a hash for a password longer than fourteen characters; the simplest way to avoid its usage (in more recent versions of Windows) is to use a password which is longer than this. This is due to the fact that the two halves of the password are each used as DES keys which encrypt a fixed plain text; the space of 7-character ASCII passwords is encapsulated by the space of 56-bit keys. Additionally, when dealing with a system in which users cannot "reasonably" be expected to use passwords of sufficient length, versions of Windows which default to storing an LM hash for backwards-compatibility can have this behavior disabled.

6 Physical Attacks

Discovering the password to an account is clearly possible though direct attacks such as brute force, dictionary or rainbow tables. However, these are also far from trivial to implement, and there are serious time-space trade-off concerns. In certain situations it can be preferable to crack passwords through less traditional means. The most basic of physical methods would be to actually physically force someone to give up their password for a very important account. This type of literal attack is commonly referred to as "rubber hose cryptography" and is only effective when an attacker who can be very persuasive is able to gain physical access to a person who knows the password in question.

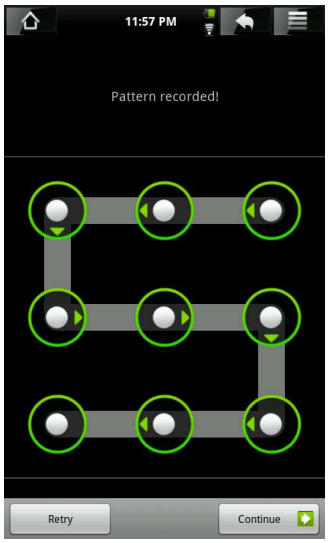
If there are moral or logistical issues with such an attack, there are less invasive ways to obtain passwords as well. A side-channel attack is one which gathers information that leaks out of the system in some way. The most basic form of this would be to observe someone typing in their password (perhaps with a well placed camera). With a tactic like this there is no need to look at password files, worry about salt values or create a chain of hashes; it is only necessary to be able to view the terminal in which someone inputs a password.

A more sophisticated form of side-channel attack would be to recreate a password from the sounds that typing on a keyboard generates[10]. Especially on older keyboards, each input key generates a slightly different sound upon being pressed. Researchers at Berkeley have utilized extremely sensitive microphones to decipher which keys have been pressed on a keyboard by just the sound. It's not even necessary to supply training data to the program (that is, data which informs the system which sounds correspond with which keys). The system is capable of listening

to ten minutes of normal keyboard input (such as someone writing a paper for a class) and reconstruct which sounds represent which keys using standard machine learning and speech recognition techniques.

In light of the multitude of techniques available for password cracking, it should be clear that there is no distinctly "best" option. The most useful password cracking technique will be highly dependent on the characteristics of the attacker and of the target.

7 Graphical Passwords



(a) A sample input for a graphical password.

(b) Extremely visible oil residue.

Figure 9: Android phone graphical passwords.

The advent of touch screen devices has brought about a new way of authentication that is commonly seen now on Android phone devices. These devices provide users with the option of authenticating using a graphical password, as shown in Figure 9. The default Android program requires the user to create a password which connects at least four dots in any order (though the user can make a longer password by connecting more dots). One benefit to graphical passwords such as this is that humans typically are much better at remembering patterns and pictures than attempting to remember a long string of random characters[7]. In addition, graphical passwords are difficult to attack in most traditional senses. Automating attempts at entry is difficult on a graphical password system, even if the search space is much smaller than typical text passwords. However, the Android graphical password scheme has been found to have a powerful security hole. Many users leave behind a visible oil residue from entering in their graphical password many times into the system. A stolen phone can potentially be broken into just by looking at the screen smudges.

8 Conclusion

In the preceding sections, we have discussed various facets of password-based authentication, including its origins, common implementations, and various attacks against it. It is worth noting, however, that the technology to create a password-storage scheme which will withstand any computational attack (within a reasonable time frame) has existed for decades. The prevalence of attacks against password hashes are attributable almost entirely to a combination of laziness (or ignorance) on the part of people who design and administrate systems which store passwords and laziness (or ignorance) on the part of people who make use of these same systems. Just as storing only salted hashes of passwords reduces the effectiveness of most computational attacks to the point where they are rendered nearly useless, the use of large passwords drawn from as large a search space as possible (i.e. including characters from as large a set as is available) greatly reduces the effectiveness of these same attacks as well. In addition, taking care when entering a password (i.e. not writing it on a post-it on your monitor, and checking for anyone who may be standing over your shoulder as you enter it) is sufficient to deter all but the most dedicated of would-be physical attackers (and greatly reduces the chance of such an attack being financially viable for the attacker).

References

[1] Anonymous. http://ophcrack.sourceforge.net/, July 2009.

[2] dazzlepod. http://dazzlepod.com/site_media/txt/passwords.txt, January 2012.

[3] Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980.

[4] Troy Hunt. http://www.troyhunt.com/2011/06/brief-sony-password-analysis.html, June 2011.

[5] Gershon Kedem and Yuriko Ishihara. Brute force attack on unix passwords with simd computer. In *Proceedings of the 8th conference on USENIX Security Symposium - Volume 8*, pages 8–8, Berkeley, CA, USA, 1999. USENIX Association.

[6] Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In *CRYPTO*, pages 617–630, 2003.

[7] Julie Thorpe and P. C. van Oorschot. Graphical dictionaries and the memorable space of graphical passwords. pages 10–10, 2004.

[8] David Wagner. http://www.nmrc.org/pub/faq/hackfaq/hackfaq-28.html, July 2003.

[9] Thomas Wu. A real-world analysis of kerberos password security. 1999.

[10] Li Zhuang, Feng Zhou, and J. D. Tygar. Keyboard acoustic emanations revisited. *ACM Trans. Inf. Syst. Secur.*, 13:3:1–3:26, November 2009.

Modern Game Console Exploitation

Eric DeBusschere, Mike McCambridge

Abstract

The goal of this paper is to provide a high-level, technical summary of the significant exploitations of the Xbox 360 and PlayStation 3. Few academic resources discussing console exploitation exist, especially resources considering the current generation systems, and thus a technical survey of successful exploitations represents a significant contribution to the academic community. The security of both the Xbox 360 and PS3 are discussed, along with the three main Xbox 360 exploits: the King Kong glitch/JTAG exploit, Timing Attack, and the Glitch Attack, as well as the three significant PS3 hacks: Hypervisor Exposure through glitching, PS Jailbreak, and the Root Key Discovery.

1 Introduction

Although the successful exploitation of the original Xbox drew considerable media attention and paved the way for sophisticated homebrew software developed by a large community of hackers, pirates, and Linux enthusiasts, modifying or hacking consoles of the current generation has never had the same appeal. Perhaps it is because modern gamers highly value the ability to play online, or the higher degree of technical expertise required to perform current modifications, or maybe the emulators, roms, and homebrew software just cannot compare with contemporary games and entertainment. Whatever the reason, the hacking of current generation consoles has been largely ignored, even in academic communities, despite the fact that modern exploitations are far more complex and interesting, and are mounted against far more sophisticated security systems, than ever before.

2 Modern Console Exploitation

2.1 Security

Gaming consoles of the current generation were built from the ground up with security as the foremost concern. A console that is easily exploited represents an extraordinary loss in profits for a system manufacturer due to piracy, especially for high-end systems such as the PlayStation 3 that are sold at a loss to the manufacturer with the expectation that profits will be redeemed with the sale of games. The Wii, for example, uses protection almost identical to Nintendo's previous generation console, the GameCube, and thus was hacked almost immediately. The consequence is that over one million Wiis run homebrew software, and a large percentage of these users play pirated games, representing 10s of millions of dollars in losses to Nintendo [25].

Because the Xbox 360 and PlayStation 3 are sold at a much higher price point than the Nintendo Wii, Microsoft and Sony have far more to lose if their systems are exploited, and because of this both companies implement the strictest security policies achievable at a reasonable cost. Software security is relatively inexpensive, and both the Xbox 360 and PlayStation 3 acheive complete software protection by only running signed code, encrypting memory, and utilizing firmware updates

to patch vulnerabilites. Additionally, the PlayStation 3 uses a layered Operating System and secure processors to acheive separation of privilege, forcing hackers to compromise multiple levels before gaining complete access. Hardware protection is more expensive, but both the 360 and PS3 use a secure boot process and a chain of trust seeded by a unique hardware console key. Furthermore, the Xbox 360 uses eFuses, which are physical bits that can be blown to create a hexadecimal value. The 360 blows an eFuse whenever an update is performed, physically preventing a console from downgrading [30].

Modern Console Security Practices

	360	PS3	Wii	Xbox
Per Console Key	●	●	●	
Manufacturer Private Key	●	●	●	●
Encrypted Memory	●	●		
Firmware Updates	●	●	●	
Secure Boot	●	●	●	●
Internet Banning	●	●		
eFuses	●			
Hypervisor	●	●		
Signed Executables	●	●		●
Security Coprocessor	●			
Encrypted Hard Disk		●	●	

Figure 1: **An overview of modern console security**

Although the Xbox 360 and PlayStation 3 have rigorous software and hardware protection, the most effective security method is through the internet. Both Sony and Microsoft enforce firmware updates via Xbox Live and the PlayStation network, allowing them to immediately respond to a vulnerability. Also, internet updates force hackers to choose between running an exploited system and playing online, or risk having their console banned. Sony has gone so far as to require updates to play the latest games, once again forcing hackers to choose between homebrew and a complete gaming experience.

2.2 Exploitation Strategies

2.2.1 Software Attacks

Practically speaking, all modern consoles are impenetrable from a software perspective. This is not to say that software attacks do not occur, but for the most part, they are memory overflows/bugs that are patched quickly via a software update. Oftentimes these mistakes are not made by Sony or Microsoft, but by third party developers that program games or manufacture console accessories. For example, until August 2011, all Xbox 360 exploits used a direct memory attack exposed by unchecked shaders in Peter Jackson's King Kong game [34].

When software attacks are mounted successfully, they often build off of a hardware attack. For example, if a hardware attack can dump the hypervisor or decrypt executables, the code can then be thoroughly examined for bugs that would normally be masked by encryption.

After a software vulnerability is discovered and patched, downgrading to a vulnerable kernel

becomes a possibility, and this strategy is used on both the PlayStation 3 and Xbox 360. Although this does not allow a user to play online or buy the most recent games, for some running Linux is more important.

2.2.2 Hardware Attacks

Almost all successful attacks on modern gaming consoles are hardware initiated, and utilize either a Timing Attack-measuring the time execution takes on branches-or a Glitch Attack-slowing down the processor and sending it an appropiate signal. These attacks are effective because physically securing a console is expensive, and likely is not a cost-effective strategy for combating piracy. However, one of the main dangers of hardware attacks is that they often cannot be fixed through firmware updates, and can sometimes be turned off, allowing a user to play online without risk of being banned. Because of this, both Microsoft and Sony fix exposed hardware vulnerabilites on newly sold consoles, so a successful hardware attack will not work on a newly purchased system [28].

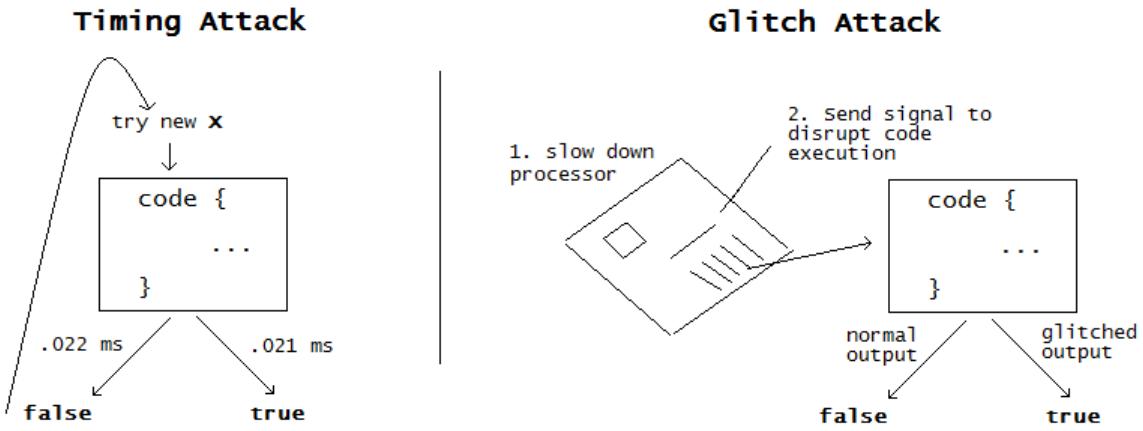


Figure 2: **Both Timing and Glitch Attacks use physical exploits to manipulate code execution.**

In most cases, hackers use a combination of software and hardware attacks to successfully penetrate a console's security. For example, GeoHot, a famous device hacker, used software commands sent via Sony's OtherOS to prepare the Hypervisor for exploitation. Then, he used hardware-based glitching to sidestep they Hypervisor's security checks. In August 2007, a hardware based timing attack was discovered on the Xbox 360 that allowed downgrading to an exploitable kernel. From here, the software based King Kong glitch could be performed [34].

2.2.3 Exploit Timeline

Although the Xbox 360 and PlayStation 3 present tremendous challenges to console hackers, both systems were eventually exploited. Interestingly, the PlayStation 3 was not successfully hacked for more than three years after its launch. Many speculators believe this is because Sony chose to support Linux on the PlayStation 3 until early 2010, only disabling it with the release of the new Slim design. Shortly after disabling Linux, the PlayStation 3 was hacked, and was found to have far greater vulnerabilites than the Xbox 360.

Although the Xbox 360 was exploited early in its lifecycle, the physical eFuses used by Microsoft to prevent downgrading made it exceedingly difficult for the exploit to propogate after it was patched via a firmware update. The real security mistake was allowing a particular manufacturer setting to bypass the eFuses. A timing attack discovered in mid 2007 allowed hackers to take advantage of this setting and downgrade to an exploitable kernel.

3 XBOX 360

3.1 Microsoft Security Practices

From a software standpoint, the Xbox 360 was designed to be totally secure. Microsoft implements a variety of security techniques designed to make exploiting its system as difficult as possible. First, the Operating System only runs signed code [30], so modifying even a single bit of a game or system software component renders it unusable. Second, no unencrypted executable code is ever written to memory. This is designed to prevent common exploitations via memory snooping. Third, all vulnerabilities are patched whenever a console connects to Xbox Live and downloads the latest update [15]. Additionally, new Xbox 360s are sold with the latest software patches and hardware modifications, and because of this, hacks usually only work on particular system/firmware combinations and require that those systems not update or connect to Xbox Live.

As one would expect due to the cost of secure hardware, the Xbox 360 is more vulnerable to hardware-based exploits, though it still presents some extraordinarily difficult obstacles to hackers. Specifically, the 360 is saddled with a Xenon CPU, which contains 768 bits of eFuse, a technology created by IBM. These individual hardware fuses are grouped into 12 Fusesets and blown in clusters of 8 to make a hexadecimal value that is used in combination with the CPU key to sign and verify firmware software and secure the boot process. The main purpose of these eFuses is to prevent unwarranted downgrading via irreversible hardware changes. When an Xbox 360 connects to Xbox Live and downloads a new update, an eFuse is blown, presenting both a software barrier by way of the new firmware as well as hardware barrier through the updated eFuses. This combination, along with a tightly-controlled boot process, make even low-level exploits challenging [4].

3.2 Successful Hacks

3.2.1 Playing Backups: Custom firmware for DVD-ROM.

It did not take long for hackers to write a custom firmware for many of the Xbox DVD-ROM drives (there are a variety of make/models) that simply returns a media value equivalent to what the XBOX 360 expects to receive from game discs for all DVD+R-DL [28]. The actual content of the disc must be identical to the original game, otherwise the digital signature will not remain intact. Note that this is in no way a compromise of the Xbox 360 Operating System or Hypervisor.

3.2.2 The King Kong Glitch (Exposed Nov. 16, 2006. Fixed Feb. 28, 2007)

Renders all Xbox 360s shipped before Feb. 28, 2007 that were not updated vulnerable to exploit.

A kernel update (4532) exposed a privilege escalation vulnerability in which an unsigned shader performs a series of unchecked memory exports in certain games, namely Peter Jackson's King Kong [15]. Normally, these memory exports are for writing the results of a pixel shader into physical memory, however the hack uses the shader to overwrite the idle-thread context and instructs the kernel to jump to a particular position in memory while retaining control of some of the registers. Control of all registers is done with a second jump to the interrupt restore handler. These jumps

are possible because, on a particular system call, a 64 bit input address is only verified with a 32 bit check value, leaving the upper 32 bits of the input vulnerable to exploit [30]. Once these two jumps are made, all registers can be filled with determined values, and unsigned code is loaded from a secondary location into memory and executed.

Although the King Kong Attack allows a hacker to take complete control of the Hypervisor and run unsigned code, it requires a user start their console using the King Kong disc everytime they want to enter the exploited state. Because the expoit is a DMA (direct memory access) attack, in theory it could be initiated by any hardware or software that has the authorization to directly manipulate memory, and, shortly after the vulnerability was discovered and exploited, modders found that it could also be triggered directly via hardware means [34].

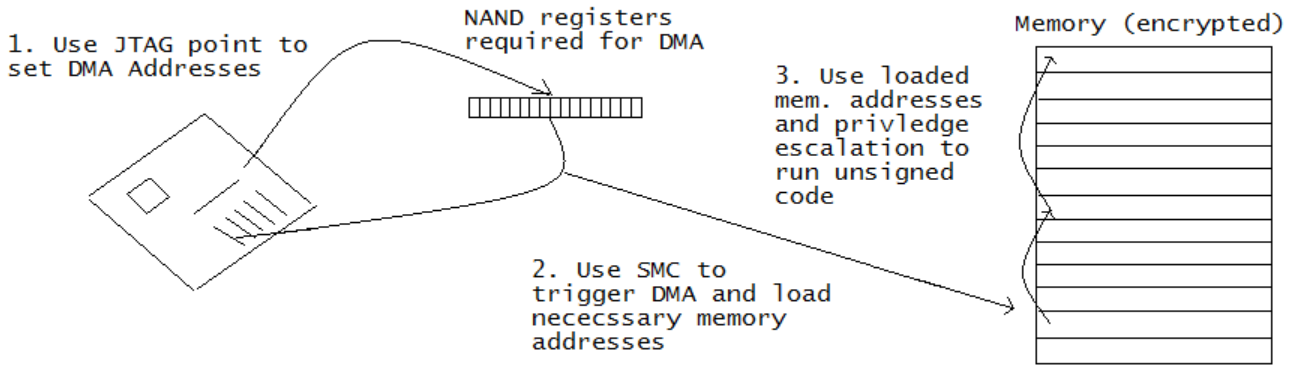


Figure 3: **A direct memory attack, possible with Kernel 4532.**

It had been known for quite sometime that the SMC (System Management Controller) could trigger a DMA. Unfortunately, the SMC cannot write the NAND registers necessary to enable the correct DMA, so it simply executes a DMA to the kernel's last DMA destination, which is usually address 0. A few months after the SMC was initially explored as a hardware trigger for the DMA attack, a separate group of hackers reverse-engineered the GPU JTAG (joint Test Action Group) and were able to successfully send it commands. JTAG was developed to build points into chips and facilitate the testing process. JTAG alone could not be used to trigger the DMA because its interface is disabled early in the boot process, however, Xbox hackers programmed the GPU JTAG with the DMA target addresses, and then used the System Management controller to initiate the DMA [34], completing the exploit in a faster, more efficient manner than continually using the King Kong disc.

The King Kong glitch was a good first step, but was discovered too late. By the time the general hacking community knew about the vulnerability, Microsoft had patched it for anyone that had connected to the internet and all new Xboxes.

3.2.3 Zero-Pairing and The Base Kernel

In early summer of 2007, the MfgBootLauncher mode was discovered. Essentially, this runs the base kernel-the initial kernel shipped with all Xbox 360s-whenever all of 2BL pairing block bits are set to 0. A flash image of this state could be extracted, but no games could be run, nor could the normal dashboard be entered. Technically, in this mode the 4BL refuses to apply update patches and thus enters the base kernel [3].

3.2.4 Microsoft Upgrades Security on Zero-Pairing

In what was most likely an effort to increase the security of the 4BL code in response to the discovery of MfgBootLauncher, the 4BL encryption key was changed to use the CPU-key for creation via a CB (4BL) update (CB 1920). This meant the CPU-Key was now required to decrypt the 4BL code during stage 2BL of the boot process.

This CB update also made two very interesting changes to the zero-pairing configuration. Because the CPU-key was now required to decrypt 4BL, simply setting all 2BL pairing bits to 0 no longer worked to enter MfgBootLauncher mode. However, Microsoft also modified the 4BL so that if Zero-Pairing mode was successfully entered (which now required the CPU key), any update-slots that are also zero-paired are applied, regardless of how many eFuses are set. In theory, assuming a hacker knew their CPU-Key, this allowed a bypass of the hardware eFuses [34]. Unfortunately, there was no method to discover a newly purchased Xbox's CPU-Key (unique to each console).

3.2.5 The Timing Attack (Exposed Aug. 2007. Fixed early 2008)

Renders all Xbox 360s shipped before December 2007 that were not updated vulnerable to exploit. It also rendered some newer 360s vulnerable depending on their CB version (has to be 1920). However, during this time Microsoft released consoles with new technical improvements (Zephyr, Falcon, Jasper, and Opus) which were patched on manufacture against the Timing Attack.

Microsoft made a critical error with CB version 1920. Because the CPU-key was now required to apply the Zero-Pairing configuration, Microsoft assumed it would be impossible to hack, and thus allowed Zero-Pairing mode to bypass the physical eFuses, and apply any Zero-Paired update [3].

This presented a "chicken and egg" problem for the eventual hackers. If a person knew their CPU key, they could run any kernel update they wanted and bypass the eFuses, however, in order to find a console's CPU-key, an earlier kernel version needed to be run-a person could extract their CPU key if they could run the King Kong exploit which required an older kernel.

To solve this dilemma, the hackers took an unmodified base kernel (which was extracted and saved during the initial discovery of the MfgBootLauncher) and patched certain values in the kernel update (SMC, KeyVault, CB, CD, and CE) with current system values, which could be found by taking a flash dump of the current system [34]. This created an unbootable base kernel, because the 4BL lockdown counter hashed from the physical eFuses still showed the current update, not the base kernel.

Although the problem was not solved, it had been shifted from finding the console's CPU-Key to finding a hash value of a lower 4BL lockdown counter, which would enable the base kernel to be loaded and Zero-Pairing state to be entered. Fortunately, this hash value was only 16 bytes long, and was checked with a simple byte-by-byte memcmp function. This meant that, if a timing attack could be executed, there were only 16 bytes * 256 different values per byte = 4096 necessary tries to decrypt the hash. Experimentation showed that a time difference of 2200 microseconds existed between a true and false byte value. A special flash emulator chip was built that would rapidly change the hash value and reboot the machine-re-flashing with a new hash-value on each try could easily have destroyed the system [2].

3.2.6 Successfully Downgrading and Extracting the CPU-Key

Once the hash value for the modified base kernel was found, it could be booted successfully. Because with CB version 1920 Microsoft now allowed any Zero-Paired update to be loaded from the base kernel, kernel 4532 could easily be loaded and the King Kong exploit could be performed. This

was the first time the Xbox 360 was hacked on a wide-scale, because, as discussed before, the King Kong exploit was discovered after it was patched, making it only available to those who had not performed a dashboard update in a long time [34].

3.2.7 Deriving CB 1921 from CB 1920. (Exposed Aug. 2009. Fixed Aug. 2009)

Renders all Xbox 360s of all types shipped before August 2009 that have not been updated vulnerable to exploit.

Microsoft realized the vulnerability with CB 1920 in early 2008, and most new consoles were manufactured to use a newer CB version, 1921, which used a memdiff function as opposed to memcmp to validate 4BL hash values.

Unfortunately for Microsoft, because the memdiff to memcmp was the only change between CB 1920 and CB 1921 on the original Xenon Xbox 360 machines, using the changes between the 1920 2Bl and 1921 2BL, along with a decrypted 1920 4BL, hackers were able to derive a zero-paired 1921 4BL which hashed validly. This was then ported to Xbox 360s with newer technical configurations (Zephyr, Falcon, Jasper, and Opus) by changing version numbers and slightly modifying other parts of the code [34].

At this point, all Xbox 360s shipped before August 2009 were vulnerable to exploit. However, around the same time Microsoft, likely in expectation of complete current model exploitation, released an invasive dashboard update that overwrote the entire first stage of the bootloader, thwarting all known exploits while statistically destroying one in a thousand consoles due to its intrusiveness [31].

Although slightly modifying the hacked bootloader allowed some newer consoles to utilize the timing attack, there were no major new Xbox 360 exploits for almost three years.

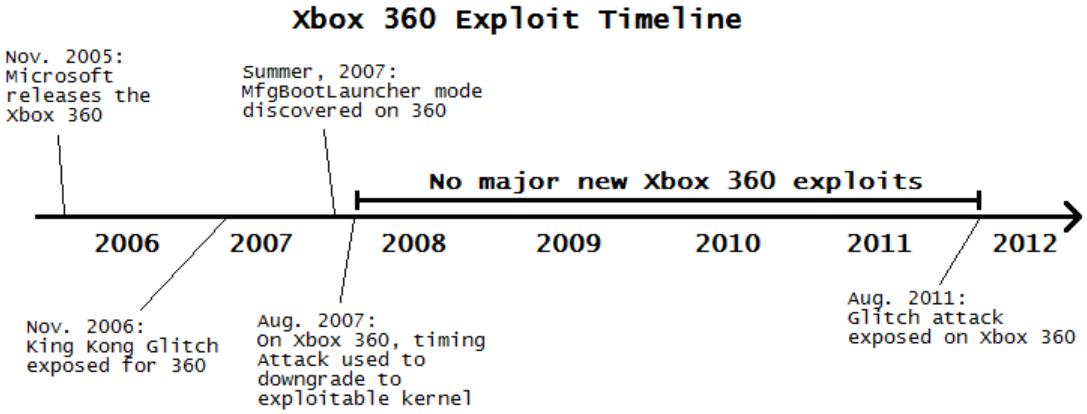


Figure 4: **No major new Xbox 360 exploits from Aug. 2007 to Aug. 2011**

3.2.8 The Glitch Attack (Exposed August 2011. Some newer consoles fixed)

Renders almost all current Xbox 360s vulnerable to exploit. This exploit cannot be fixed via a software update by Microsoft. Newly released consoles will most likely contain hardware changes that make the Glitch attack more difficult.

Because there had not been a successful exploitation of the Xbox 360 in almost two years, the Linux/open source community was in desperate need of a new strategy. A glitch attack had been

discussed since 2007, but most hackers thought it would be either require expensive hardware or be impossible to time correctly. One hacker, who had been working on the Xbox 360 for about eight months, became desperate enough to experiment with the glitch attack in the fall of 2010, and successfully ran unsigned code about a month after his first glitch attempt [12].

There are several variations of the glitch attack depending on the Xbox 360 hardware version. Generally speaking, the "Fat" consoles and the newer "Slim" models each require a different attack strategy, with some special processors requiring an even more specific approach.

On the "Fat" consoles, the bootloader glitched is the CB, or the second stage of the boot process. Asserting the CPU-PLL-BYPASS signal was found to slow the CPU execution down by a factor of 128. A counter is maintained, and at a certain count after a particular POST value, a CPU-RESET pulse is sent. If everything works correctly, this CPU-RESET pulse will correspond to the exact moment when the memcmp function is being performed to evaluate the hash of the flash image. Because the hash verification function uses a branch if equals 0 to decide whether or not continue booting, resetting the CPU registers at that exact moment means the branch instruction will evaluate to true, continuing the boot process even though the hash value in the flash image is incorrect [27].

The "Slim' consoles have an interesting change in that their CB, or 2BL, is split into two distinct phases, nicknamed CB-A and CB-B . Furthermore, a motherboard track for the CPU-PLL-BYPASS signal was not found, so another method to slow the CPU clock needed to be discovered. Experimentation showed that the HANA chip could be controlled in a way that would slow down the CPU, however only by a factor of 3, meaning that the CPU-RESET signal would have to be far more precise [35].

It was shown that, after a period of CPU-RESETS (sometimes taking as long as a few minutes), a flash image containing a normal CB-A and a patched CB-B could be validated in spite of its incorrect hash value using the glitch attack. The patched CB-B required, however, was difficult derive, because, compared with previous versions, "Slim" consoles were found to have stricter algorithmic protection of their CB code, utilizing RC4 encryption with the CPU-Key as the encryption-key [35].

In response, the hackers cleverly built upon their glitch of the "Fat" consoles by guessing that the first few bytes of CB code would be the same in the "Fat" and "Slim" consoles. Then, they simply encrypted a small amount of the known code and "dumped" the CPU Key. This exploit was successful because RC4 is a stream-cipher (it encrypts one byte at a time) and has known vulnerabilities when the content encrypted is the same as a previous instance [9].

Because the hackers guessed the first few bytes of plaintext for CB-B using their earlier exploit, they were able to obtain the pseudo-random key stream (the CPU-Key) and sign a patched CB-B . The patched CB-B is modified in a way that it always activates zero-paired mode, doesn't decrypt the CD (or 4BL), and doesn't halt the boot sequence on an incorrect CD hash, effectively allowing any unsigned code to run.

It is important to note that the glitch attack is the first known exploit that does not utilize the infamous King Kong glitch. In fact, the traditional exploits would be impossible at this point because newer CDs (or 4BLs) contain a hardcoded hash of kernel versions 4532 and 4548 (those susceptible to the King Kong exploit), and specifically check that those versions are not being loaded [5].

4 PlayStation 3

4.1 Sony's Security Practices

The Sony PS3 is similar to the Xbox 360 in that it is completely secure from a software point of view. All software running on the PS3 is encrypted, and, like the Xbox 360, the PS3 relies on hardware protection as the backbone of its security. Specifically, the PS3 uses the Cell Broadband Engine, which was one of the first mass-market processors designed with security as its primary focus [32]. Cell has eight SPEs (synergistic processing elements), which are booted in a way that all of the code controlling the SPE is entirely walled off from the remainder of the system, including the Operating System and Hypervisor. This enables code running on a particular SPE to be verified at any point in time in a way that is completely independent from the rest of the system. Cell stores a hardcoded master root key, which can only be accessed by a particular SPE if the code it is running has been verified and is confirmed to be unmodified [1].

4.2 Successful Hacks

4.2.1 The Cell Wall Remains Intact

The PS3 was the last console in the current generation to be hacked, remaining largely unexploited for an unprecedented three years. Its success may have been attributed to the fact that, until firmware 3.21, it allowed consumers to run Linux, much to the appeal of legitimate homebrew software users [25]. Most hackers who exploit gaming consoles do so under the banner of open systems, insisting that they be allowed to run homebrew software on any piece of hardware they purchase. For the most part, until firmware 3.21 (released in April of 2010), Sony allowed them to do so, and a variant of Linux running on the PS3 could access six of the seven SPEs, only being restricted from the main GPU [7].

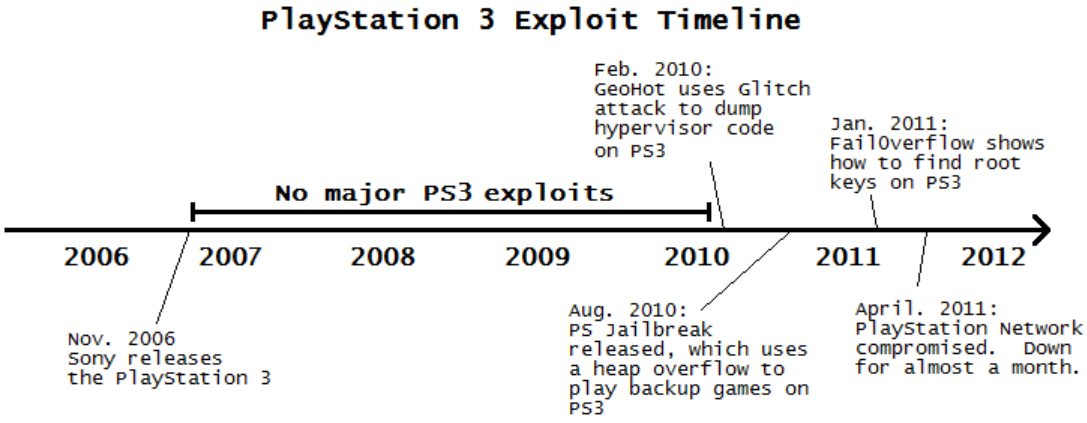


Figure 5: **The PlayStation 3 remained unexploited until February 2010.**

4.2.2 "Hello Hypervisor, I'm GeoHot."

A 21 year old hacker named George Hotz, infamous for his jailbreaking of the Iphone, decided to enter the PS3 hacking scene late in 2009. In a blog post in early 2010, he revealed how he

utilized the glitching technique to read/write arbitrarily to RAM and dump the entire contents of the hypervisor.

George's technique was a bit different from the glitch attack used to compromise the Xbox 360 a year and a half later. In fact, one could say it was superior in the sense that it did not require slowing down the processor and could be manually triggered. The pulse could be imprecise because the RAM was prepared in a way that extended the window of opportunity to the point where manual timing was sufficient.

The target of the attack was the hashed page table (HTAB), which, if compromised, would allow access to the main segment and eventually allow George complete control of memory mapping within the system. The entry point of the attack was OtherOS, Sony's tool which allowed a variant of Linux to be virtualized under the control of the Hypervisor [23].

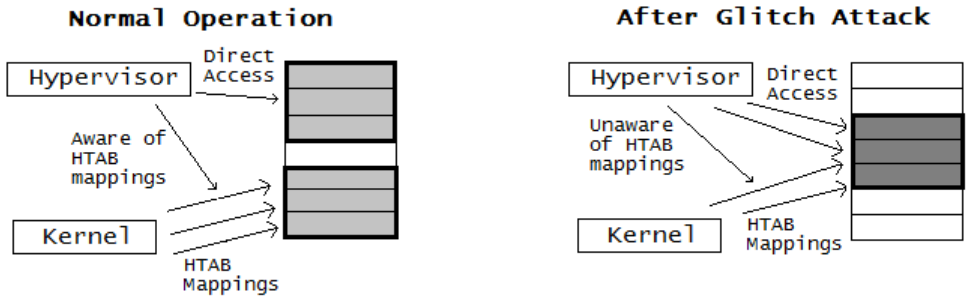


Figure 6: **Software based exploit preparation allowed the glitching to be triggered manually**

First, from OtherOS, George's software requested the Hypervisor create a large number of HTAB mappings, all of which pointed to a singular buffer it had previously requested the Hypervisor allocate. From here, the software had OtherOS request the Hypervisor deallocate the buffer without properly closing the HTAB mappings. None of this alone is enough to trigger an exploit, and under normal circumstances the Hypervisor would simply deallocate all of the mappings. However, George found that sending a pulse down the FPGA (Field-Programmable Gate Array) memory bus directly after instructing the Hypervisor to deallocate the numerous HTAB mappings causes some of the deallocation commands to fail [23], and a mapping to a deallocated buffer remains intact and invisible to the supervising kernel.

Next, George's software creates virtualized segments until it finds the segment in which the relevant HTAB is at the freed buffer's address. From here the software can overwrite the HTAB without being stopped by the Hypervisor, and the exploit writes HTAB entries allowing it complete access to the main segment, and thus all of memory once the Hypervisor switches to the corresponding virtual segment. Although this attack allowed the running of unsigned code, it did not fully compromise the Hypervisor, which would be required to run pirated/back-up games due to their on-disc encryption via a ROM Mark [23].

Sony's response to the exploit was to disable the OtherOS tool, enraging open hardware supporters enough that a lawsuit was later filed against Sony. Some speculators suggested that Sony dug its own grave by infuriating the open hardware community and giving hackers further motivation to crack the PS3 [25].

4.2.3 PS Jailbreak (Exposed August 19th, 2010. Fixed September 6th, 2010)

Renders all PS3s shipped before September, 2010 that have not been updated past firmware 3.42 vulnerable to an exploit that allows game backups to be stored and run from the consoles hard drive.

PS Jailbreak was the first exploit of the Playstation 3 that allowed a user to play backup games stored on the hard drive. It was available for purchase on a USB dongle, and the medium represented an emerging trend in PS3 exploit delivery. Although the exploit infiltrates the system via the USB port, it does not require the dongle be plugged in after the hack is successful, but the creators of the modchip designed the device so that if the dongle is removed the console will shut down. The device was made available in late August, 2010 at the steep price of $100 to $130. The high price tag gave hackers even more incentive to reverse engineer it and make it available for free, which was done shortly thereafter.

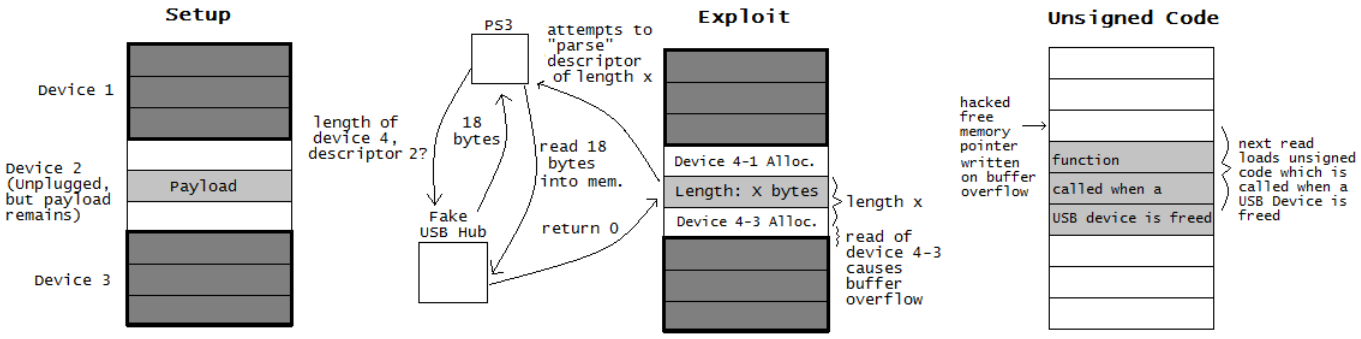


Figure 7: **Masquerading as a USB hub allows the PS Jailbreak to exploit a USB device verification vulnerability**

Initially, hackers thought the exploit was a stack overflow attack, but it was later confirmed to be a heap overflow [6]. The attack makes use of a short window of time after the PS3 is booted when it is does nothing but initialize USB devices. The dongle imitates a 6 port USB hub, and plugs/unplugs devices until the heap is corrupted and prepared for exploit.

Without being able to see the code responsible for USB device regulation, it was difficult for hackers to decipher exactly why the exploit was possible, however reverse engineering explained precisely the sequence of events that triggered it. First, the exploit software plugs in fake USB devices with large descriptors into the first three ports on its imitation USB hub. It subsequently sends a signal that the device on port two has been removed, freeing memory between the descriptor for the device on port one and port three. Then, a device is plugged into port four containing three large descriptors, which are allocated in memory between the descriptors for the devices on port one and port three [6]. This sequence of events sets up the heap for exploit.

Next, the second port four descriptor is changed in size from 18 bytes to 0, which causes the third port four descriptor to overwrite the malloc boundary tag of port three. This overwrite is possible because the PS3 determines the length of the descriptor from its first 8 bytes, and shortening the length of the descriptor between the first, length-determining read and the actual read causes the PS3 to attempt to parse the full descriptor from memory even though it was never loaded [25].

At this point the PS3 is now parsing memory it thinks is the second port four device descriptor, but is actually the payload from the previously unplugged port two device descriptor. The port two device descriptor's payload contains specially-planted bytes which tell the PS3 the second port four descriptor buffer it is parsing is extraordinarily long, and thus the PS3 overwrites the malloc boundary tag at the beginning of port three's descriptor. The attack is initialized so that the

substitute malloc boundary tag, which normally points to the next section of free memory, points to a function called after a USB device is freed.

Finally, the dongle sends a signal that an official PS3 service jig has been plugged into port five. A challenge key is sent to the dongle, and it responds with static data, which is allocated in memory at the next free location, planting unsigned code in memory. The dongle then unplugs the device on port three, which calls the overwritten function and completes the exploit [6].

The interesting aspect of this exploit was that it only compromised about 20% of the security system, ignoring the SPEs and even the Hypervisor, yet was still able to play pirated games by simply copying them to the hard drive and patching LV2 (the lowest kernel level) to run them from there.

Sony responded almost immediately to the PS Jailbreak, releasing firmware 3.42 on September 6th, 2010, disabling its use to anyone who updated and fixing the issues related to the Hypervisor and level 2 kernel security [8].

4.2.4 Root Key Exposure (Exposed January 2nd, 2011)

Renders all consoles shipped before January 2nd, 2011 that have not updated to firmware 3.56 vulnerable to complete exploit. In theory all future firmware should be exploitable, but Sony has added other obfuscation techniques and the PS3s main hackers are legally barred from pursuing further exploits on the console.

On January 7th, 2011 the hacking team Fail0verflow released an astonishing video in which they detailed how a hacker could take complete control of the PS3, even deriving the root keys.

Essentially, much of the code running on the PS3 is in the common ELF format (the classic Unix extensible linking format), but is signed with secure keys and only decrypted on an SPU in isolation from the rest of the system. While in theory this would make decrypting code extremely difficult, Fail0verflow found that the PS3 makes no attempt to verify that the lower level kernels (those which can ask for code to be decrypted) have not been compromised, so using the level 2 kernel exploit provided by PS Jailbreak, Fail0verflow could simply ask an SPU to decrypt any code it wanted access to [25].

As surprisingly simple as the SPU security hole was, the decrypted high-level code Fail0verflow now had access to revealed a far greater security flaw. As detailed before, the code running on the PS3 is in ELF format and signed using ECDSA (Elliptic Curve Digital Signature Algorithm). To maintain a high level of security, ECDSA requires a random number that is recreated each time code is encrypted. Unfortunately, Sony uses the same random number every time an ELF file is signed, making it trivial for a knowledgeable hacker to extract the root keys. George Hotz, the hacker who exposed the first real exploit on the PS3, released the root keys several days later, as well as a custom firmware (version 3.55) [29].

Although George Hotz's firmware specifically did not allow a user to play game backups, Sony's response was to sue both George Hotz and Fail0verflow, neither of whom returned to the scene. Additionally, Sony released a new firmware which patched many of the security flaws and attempted to disabled downgrading [8]. However, exploits have been revealed which allow downgrading any current firmware, and in theory all future software updates are exploitable.

5 Conclusion

Unfortunately for the console hacking community and Linux enthusiasts, game system modification has lost much of its appeal in recent years. This dampening trend is likely due to both the unprecedented security measures built into current generation systems as well as the increased cost

and risk associated with modern day console modifcations. Whereas the original Xbox could be modified with a relatively inexpensive, soder-free chip, users nowadays are often forced to perform more complex installations. Furthermore, those who do successfully install a mod chip are not only cut off from the latest games but risk a permanent ban from online gameplay. Considering that both the Xbox 360 and PlayStation 3 had three year periods in which they were largely unexploited, it is not hard to imagine a future in which gaming console security is too sophisticated or too much work for the hobbyist hacker to compromise. These future gaming systems will undoubtedly utilize an entirely online platform, forcing users to remain online even while playing in single-player mode, shifting the backbone of console security to the gaming network, a domain which can be aggressively monitored and quickly patched by its parent company.

References

[1] Playstation 3 secrets. `http://www.edepot.com/playstation3.html#PS3_Security`.

[2] Timing attack. `http://free60.org/Timing_Attack`, 2007.

[3] Xboxhackerbbs. `xboxhacker.org`, 2009.

[4] Fusesets. `http://www.free60.org/Fusesets`, oct 2011.

[5] How we run xell on up to date fat and slim 360s, aka the reset glitch hack. `http://www.xboxhacker.org/index.php?topic=16949.0`, 2011.

[6] Ps3 wiki. `http://ps3wiki.lan.st/index.php/Main_Page`, feb 2011.

[7] Playstation 3. `http://en.wikipedia.org/wiki/PlayStation_3#OtherOS_support`, mar 2012.

[8] Playstation 3 system software. `http://en.wikipedia.org/wiki/PS3_System_Software#Version_3`, mar 2012.

[9] Rc4. `http://en.wikipedia.org/wiki/RC4`, 2012.

[10] David Becker. Microsoft wants to hire xbox hacker. *CNET News*, Sep 2002.

[11] David Becker. Lindows ceo funds xbox hacking contest. *CNET News*, Jan 2003.

[12] GliGli. Glitched! `http://gligli360.blogspot.com/2011/08/glitched.html`, aug 2011.

[13] Joe Grand. *Game Console Hacking: Xbox, PlayStation, Nintendo, Game Boy, Atari, & Sega*. Syngress Publishing, 2005.

[14] Joe Grand, Ryan Russell, and Kevin Mitnick. *Hardware Hacking: Have Fun While Voiding Your Warranty*. Syngress Publising, 2004.

[15] Anonymous Hacker. Xbox 360 hypervisor privilege escalation vulnerability. `http://securityvulns.com/Qdocument211.html`, feb 2007.

[16] Ben Heckendorn. *Hacking Video Game Consoles*. Wiley Publishing, 2005.

[17] Ulrika Hedquist. Crackstation uses game console for hacking. *PCWorld*, Nov 2007.

[18] Andrew Huang. *Hacking the Xbox: An Introduction to Reverse Engineering*. No Starch Press, 2003.

[19] iQD. iwd: Ps3 is hacked - the urban legend continues. http://www.ps3news.com/forums/ps3-hacks-jailbreak/iqd-ps3-hacked-urban-%legend-continues-109555.html, jan 2010.

[20] Kakaroto. Status update on the ps3 4.0 hen. http://kakaroto.homelinux.net/2012/01/status-update-on-the-ps3-4-0-hen/%.

[21] David Kravets. First criminal trial over game-console modding begins tuesday. *WIRED*, Nov 2010.

[22] Ben Kuchera. Modder arrest a reminder that most console hacks are illegal. *ARS Technica*, 2010.

[23] Nate Lawson. How the ps3 hypervisor was hacked. http://rdist.root.org/2010/01/27/how-the-ps3-hypervisor-was-hacked/, jan 2010.

[24] Robert Lemos. Old game machine gets hack trick. *CNET News*, Aug 2002.

[25] Yifan Lu. 27c3 - console hacking 2010. http://vimeo.com/18278625, dec 2010.

[26] Jon Peck. Why hack your game console? *Free Software Magazine*, Nov 2006.

[27] Powerslave. How the rgh actually works. http://techmantis.net/forums/index.php?/topic/952-how-the-rgh-actually-%works/page__pid__11576#entry11576, sep 2011.

[28] Ranger72. What type of mod is right for me? http://forums.xbox-scene.com/index.php?showtopic=718103, dec 2011.

[29] Gregory Rasputin. Ps3 history. http://ps3history.consolehistories.net/, sep 2011.

[30] Michael Steil. Why silicon based seucrity is still that hard: Deconstructing xbox 360 security. http://www.youtube.com/watch?v=XtDTNnEvlf8, dec 2007.

[31] Michael Steil. Dangerous xbox 360 update killing homebrew. http://www.free60.org/index.php?title=849x_System_Update&redirect=no, aug 2009.

[32] Jon Stokes. Cell's security architecture: Ibm's prize and sony's achilles heel. http://arstechnica.com/old/content/2006/04/6694.ars, 2006.

[33] tmbinc. King kong exploit - how anon knew what to change? http://www.xboxhacker.org/index.php?topic=9714.msg62740#msg62740, may 2008.

[34] tmbinc. Smc hack. http://free60.cvs.sourceforge.net/viewvc/free60/imgbuild/hack.txt?revis%ion=1.1&view=markup, aug 2008.

[35] Xbox-Scene. The xbox 360 reset glitch hack - new homebrew hack! http://forums.xbox-scene.com/index.php?showtopic=735015, aug 2011.