Software Protection: How to Crack Programs, and **Defend Against Cracking** Minsk, Belarus, Spring 2014 Christian Collberg University of Arizona www.cs.arizona.edu/~collberg

© May 27, 2014 Christian Collberg

About me



PhD from

- Five years on the faculty at
- One year at



中国科学院自动化研究所

NSTITUTE OF AUTOMATION CHINESE ACADEMY OF SCIENCES





Professional Interests

Software Protection (tigress.cs.arizona.edu)

- Compilers
- Programming Languages
- Scientific Ethics
- Secure Provenance (haathi.cs.arizona.edu).

Personal Interests

- Travel (35 countries so far...)
- Photography:

```
www.cs.arizona.edu/~collberg/#travel
```

- Foreign Languages
- Music:



Education



 I teach courses on programming languages, compilers, computer security.

Contact me

- www.cs.arizona.edu/~collberg
- collberg@gmail.com
- Office: Room 571
- MSU, Sector B!
- See me if you want to talk about
 - this course
 - studying in the US
 - computer security research
 - travel, Russia, music, languages ...
 - ... anything, really!



This Course

Overview

- What is Software Protection?
- When do you need Software Protection?
- Making programs hard to read (obfuscation).
- Making programs hard to modify (anti-tamper).



When? Wednesday 18:00

When? Wednesday 18:00 Where?

When? Wednesday 18:00 Where? Auditorium 612

- When? Wednesday 18:00
 Where? Auditorium 612
- Why?

When? Wednesday 18:00
Where? Auditorium 612
Why? It will be super cool!

When? Wednesday 18:00
Where? Auditorium 612
Why? It will be super cool!
WWW?

When? Wednesday 18:00
Where? Auditorium 612
Why? It will be super cool!
WWW?

www.cs.arizona.edu/~collberg/

Teaching/bsuir/2014

What we will learn

- How to be crack code (be a bad guy!).
- How to be protect code (be a good guy!).
- Using software protection tools (Tigress).

What we will do

- I will lecture...
- In-class exercises...
- "Homework" exercises:
 - Hack this code!
 - Protect this code from hacking!

Prerequisites

- An understanding of C programming.
- Some assembly code knowledge is good.
- Some Unix, like shell commands, editing (emacs, vi), compiling (gcc), debugging (gdb).
- It's good if you know a little about cryptography, compilers, and computer security, but not necessary.

Class behavior

- Please interrupt with questions!
- You can ask a question in Russian (if someone will translate for me!).
- Bring a linux/Mac OS laptop to class if you want, but don't play games, check email, etc.

Today's lecture

- Software Protection Scenarios
- Software Protection Tools
- Overview of Obfuscation
- Overview of Tamperproofing



MATE Scenarios

Software piracy



• Alice is a software developer.

Software piracy



- Alice is a software developer.
- Bob buys one copy of Alice's program.

Software piracy



- Alice is a software developer.
- Bob buys one copy of Alice's program.
- Bob illegally sells copies to his friends.







 Bob removes license checks to be able to run the program whenever he wants.



- Bob removes license checks to be able to run the program whenever he wants.
- Alice protects her program so that it won't run after being tampered with.



• Alice's program contains a valuable trade secret (a clever algorithm or design).



- Alice's program contains a valuable trade secret (a clever algorithm or design).
- Bob, a rival developer, copies *M* into his own program (code lifting).



- Alice's program contains a valuable trade secret (a clever algorithm or design).
- Bob, a rival developer, copies *M* into his own program (code lifting).



- Alice's program contains a valuable trade secret (a clever algorithm or design).
- Bob, a rival developer, copies *M* into his own program (code lifting).

Digital rights management (DRM)





 A DRM media player contains cryptographic keys that unlock and play encrypted music files.

Digital rights management (DRM)



 A DRM media player contains cryptographic keys that unlock and play encrypted music files.

Digital rights management (DRM)



 A DRM media player contains cryptographic keys that unlock and play encrypted music files.
Digital rights management (DRM)



 A DRM media player contains cryptographic keys that unlock and play encrypted music files.

Digital rights management (DRM)



 A DRM media player contains cryptographic keys that unlock and play encrypted music files.













Protocol discovery



Alice sells voice-over-IP call minutes.

Protocol discovery



- Alice sells voice-over-IP call minutes.
- Bob examines the VoIP client to discover proprietary protocols to build his own rival client.

Protecting military software



 The military want to be able to track classified software.

Protecting military software



- The military want to be able to track classified software.
- In 2001, an EP-3 spy/reconnaissance plane landed on Hainan Island in China after a collision. The crew was unable to destroy all equipment.

Protecting military software



- The military want to be able to track classified software.
- In 2001, an EP-3 spy/reconnaissance plane landed on Hainan Island in China after a collision. The crew was unable to destroy all equipment.

























Ubruscat	Algorithm: Boolean Splitter :
Input File	Browse
Output File	▼ Browse
Methods	[All Methods] Select Methods
Obfuscate	Help
v	
This algorithm o	letects boolean variables and arrays and modifies all uses and





Tigress

tigress.cs.arizona.edu/#flatten 🔺 🕨 🙆 🧀 🗛 🖨 🖻 🖄 🖕 😚 tigress.cs.arizona.edu /# tigress.cs.arizona.edu/#overview The Tigress Diversifying C Virtualizer Christian Collberg What is Tigress? Transformations Tigress is a virtualizer for the C language that supports many novel defenses against well-known de-virtualization attacks, such as Rolles' and Rotalume. In addition to the virtualization transformation, Tigress contains a collection of traditional obfuscating transformations such as control-flow flattening, opaque predicate insertion, and function merging and splitting. These are used to make the generated interpreters stealthier, more diverse, and more resilient to attack.

In the past we have used Tigress to build a system for distributed application tamper detection via continuous software updates and we are currently using it for studies into diversity.

Design, Tigress is a source-to-source transformer built in OCaml on top of the CIL infrastructure. This has multiple advantages: Tigress supports all of the C language, including acc extensions: the transformed code can be easily examined, which is useful in a pedagogical setting; and Tigress' output, once compiled and stripped of symbols, becomes a good target for reverse engineering and de-virtualization exercises.

Diversity. Tigress is designed such that, from a single source program, it is possible to generate large numbers of highly diversified variants. This diversity is both static and dynamic, i.e. two variants will differ both in their machine code and in the resulting instruction traces. In essence, every decision Tigress makes is dependent on a randomization seed, controllable by the user.

Future, Tigress is under active development and we continue to add new features to the virtualizer. A further goal is to make Tigress the first freely available C language obfuscator that supports a large collection of classical obfuscating and tamperproofing transformations, the way that SandMark did for Java. The absence of a general tool for experimentation into the security and performance of software protection algorithms for binary code has severely hampered progress in the area, and we hope Tigress will fill this void.

Function Virtualization Control-Flow Flattening Function Splitting Function Merging Control-Flow Splitting

Argument Randomization Obfuscation of Literals

Libraries

Opaque Expressions Collecting Entropy

Usage

Controlling Tigress Machine Dependence Examples

Community

Help Crack Tigress! Learning More Contributors



Code Obfuscation

Code obfuscation

 To obfuscate a program means to transform it into a form that is more difficult for an adversary to understand or change than the original code.

Code obfuscation

- To obfuscate a program means to transform it into a form that is more difficult for an adversary to understand or change than the original code.
- Vague definition of *difficult*:

The obfuscated program requires more human time, more money, or more computing power to analyze than the original program.

Code obfuscation — Example obfuscated code

```
public class C {
 static Object get0(Object[] I) {
  Integer I7, I6, I4, I3; int t9, t8;
  I7=new Integer(9);
  for (;;) {
   if (((Integer)I[0]).intValue()%((Integer)I[1]).intValue()==0)
       {t9=1; t8=0; } else {t9=0; t8=0; }
   I4=new Integer(t8);
   I6=new Integer(t9);
   if ((I4.intValue()^I6.intValue())!=0)
     return new Integer(((Integer)I[1]).intValue());
   else {
     if ((((I7.intValue()+ I7.intValue()*I7.intValue())%2!=0)?0:1)!=1)
        return new Integer(0);
     I3=new Integer(((Integer)I[0]).intValue()%
           ((Integer)I[1]).intValue());
     I[0]=new Integer(((Integer)I[1]).intValue());
     I[1]=new Integer(I3.intValue());
```

Code obfuscation — Example original code

```
public class C {
    static int gcd(int x, int y) {
        int t;
        while (true) {
            boolean b = x % y == 0;
            if (b) return y;
            t = x % y; x = y; y = t;
        }
}
```

- An obfuscation tool turns the original code into obfuscated code.
- We want obfuscating transformations that make the program as hard to understand as possible.

Abstraction transformations

Destroy module structure, classes, functions, etc.!

Abstraction transformations

- Destroy module structure, classes, functions, etc.!
- 2 Data transformations
 - Replace data structures with new representations!

Abstraction transformations

- Destroy module structure, classes, functions, etc.!
- 2 Data transformations
 - Replace data structures with new representations!
- Control transformations
 - Destroy if-, while-, repeat-, etc.!

Abstraction transformations

Destroy module structure, classes, functions, etc.!

2 Data transformations

- Replace data structures with new representations!
- Control transformations
 - Destroy if-, while-, repeat-, etc.!
 - Dynamic transformations
 - Make the program change at runtime!

Obfuscation example: original program



After abstraction transformation

```
int main() {
    int y = 6;
    y = foobar(y,99,1);
    foobar(y,42,2);
}
```

```
int foobar(int x, int z, int s) {
    if (s==1)
        return x*7;
    else if (s==2)
        if (x==z)
            printf("%i\n",x);
}
```

 It appears as if main calls the same function twice!

After data transformation

```
int main() {
    int y = 12;
    y = foobar(y,99,1);
    foobar(y,36,2);
}
```

```
int foobar(int x, int z, int s) {
    if (s==1)
        return (x*37)%51;
    else if (s==2)
        if (x==z) {
            int x2=x*x % 51, x3=x2*x % 51;
            int x4=x2*x2 % 51, x8=x4*x4 % 51;
            int x11=x8*x3 % 51; printf("%i\n",x11);
        }
}
```

After control transformation

```
int foobar(int x, int z, int s) {
   char* next = &&cell0;
   int retVal = 0;
   cell0: next = (s==1)?&&cell1:&&cell2; goto *next;
   cell1: retVal=(x*37)%51; goto end;
   cell2: next = (s==2)?&&cell3:&&end; goto *next;
   cell3: next = (x==z)?&&cell4:&&end; goto *next;
   cell4: {
      int x2=x*x % 51, x3=x2*x % 51;
      int x4=x2*x2 % 51, x8=x4*x4 % 51;
      int x11=x8*x3 % 51;
      printf("%i\n",x11); goto end;
   end: return retVal;
```


Anti-Tamper

What is code tampering?

- Bob wants to modify the program binary so that it does something different than we want:
 - remove functionality (license check)
 - change data (password, cryptographic key)
 - add functionality (print, save game)
- Tamperproofing the code makes it stop working if Bob changes as little as a byte of the binary!

- Tamperproofing has to do two things:
 - detect tampering
 - respond to tampering

- Tamperproofing has to do two things:
 - detect tampering
 - respond to tampering
- Essentially:

if (tampering-detected()) abort
but this is too unstealthy!

Detection:



has the code been changed?

are variables in an OK state?

Detection:



- has the code been changed?
- are variables in an OK state?
- Response:



- refuse to run,
- crash randomly,
- phone home, . . .



Hash functions

```
uint32 hash1 (addr_t addr,int words) {
    uint32 h = *addr;
    int i;
    for(i=1; i<words; i++) {
        addr++;
        h ^= *addr;
    }
    return h;
}</pre>
```

Hash functions

```
void important_function () {
    ...
}
int main () {
    int v = hash (important_function,1000);
    if (v != 0x4C49F346) {
        crash the program
    }
    important_function(...)
}
```











Repairing Hacked Functions

```
void copy_of_important_function () {
}
void important_function () {
   . . .
}
int main () {
   int v = hash (important_function, 1000);
   if (v != 0x4C49F346) {
      memcpy(important_function,
             copy_of_important_function,
              1000)
   important function(...)
```

Checker network



- code code blocks
- c_i checkers
- r_i repairers

Skype's hash function

```
uint32 hash7() {
   addr t addr=(addr t)((uint32)addr^(uint32)addr);
   addr = (addr t) ((uint32)addr + 0x688E5C);
   uint32 hash = 0x320E83 \circ 0x1C4C4;
   int bound = hash + 0xFFCC5AFD;
   do {
      uint32 data=*((addr_t)((uint32)addr + 0x10));
      goto b1; asm volatile(".byte 0x19"); b1:
      hash = hash \oplus data; addr -= 1; bound--;
   } while (bound!=0);
   qoto b2; asm volatile(".byte 0x73"); b2:
   goto b3; asm volatile(".word 0xC8528417,..."); b3:
   hash-=0x4C49F346; return hash;
```

Next week's lecture

- Software Hacking Techniques
- Bring a Linux/MacOS laptop, if you have. Make sure gcc and gdb have been installed.
- Look at Exercise 1 files on the website.