LINUX AND C++ UNDER ATTACK!

Professor Ken Birman
CS4414 Lecture 26

IDEA MAP FOR TODAY

SYN ACK attacks

Stack Overflow Exploits

Kernel Level Exploits

Key Theft.

Covert Channels

HOW TO ATTACK LINUX SYSTEMS

Limited only by creativity!

- System programs that have "backdoor" control features. *May be much more common than we realize.*
- Code that does a poor job of checking argument lengths
- Programs that get confused by certain mixes of parameters
- System calls that can be tricked into returning in privileged mode

HOW TO ATTACK LINUX SYSTEMS

… and that’s not all!

- Ways to replace a standard program with a non-standard version, and then perhaps tricking some tool into running it
- Ways of crashing the machine, or making it run extremely slowly
- Tricking a person into revealing a password, or resetting it
- Tricking someone into executing a compromised piece of code

CORNELL'S HISTORY IN HACKING

The very first Internet "worm" was born at Cornell!

Robert Morris, a new PhD student who arrived a bit early was bored and was fooling around.

His decided to create a new lifeform: a little Internet program that might still be wandering around in hundreds of years.

GOALS

His “worm” would just be a single process, or maybe a few

It would live on some machine for a while, then jump to its next host.

So it would wander the Internet… forever…

HOW IT WORKED

His “worm” would be installed on some computer and would use the “at” command to schedule itself after a random sleep.

It would scan /etc/hosts to look for machines reachable from this one. It would then try to “jump” to the new host.

HOW IT WORKED

How to move from place to place

- ssh or rsh or rcp: Copy itself (or perhaps login first, then copy itself)
- Exploit email issue: Sendmail had a remote-access feature for use in debugging: it could copy files in, or out, via a special command.
- Login in as admin or root (try some common passwords like admin, guest, secret, nullpass, etc)
- Bug in the “finger” program allowed it to copy a file in.

BUT OF COURSE, THE WORM MIGHT “FAIL”

Robert worried that various things could kill his worm.

An administrator might notice and remove it.

It could jump to a machine just as that machine crashed and was removed from the system permanently.

SO…

He decided that with some small probability, the worm should “duplicate” itself by spreading to two machines, or reinfecting a machine where it already was installed.

He did this by picking a random number.

His intended value for R_0 was around 1.001

THEN WHAT?

He tested his program… it immediately escaped into the wild!

"R_0" was much larger than anticipated. Closer to 2.5. *But even 1.001 would have been much too big.*

Within hours, the worn spread to every Unix machine in the world. And continued to spread: it reinfected them again and again.

THEN WHAT?

He tested his program… it immediately escaped into the wild!

“R an anticipated. Closer to 2.5. *But even 1. much too big.*

These were “pre-Linux” days, but Linux was born as an open-source clone of Unix, so it isn’t immune to such issues

Within hours, the worn spread to every Unix machine in the world. And continued to spread: it reinfected them again and again.

COULD THIS CAUSE HARM?

Infected machines quickly became overloaded and crashed. If rebooted, they crashed again.

Some Linux machines run respirators and dialysis units and X-ray units in hospitals. Some run floodgates for dams.

Linux computers control traffic lights in many cities. Some control power grid components or weapons systems.

THE WORM WAS ACCIDENTAL…

It was a dumb idea, illegal, and it could have caused deaths.

But since then, many viruses have been deliberately designed using similar ideas!

Some have infected and damaged huge numbers of machines. And they can sweep the vulnerable machines within minutes.

WHY SOME COUNTRIES CREATE WORMS

Many people have heard about Stuxnet. It was used to disrupt a nuclear weapons facility in Iran.

Some virus attacks are malicious. These often originate as part of geopolitical disputes between countries and are a form of warfare.

HOW KEN GOT TO "CHAT" WITH ASH CARTER

Sec. Def. Ash Carter

Secretary of Defense Ash Carter was on NPR.

Ken called in and we talked for a few minutes.

Question: Why are the US and Russia hacking each other's power grids, and why is it so "open"

CARTER DOCTRINE

If you hack us, we'll do even worse to you.

And we might not limit ourselves to exact symmetry.

And we'll talk to the NY Times about it to make sure you don't miss that we are doing it, since our techniques are very subtle.

RUSSIA IN UKRAINE

Russia decided to flex its muscles….

They devastated the power grid control systems in Ukraine.

Ukraine's power control systems had to be rebuilt from scratch!

SECURITY

ILLUSTRATION: CURT MERLO

ANDY GREENBERG SECURITY 06.20.2017 06:00 AM

How an Entire Nation Became Russia's Test Lab for Cyberwar

Blackouts in Ukraine were just a trial run. Russian hackers are learning to sabotage infrastructureand the US could be next.

THE CLOCKS READ zero when the lights went out.

It was a Saturday night last December, and Oleksii Yasinsky was sitting on the couch with his wife and teenage son in the living room of their Kiev apartment. The 40-year-old Ukrainian cybersecurity researcher and his family were an hour into Oliver Stone's film *Snowden* when

https://www.wired.com/story/russian-hackers-attack-ukraine/

HOW DO THEY DO IT?

Sadly, computer systems are very easy to attack.

Understanding this will help you build software that won’t be quite so “porous”!

EXAMPLE: STACK OVERRUN EXPLOIT

Suppose a C or C++ program reads data from a command line or file, and needs to turn something into a string.

The data is in a char* buffer, so in a normal situation, the program allocates memory (strlen(s)+1 bytes) and calls strcpy.

But sometimes people do other things

EXAMPLE: STACK OVERRUN EXPLOIT

Suppose that “Device names” are limited to 15 characters (plus 1 for a null), and the application is constructing a struct.

The struct might have a space for a 16 character name in it.

But in this case it would be easy to skip the strlen(s), so the program might get tricked into copying a much longer string.

WHAT HAPPENS WITH A STRING OVERFLOW?

Strcpy won’t notice: it just copies beyond the end of the array.

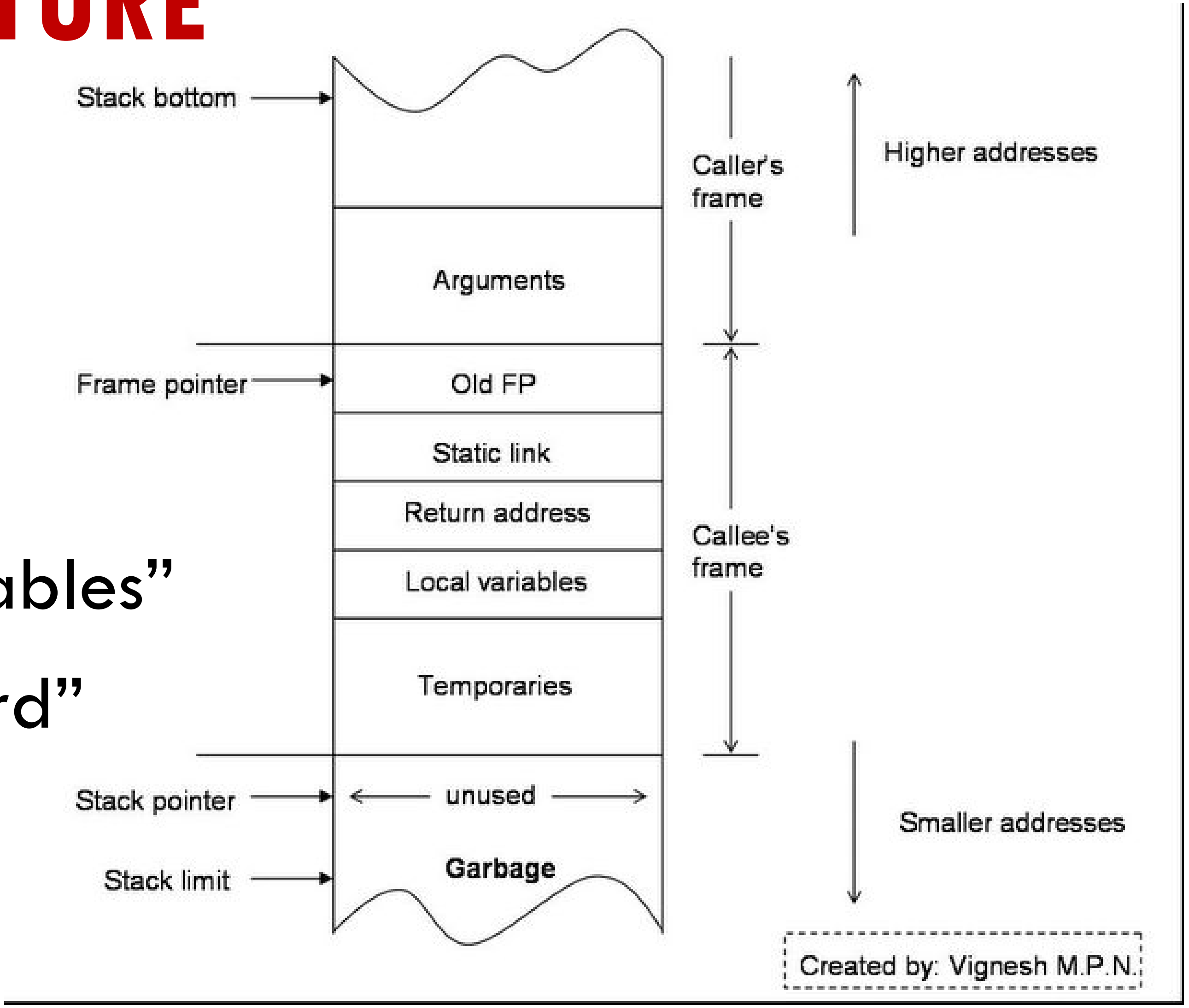
… where is the array in memory, and what is beyond it? In a stack overrun exploit, the struct would be on the stack, and because stacks grow in the “downward” direction, the saved registers and return PC are in the “upward” direction

STACK OVERRUN PICTURE

Our array is allocated in "callee's frame.". Smaller addresses: top of the stack.

Array would be in "local variables"

Overflow would occur "upward"

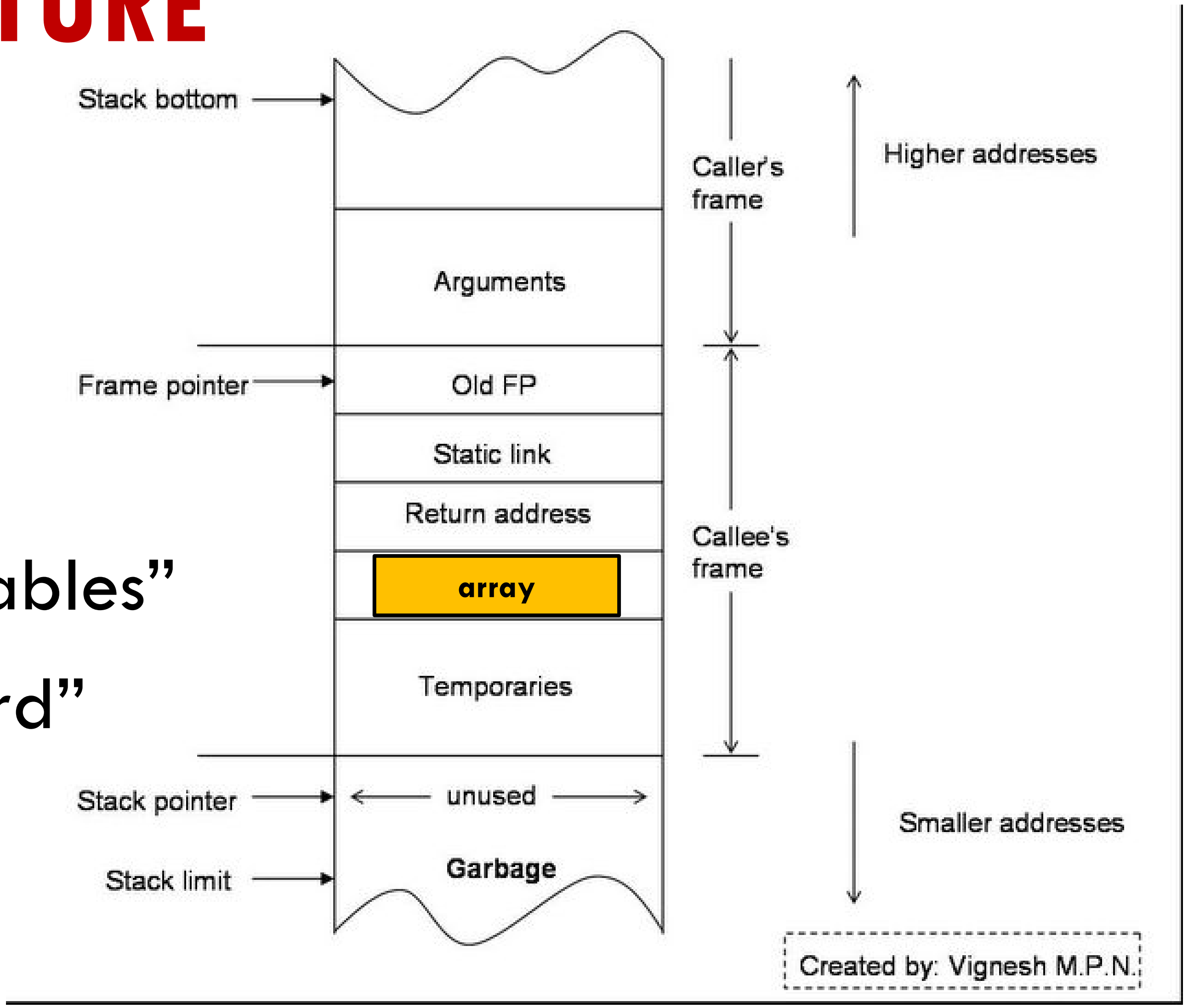


STACK OVERRUN PICTURE

Our array is allocated in "callee's frame.". Smaller addresses: top of the stack.

Array would be in "local variables"

Overflow would occur "upward"

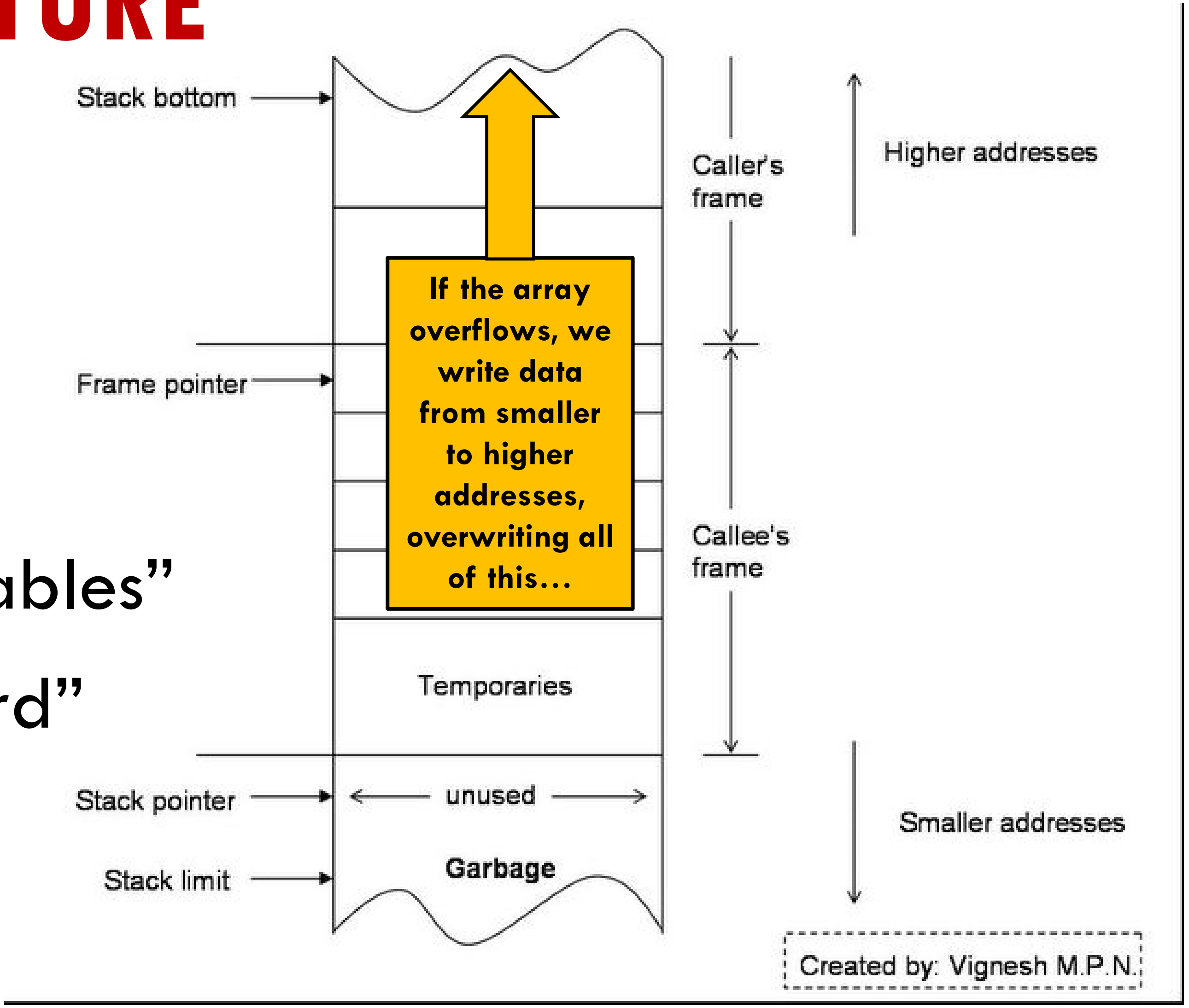


STACK OVERRUN PICTURE

Our array is allocated in "callee's frame.". Smaller addresses: top of the stack.

Array would be in "local variables"

Overflow would occur "upward"

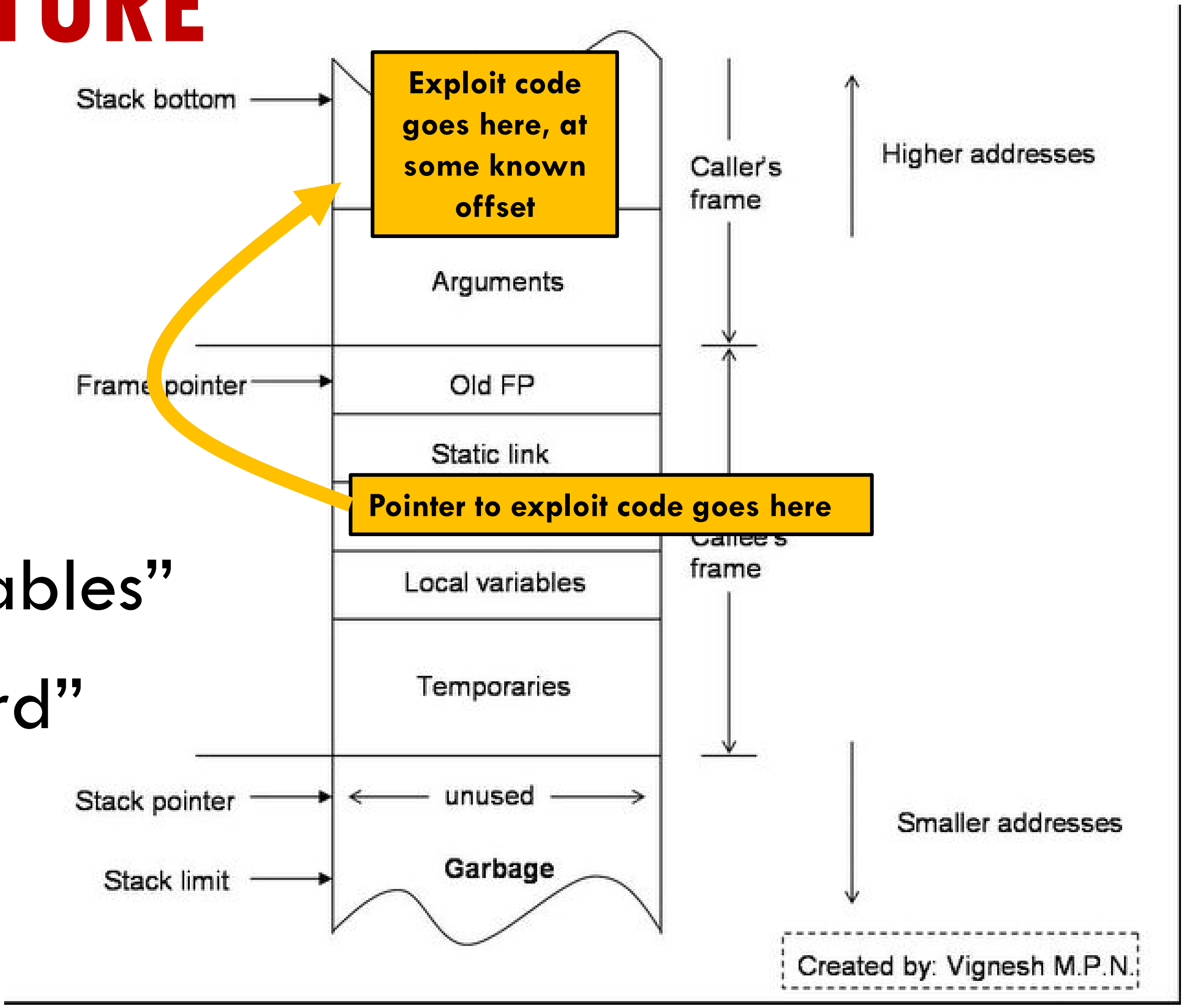


STACK OVERRUN PICTURE

Our array is allocated in "callee's frame.". Smaller addresses: top of the stack.

Array would be in "local variables"

Overflow would occur "upward"



HOW WOULD WE GUESS THE ADDRESSES?

No need! Many applications start up in a predictable way.

This determinism means that every single time, they call the "read a command" method in the same state.

So… the hacker just takes gdb and finds the address!

NOW WE HAVE OUR WHOLE ATTACK…

If the attacker has a way to know where this stack generally lives in memory, they can copy their own “bootstrap” program in, and put a jump to the start of it into that return pc.

When the “read the input” method tries to return, the exploit takes control of the process.

UNIX, LINUX, C AND C++ ARE FULL OF RISKS LIKE THIS!

It is easy to say "always make sure the char* object won't overrun the string" but this rule depends on a human being!

C++ is strongly type checked… mostly. But strcpy is an unsafe operation and type checking won't catch such issues.

And because C++ is very fast and light weight, it certainly won't check for array indexing errors!

THERE ARE SOME “REMEDIES”, BUT THEY AREN’T WIDELY USED

Stack and address space randomization: by allocating some random sized chunks of memory, we can easily make addresses much less deterministic and repeatable.

We can mark code segments as execute-only, and data segments as non-executable.

There are tools to check for buffer overflows (**valgrind** is great!)

NSA ULTRA-HACKER REMARK

"Sure… it would help if people would learn to lock the front doors. But keep in mind that we also know how to climb in the windows, or up to the balcony.

… And in the worst case we can break a window or cut an actual hole in the wall.

It isn't so easy to keep us out!"

FROM A SECURITY COMPANY: SECURITY INNOVATION, INC

Relative Paths in Command Execute in Unix

Suppose that some program with superuser privilages includes this sequence of lines:

```
        system(“cat /etc/shadow > /tmp/shadow.tmp”);
         system(“chmod 600 /tmp/shadow.tmp”);
```

This seemingly trivial logic is full of risks!.

FROM A SECURITY COMPANY SECURITY INNOVATION INC

What if the attacker put a symbolic link from /tmp/shadow.tmp to some file that normally can't be overwritten?

Relati

Suppose that some program with ser privilages includes this sequence of lines:

```
system("cat /etc/shadow > /tmp/shadow.tmp");
 system("chmod 600 /tmp/shadow.tmp");
```

This seemingly trivial logic is full of risks!.

FROM A SECURITY COMPANY SECURITY INNOVATION, INC

The official Linux "cat" lives in /bin. But suppose the attacker put a fake version of cat in the current directory? It will be discovered first!

Relative

Suppose that som gram with superuser privilages includes this sequence of es:

```
system("cat /etc/shadow > /tmp/shadow.tmp");
system("chmod 600 /tmp/shadow.tmp");
```

This seemingly trivial logic is full of risks!.

FROM A SECURITY COMPANY SECURITY INNOVATION, INC

Relative P

Chmod is also using a "relative" path notation.

Suppose ... ges includes this sequence of line

system("cat / /shadow > /tmp/shadow.tmp");

system("chmod 600 /tmp/shadow.tmp");

This seemingly trivial logic is full of risks!.

FROM A SECURITY COMPANY SECURITY INNOVATION, INC

Relati

Race condition. The file permissions will initially be 666 after the cat and before the chmod executes.

Suppo ges includes this sequence of lines:

```
system("cat /etc/shadow > /tmp/shadow.tmp");
system("chmod 600 /tmp/shadow.tmp");
```

This seemingly trivial logic is full of risks!.

WHAT MIGHT THE ATTACKER PUT INTO THIS FAKE VERSION OF CAT OR CHMOD?

The commands will run with the same privelage as the parent program, so the hacker will take advantage, perhaps by having "./cat" contain this (and with permissions 777)

```
#/bin/sh
/bin/cp /bin/sh /tmp/rootshell
/bin/chmod 4777 /tmp/rootshell
```

… leaves a "hook" for obtaining an "su" shell in /tmp

NOT EVERY “RACE” IS AS EASY TO NOTICE

Suppose that our superuser program does this:

```
system(“/usr/bin/cat /etc/passwd | awk ‘{print $1}’
          | xargs `email –s “fractus will be down for service tonight”’);
```

This will send email to each user permitted to use this system with subject “fractus will be down for service tonight.”

NOT EVERY "RACE" IS AS EASY TO NOTICE

Cat has a full pathname, but awk, xargs and email don't. Same issue but harder to notice

```
system("/usr/bin/cat /etc/passwd | awk '{print $1}'
          | xargs `email –s "fractus will be down for service tonight"");
```

This will send email to each user permitted to use this system with subject "fractus will be down for service tonight."

OTHER COMMON ISSUES

Many examples of sloppy memory management and sloppy checking on pointers and copying.

Frequently found code that ignored error returns from system calls.

They found that obscure Linux features (like symbolic file links, international text strings with characters from many fonts) were often sources of developer confusion, causing errors that could be attacked.

ANCIENT BUGS

By just reading Linux source code, it is easy to find bugs.

The Linux sendmail bug originated in 1980, more or less. The Morris worm exploited it in 1988. By then Eric Allman was focused on a startup and sendmail was mostly replaced by a follow-on system.

Yet the sendmail daemon was still launched on Linux at startup!

WHY TRY TO CRASH A SYSTEM?

Attacks sometimes seek to take a computer over, but crashing a computer that has an important role can be just as effective.

For example, if that computer has an important server, crashing it means the server will also be down.

Now the attacker has crippled the company and could demand ransom. If the server was doing building security… then the security system just shut off.

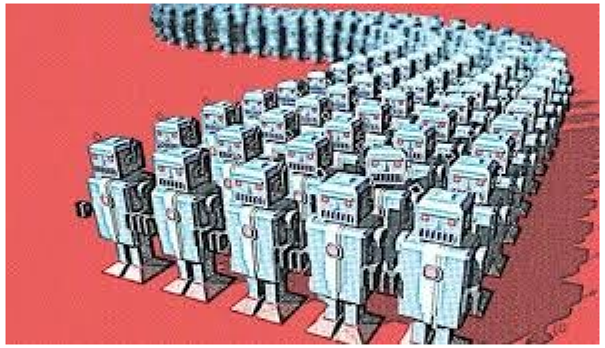
WORMS AND VIRUSES

These are a kind of attack that tries to spread itself from machine to machine.

A worm is usually passive, although it may leave the machine open for the attacker to take control later.

A virus deliberately does damage to each machine, or tries to steal information like bank accounts, passwords, etc.

BOTS

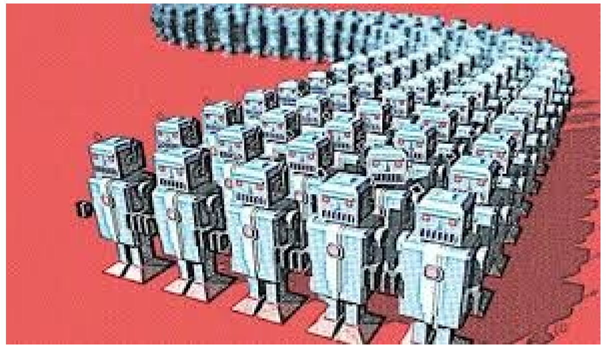


After taking control an attacker might not damage the machine

Many attacks are done to leave a form of “remote control” endpoint called a “bot”.

It allows the attacker to tell the machine to do things later… by some estimate, millions of machines are bots.

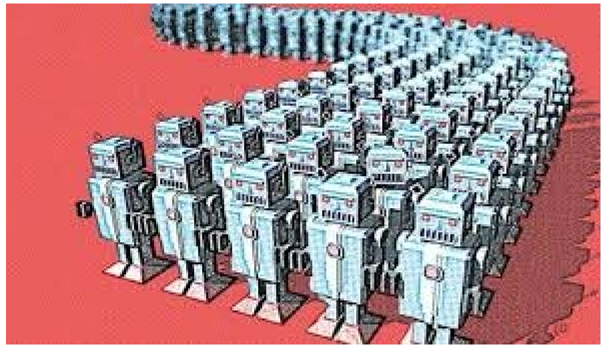
UKRAINE BOTS



In Ukraine, once Russian bots were installed, they left them waiting for instructions.

When told to attack, the bots damaged the hardware (by changing the “firmware” into garbage). The infected machines crashed and couldn’t be restarted.

OTHER PURPOSES?

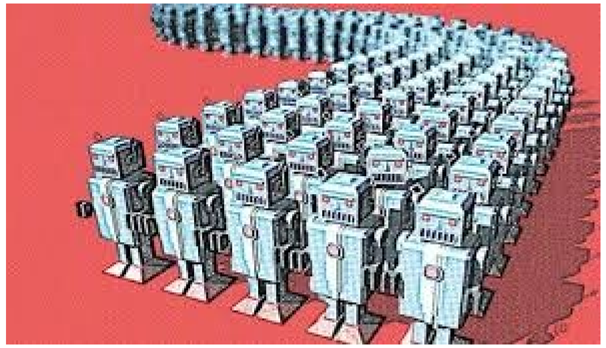


Some bots are normal personal computers.

Instead of telling the bot army to do things, perhaps the attacker uses the bots to search for interesting files.

…. Like files with bank-related stuff. Or military intelligence.

DDOS ATTACKS



A distributed denial of service attack is another way to leverage a bot army.

Attacker demands a ransom from Amazon or similar company.

If they don't pay… the bots go shopping

FILE ENCRYPTION ATTACKS

In an attack of this kind, a virus or bot encrypts all the data files, and the attacker demands a payment for the decryption key.

Without payment… the files never get decrypted (and sometimes, even *with* payment, they never do!)

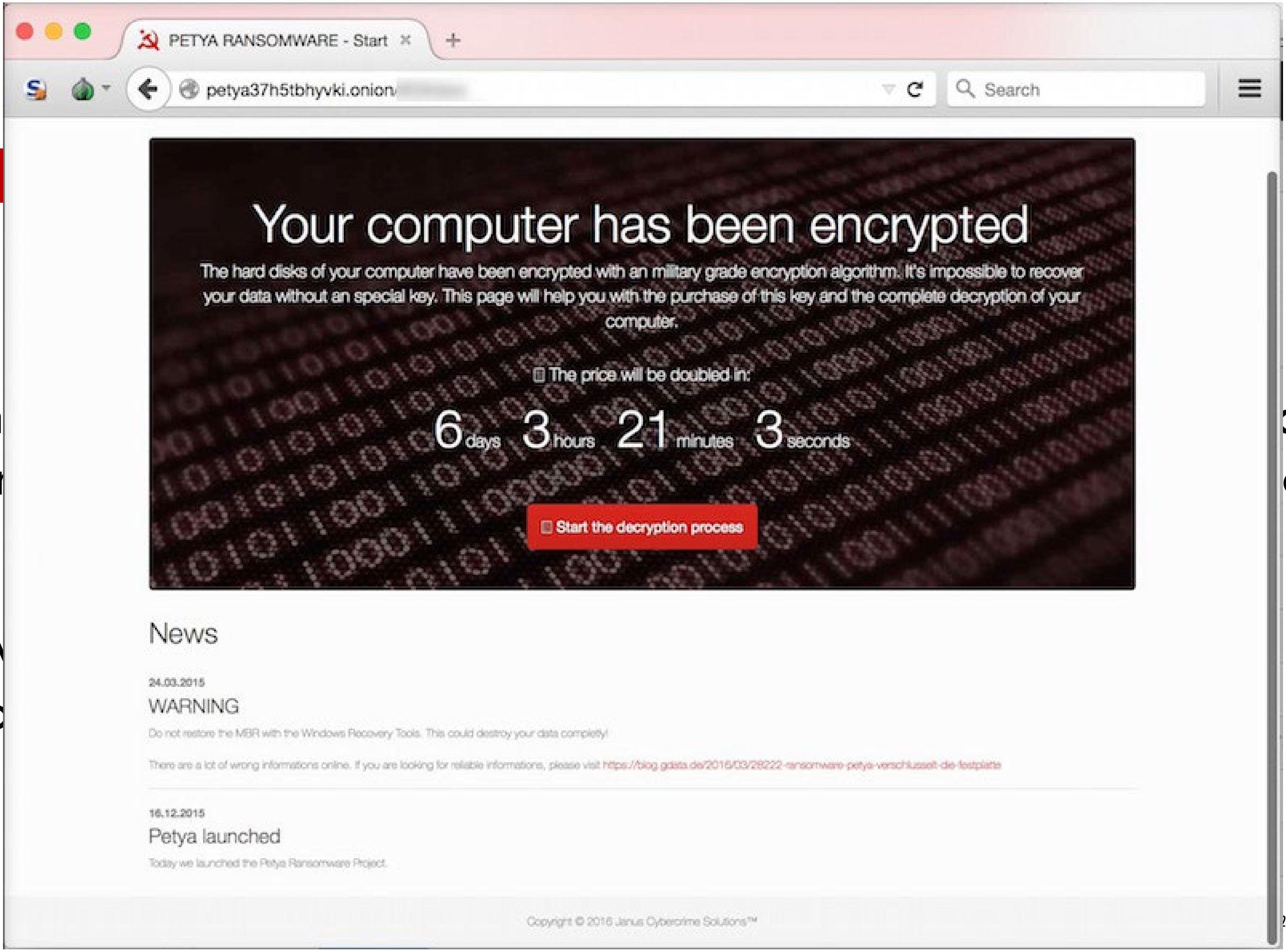
FI

In a files,

ar ey.

W

so


PETYA RANSOMWARE - Start

petya37h5tbhyvki.onion/[redacted]

Your computer has been encrypted

The hard disks of your computer have been encrypted with an military grade encryption algorithm. It's impossible to recover your data without an special key. This page will help you with the purchase of this key and the complete decryption of your computer.

The price will be doubled in:

6 days 3 hours 21 minutes 3 seconds

News

24.03.2015

WARNING

Do not restore the MBR with the Windows Recovery Tools. This could destroy your data completly!

There are a lot of wrong informations online. If you are looking for reliable informations, please visit https://blog.gdata.de/2016/03/28222-ransomware-petya-verschlusselt-die-festplatte

16.12.2015

Petya launched

Today we launched the Petya Ransomware Project.

Copyright © 2016 Janus Cybercrime Solutions™

STEALTH WEB SURFING ATTACKS

It is illegal to possess many kinds of information.

Depends on the country, but this could include child porn or violent videos, how-to manuals for creating bombs, subversive political materials, terrorism plans, etc.

What if the attacker downloads that sort of stuff, then calls the policy and denounces you? They find it on your machine…

STEALTH WEB SURFING ATTACKS

In fact a bot could even visit those sites every day, and download stuff for months or years

That leaves a trail of how active you are in their "group". A bot could even *upload* child porn or other illegal things.

Then, bot erases itself. And the attacker calls the police.

CAN WE FIGHT BACK?

In the next two (and final) lectures, we'll see a few excellent ideas that really can help.

But one idea worth mentioning today is *synthetic diversity*.

Notice that the buffer flow attacks depended on being able to know the contents of the address space of the victim program.

CAN WE MAKE ADDRESSES UNPREDICTABLE?

Yes, we can! In fact it is easy.

The compiler could randomly put some gaps into memory at compile time, just by inserting objects we don't plan to use (make them cache-line-size * N to avoid causing alignment errors).

So could the linker and even the Linux loader. Entire segments can show up at unexpected addresses.

WHY DOES THIS HELP?

Think about the buffer overflow attack: it can trick the victim into jumping to some address, but now the attack has no good target address it can be sure of!

Of course, some attacks inject position-independent code that uses system calls to download a bot and execute it, but we can prevent that too, by marking data segments as “non-executable.”

WOULD THIS PREVENT JIT APPROACHES?

Java and some other languages do just in time compilation.

They generate code, then execute it– so “data segments” end up being “executable segments”.

But Linux already has an answer: a program can ask to change permissions on a data segment, from pure data to executable instructions. So synthetic address diversity really works!

HACKING SUMMARY

Sadly, it is very easy to do! There are even classes on this and whole conferences (like Defcon)!

"Rootkits" readily available on the dark web.

Learning about Linux protection features, and using them, plus writing better code, is the most effective option. In Lecture 27 we will learn about Rust, a new C++ variant with stronger protection.