Welcome to

4. Network Attacks and Advanced Vulnerabilities

KEA Kompetence Penetration Testing

Henrik Kramselund Jereminsen hkj@zencurity.com @kramse

Slides are available as PDF, kramse@Github

4-network-attacks-and-adv-vulns-2021.tex in the repo security-courses

Plan for today

Subjects

- Network Attacks
- Nmap Workshop materials
- Detecting network attacks
- Talk about advanced Vulnerabilities

Exercises

- From the book mostly, and only Linux - on our Debian
- Install and prepare OWASP JuiceShop

Reading Summary

- Grayhat chapters 12: Advanced Linux Exploits
- Grayhat chapters 13: Windows Exploits
- Grayhat chapters 14: Advanced Windows Exploits – not today!

Reading Related resources:

- *Return-Oriented Programming:Systems, Languages, and Applications*
- *Removing ROP Gadgets from OpenBSD*

Goals for today

.

Take a slow day

Talk and explain in detail buffer overflows, and some of the parts

Go through examples from the book

Reproduce the parts we can

Redo buffer overflow on ARM, Raspberry Pi – talk about shell code for systems

Go through the OpenBSD papers and see how one operating system has decided to handle this

Hacking is magic

Hacking looks like magic – especially buffer overflows

Hacking is not magic

Hacking only demands ninja training and knowledge others don't have

It is like a puzzle, we need this, this and that. Make it happen in a repeatable way.

Nmap Workshop

- Portscanning with Nmap is a critical task for any pentest
- We will now switch to the materials found in my Nmap Workshop and perform some Nmap scans
  `https://github.com/kramse/security-courses/tree/master/courses/pentest/nmap-workshop`
- We will NOT do all the exercises
- But make sure you do the ones named on the following pages
- Nmap can output in XML and can be converted easily into HTML
- Nmap output can also be easily imported into a database, example Metasploit database
- Nmap, and other active tools procuce network traffic which can often be captured and analyzed, so we will also reference a few example capture tools

After we mention Metasploit - we can talk about features within this framework ☺

Exercise

Now lets do the exercise

Discover active systems ping sweep

which is number **5** in the exercise PDF.

Exercise

Now lets do the exercise

Execute nmap TCP and UDP port scan

which is number **6** in the exercise PDF.

Exercise

Now lets do the exercise

Perform nmap OS detection

which is number **7** in the exercise PDF.

Exercise

Now lets do the exercise

Perform nmap service scan

which is number **8** in the exercise PDF.

Exercise

Now lets do the exercise

Nmap full scan

which is number **9** in the exercise PDF.

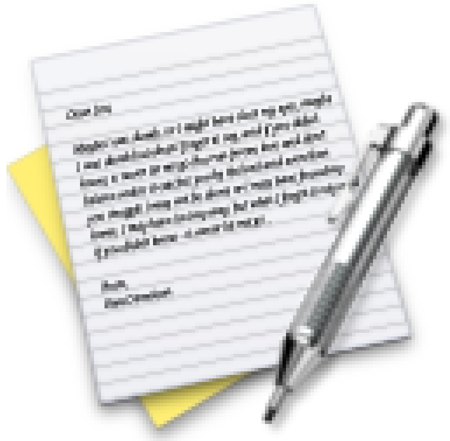
Exercise

Now lets do the exercise

Reporting HTML

which is number **10** in the exercise PDF.

Exercise

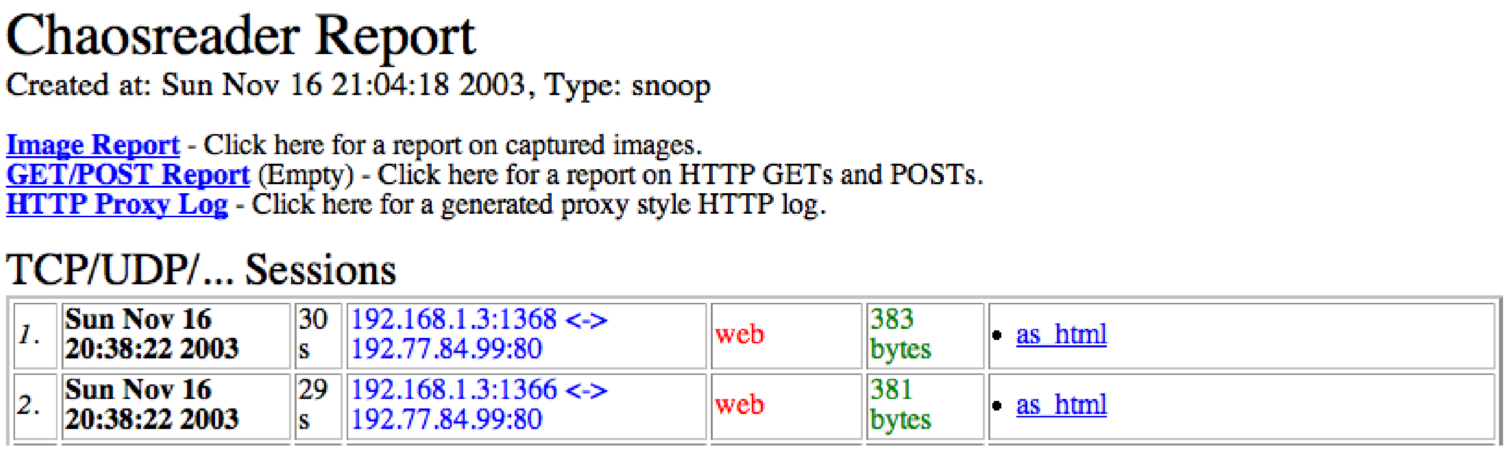


Now lets do the exercise

Nmap Scripting Engine NSE scripts

which is number **12** in the exercise PDF.

Chaosreader



Chaosreader Report

Created at: Sun Nov 16 21:04:18 2003, Type: snoop

Image Report - Click here for a report on captured images.
GET/POST Report (Empty) - Click here for a report on HTTP GETs and POSTs.
HTTP Proxy Log - Click here for a generated proxy style HTTP log.

TCP/UDP/... Sessions

1.	Sun Nov 16 20:38:22 2003	30 s	192.168.1.3:1368 <-> 192.77.84.99:80	web	383 bytes	• as_html
2.	Sun Nov 16 20:38:22 2003	29 s	192.168.1.3:1366 <-> 192.77.84.99:80	web	381 bytes	• as_html

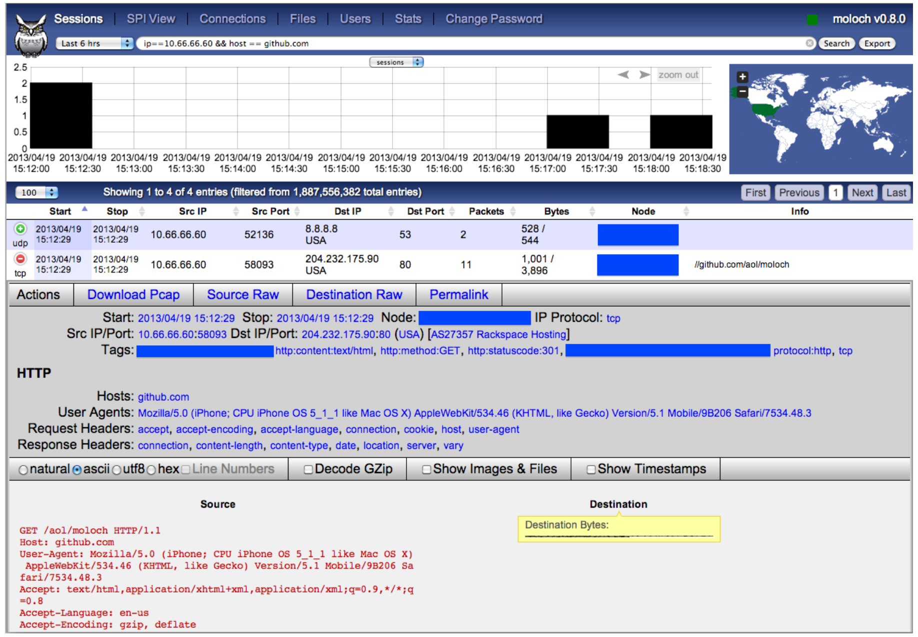
Simple but illustrative program

Read a pcap - packet capture into this tool chaosreader

Output HTML with nice index - usefull for quick demos

`http://chaosreader.sourceforge.net/`

Big data example Moloch



ip==10.66.66.60 && host == github.com

Showing 1 to 4 of 4 entries (filtered from 1,887,556,382 total entries)

	Start	Stop	Src IP	Src Port	Dst IP	Dst Port	Packets	Bytes	Node	Info
udp	2013/04/19 15:12:29	2013/04/19 15:12:29	10.66.66.60	52136	8.8.8.8 USA	53	2	528 / 544	[redacted]	
tcp	2013/04/19 15:12:29	2013/04/19 15:12:29	10.66.66.60	58093	204.232.175.90 USA	80	11	1,001 / 3,896	[redacted]	//github.com/aol/moloch

Start: 2013/04/19 15:12:29 Stop: 2013/04/19 15:12:29 Node: [redacted] IP Protocol: tcp
Src IP/Port: 10.66.66.60:58093 Dst IP/Port: 204.232.175.90:80 (USA) [AS27357 Rackspace Hosting]
Tags: [redacted] http:content:text/html, http:method:GET, http:statuscode:301, [redacted] protocol:http, tcp

HTTP

Hosts: github.com
User Agents: Mozilla/5.0 (iPhone; CPU iPhone OS 5_1_1 like Mac OS X) AppleWebKit/534.46 (KHTML, like Gecko) Version/5.1 Mobile/9B206 Safari/7534.48.3
Request Headers: accept, accept-encoding, accept-language, connection, cookie, host, user-agent
Response Headers: connection, content-length, content-type, date, location, server, vary

natural [✓ ascii] utf8 hex ☐ Line Numbers ☐ Decode GZip ☐ Show Images & Files ☐ Show Timestamps

Source

```
GET /aol/moloch HTTP/1.1
Host: github.com
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 5_1_1 like Mac OS X)
 AppleWebKit/534.46 (KHTML, like Gecko) Version/5.1 Mobile/9B206 Sa
fari/7534.48.3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q
=0.8
Accept-Language: en-us
Accept-Encoding: gzip, deflate
```

Destination

Destination Bytes:

Picture from `https://github.com/aol/moloch`
Be your own GCHQ ... capture all, index all, search all

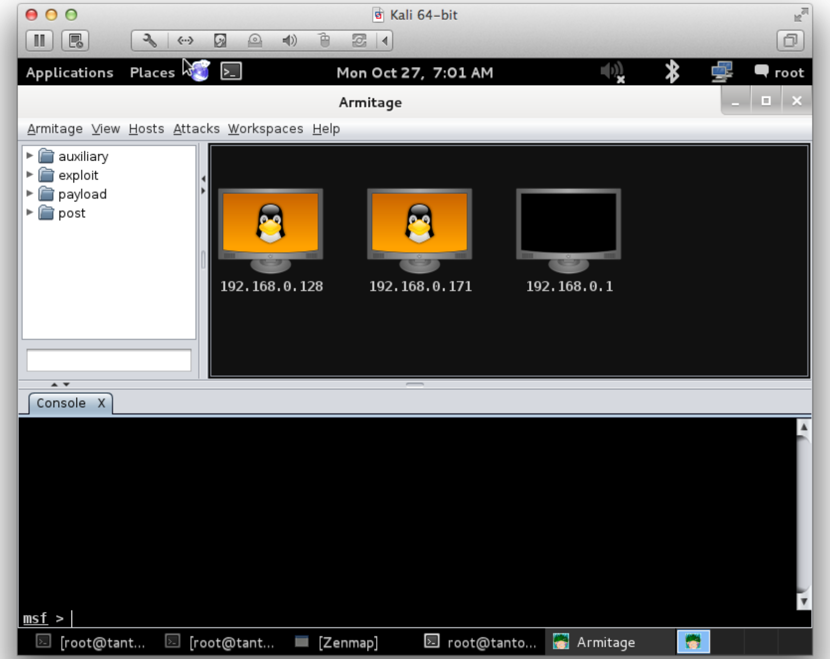
Exercise

Now lets do the exercise

Zeek on the web 10min

which is number **6** in the exercise PDF.

Demo: Metasploit Armitage



```
msf >
```

Exercise

Now lets do the exercise

Try Nmap from Metasploit

which is number **14** in the exercise PDF.

Exploit components

We will dive into the book: Grayhat chapters 12-14

We need to understand the parts of exploiting

Difference between the oldest, most simple stack based overflows

The parts of a shell code running system calls

How to avoid having shell code - return into libc, calling functions

This will teach us why modern operating systems have multiple methods designed to remove each case of exploiting

Allow us to understand the next subject, Return-Oriented Programming (ROP)

Exploit techniques – quick overview

Buffer Overflow Exploits – the quick overview:

Stack Based Buffer overflow – like a machine gun.

- Shooting lots of data into buffer, overflowing data structures
- Simple to execute
- Often the return address is POPed from the stack when returning from functions
- Goal: overwrite return address, control EIP Extended Instruction Pointer
- Control EIP – control program execution

Useful for generic exploiting of programs

Exploit techniques – quick overview

Format string – reading data like a microscope

- Abusing printf format strings to read some data
- Goal: read specific data

Useful to reveal something secret, a key, a stack canary

Exploit techniques – quick overview

Format string – writing data like a sniper

- Abusing printf format strings to **write** some data
- Goal: write specific data

Useful to overwrite something specific, when generic overflowing is not allowed/can be defended

Overwrite a stack canary or a return address

End goal: execution of code

Exploit techniques – quick overview

Exception Handlers – breaking apart from the program

- Abusing exception handling to execute code we control
- Goal: execute code

Example usage *Defeating the Stack Based Buffer Overflow Prevention Mechanism of Microsoft Windows 2003 Server* by David Litchfield (david@ngssoftware.com), 8 th September 2003

Exploit techniques – quick overview

Return to libc – execute function not code like importing a standard library

- Avoid non-executable memory protection, by putting parameters on the stack not code, and then calling a function
- Goal: execute shell code, without code

Useful when basic stack protection is in place

Came before Return-Oriented Programming (ROP), but has similarities

Return-Oriented Programming (ROP)

We will no look into Return-Oriented Programming (ROP) hopefully prepared by the chapters for today, and exercises

Return-Oriented Programming:Systems, Languages, and Applications Ryan Roemer, Erik Buchanan, Hovav Shacam and Stefan Savage University of California, San Diego

`https://hovav.net/ucsd/dist/rop.pdf`

Them we will look into how a security oriented operating system has decided to prevent this method:

Removing ROP Gadgets from OpenBSD Todd Mortimer

`https://www.openbsd.org/papers/asiabsdcon2019-rop-paper.pdf`

Setup the OWASP Juice Shop

If we have too much time, we will look into running the OWASP Juice Shop

This is an application which is modern AND designed to have security flaws.

Read more about this project at: `https://www2.owasp.org/www-project-juice-shop/` and `https://github.com/bkimminich/juice-shop`

It is recommended to buy the Pwning OWASP Juice Shop Official companion guide to the OWASP Juice Shop from https://leanpub.com/juice-shop - suggested price USD 5.99. Alternatively read online at https://pwning.owasp-juice.shop/

Sometimes the best method is running the Docker version

Later we will start hacking this awesome application

Exercise

Now lets do the exercise

Run OWASP Juice Shop 45 min

which is number **19** in the exercise PDF.

Exercise

Now lets do the exercise

Setup JuiceShop environment, app and proxy - up to 60min

which is number **20** in the exercise PDF.

For Next Time

Think about the subjects from this time, write down questions

Check the plan for chapters to read in the books

Visit web sites and download papers if needed

Retry the exercises to get more confident using the tools