Global Information Assurance Certification Paper

"Unicode Vulnerability" – How & Why?

By Andrew Brannan

Assignment Version 1.2e

August 7, 2001

"Unicode" Vulnerabilty – How & Why?

What is the Unicode Vulnerability?

This vulnerability began its life as the MS00-057 "File Permission Canonicalization" vulnerability, a vulnerability which could allow a malicious user to gain higher levels of access to certain types of files than would normally be given. However, in early October 2000, an anonymous user on the Packetstorm forum posted details as to how the vulnerability detailed in the MS00-057 bulletin could be used to run a dir command on the target server. This "Web Server Folder Traversal", or "Unicode" vulnerability was identified by Microsoft in Microsoft Security Bulletin MS00-078. This bulletin revealed that a malicious user could "access files and folders that lie anywhere on the logical drive that contains the web folders." and "cause widespread damage" to the affected system. What this means is that an attacker, by exploiting this vulnerability, has execute privileges on any file that resides on the same logical drive as the web root folder. Since IIS by default installs the inetpub folder on the c:\ drive along with the winnt folder, this means an attacker can use cmd.exe, TFTP.exe, attrib.exe or any other executable that can be accessed from the command line.

This vulnerability gained the nickname "Unicode" due to the Unicode encoded characters that are used to exploit the vulnerability. However it is not actual Unicode characters that are used to exploit this vulnerability. It is the UCS Transformation Format (UTF) characters, used to bridge between Unicode and ASCII character sets, that are actually used to exploit this vulnerability. However, for consistency's sake we will continue to refer to the encoded characters as "Unicode" throughout this paper.

A more detailed look at the history of this vulnerability and the UTF and Unicode character sets can be found at: http://www.sans.org/infosecFAQ/threats/traversal.htm

How Does the Vulnerability Work?

In normal file path representation the characters '../' or '..\'can be used to denote a parent directory of the current working directory. So, for example, if I am in c:\inetpub\scripts and I want to access c:\inetpub\pictures, I can call '../pictures' from c:\inetpub\scripts. In this case the whole path would become c:\inetpub\scripts\..\pictures.

Microsoft's IIS performs a security check on every requested CGI URL to make sure that the request does not use any "../" directory traversals to traverse outside the normal *inetpub* web folder. If this check did not exist, any user on the internet would be able to access any file on the drive. So, a request to http://www.myserver.com/scripts/../../winnt/system32/cmd.exe would be rejected by this security check and the request would fail.

Here is where Unicode comes into play. After IIS performs the security check, it then passes the URL through a decode routine to decode any extended Unicode characters, then fulfils the request. This decode routine exists after the security check to decode any special characters in the parameters passed to a CGI script. If the "/" character is encoded in Unicode as "%c0%af", the URL will pass the security check, as it

does not contain any "../" patterns. Instead the security check only sees "..%c0%af", which it does not recognize as a malicious pattern. However, once the decode routine is done with the URL, the innocuous "..%c0%af" is now the malicious "../". So, while a request to http://www.myserver.com/scripts/../../winnt/system32/cmd.exe would fail, a request to http://www.myserver.com/scripts/..%c0%af../winnt/system32/cmd.exe would succeed. SO by using encoded characters to fool the security check, a malicious user can gain access to any file on the logical drive that the web folders are hosted on.

What's the big deal?

The Unicode decoding operation by itself is not the worst part of the problem. It is still the IUSR_<machinename> account, the anonymous account used by IIS to access the system, that is making the request. The problem lies in the fact that the IUSR account is a member of the Everyone and Users groups, which have execute privileges on most system files, including cmd.exe and command.com. This gives the remote malicious user the ability to arbitrarily execute commands on your web server including, but not limited to, deletion of files, and opening new network connections (TFTP, etc.). Any command or series of commands that can be issued at a command prompt or DOS prompt can now be issued through a series of http requests to the server. And since the requests are using port 80 to connect to the server, a firewall cannot be used to filter this traffic out from other, legitimate web traffic to the server.

This ability has been used as the basis for several different "script kiddie" attacks on IIS Web servers, ranging from a perl script that provides a remote shell to the attacker to scripts that set up "root kits" on the server for use as staging points for further network attacks. The sheer range of uses of the vulnerability combined with the ease with which attacks can be constructed quickly made the vulnerability very popular with hackers everywhere.

My system was hacked, was it Unicode?

Fortunately, many of the "script kiddie" tools currently in use don't clean up after themselves, and plenty of evidence of the attack is left in the IIS Web logs. Also, many of these tools attempt several different URLs to determine the correct path to use to exploit this vulnerability. The most common way to determine if a server is vulnerable to the Unicode Folder traversal attack is to issue a dir command on the c:\ drive. To do this the attacker will send a URL like the following:
http://www.myserver.com/scripts/..%c0%af../winnt/system32/cmd.exe?c+dir+c:\

If successful, the web server will respond with a directory listing of the root of c:\. Once a system has been identified as vulnerable, most attacks will make a copy of cmd.exe in the scripts directory. These copies of cmd.exe are usually named something different to help escape detection, such as cmd1.exe, oct3.exe, sensepost.exe or root.exe. These copies allow the attacker to use redirection to echo commands into .BAT files for later execution. For example:

cmd.exe?/c+echo+you+got+h4x0r3d+>+index.html

Assuming these two commands are common to most attacks you should see something similar to the following in your IIS web logs.

```
2001-08-04 16:16:33 10.254.0.1 - 10.254.0.15 80 GET
/scripts/../../winnt/system32/cmd.exe /c+dir 200 -
2001-08-04 16:16:33 10.254.0.1 - 10.254.0.15 80 GET
/scripts..\../winnt/system32/cmd.exe /c+dir 404 -
2001-08-04 16:16:33 10.254.0.1 - 10.254.0.15 80 GET
/scripts/..Á%pc../winnt/system32/cmd.exe /c+dir 500 -
2001-08-04 16:16:33 10.254.0.1 - 10.254.0.15 80 GET
/scripts/..À%9v../winnt/system32/cmd.exe /c+dir 500 -
2001-08-04 16:16:33 10.254.0.1 - 10.254.0.15 80 GET
/scripts/..À%qf../winnt/system32/cmd.exe /c+dir 500 -
2001-08-04 16:16:33 10.254.0.1 - 10.254.0.15 80 GET
/scripts/..Á%8s../winnt/system32/cmd.exe /c+dir 500 -
2001-08-04 16:16:33 10.254.0.1 - 10.254.0.15 80 GET
/scripts/..Á_../winnt/system32/cmd.exe /c+dir 500 -
2001-08-04 16:16:33 10.254.0.1 - 10.254.0.15 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+dir 200 -
2001-08-04 16:16:33 10.254.0.1 - 10.254.0.15 80 GET
/scripts/..o../winnt/system32/cmd.exe /c+dir 404 -
2001-08-04 16:16:33 10.254.0.1 - 10.254.0.15 80 GET
/scripts/../../winnt/system32/cmd.exe /c+dir 200 -
2001-08-04 16:16:33 10.254.0.1 - 10.254.0.15 80 GET
/scripts/..ðÇÇ¯../winnt/system32/cmd.exe /c+dir 404 -
```

Astute readers have probably notices that not all of these URLs were successful. Some of the requests resulted in '404' errors. These requests are from other alternate ways of encoding the '/' or '\' characters, which may work on other versions of IIS or Windows. Many attack scripts will use 20 or so different encodings to ensure that all vulnerable systems will be identified.

Countermeasures

The first, most obvious step to take in securing your IIS server is to apply the Microsoft recommended patch. MS00-078 recommends applying the MS00-057 patch to fix this vulnerability. The MS00-057 patch was first issued on August 10, 2000. This patch will be included in Service Pack 7 for Windows NT and Service Pack 3 for Windows 2000, but as of the time of writing these Service Packs are not yet available.

Installing the security patches is not enough, as we will see later in this paper. To further secure your Web Server, you should specifically deny IUSR any privileges on cmd.exe, command.com, tftp.exe and attrib.exe. By denying IUSR rights to these files you create a condition where, even if a Unicode attack is launched against your server, the attacker (running as IUSR) cannot access the necessary commands to exploit your server. In MS00-078, Microsoft also recommends removing the Everyone and Users groups from permissions on the server.

To deny IUSR rights to cmd.exe and other files, right click on the file and select "Properties". Select the Security Tab, and then select "Permissions". Add a rule and select the IUSR_MACHINENAME account and select "deny all" as the access level. Repeat this process for any file you suspect could be used by an attacker, including tftp.exe, attrib.exe, and command.com, to name a few prime targets.

If possible, install your web folders on a different logical drive from your \WINNT folder(e.g. d:\inetpub). If the web folders are installed on a different logical drive,

no amount of folder traversal will allow the attacker access to any commands they might use to exploit your system, as they cannot reach the winnt system folder where the commands they seek to access are stored. To change the location of your web folders, select the Web Service from the Internet Information Service Manager, and change the home directory to a folder located on a different logical drive. D:\inetpub should be sufficient. You can then copy any existing web site material into this new directory and run your web site from there.

These two simple steps will defeat most of the "script kiddie" Unicode attacks today. When used in combination with other good security practices such as applying recommended security patches, and regular review of the Security, Event, and IIS web logs, they create a condition where it is very difficult to use the Unicode exploit to attack your server. As always, if you are not using the Web Service on your Windows System, stop and disable the service, in the services Control Panel or uninstall IIS if it is not needed on the system.

If you have an IDS system, such as Shadow or Snort, you can create a rule that logs any URL call to cmd.exe to alert you of attempted attacks. You should also monitor your IIS Web Logs for any call to cmd.exe. It is important to note that there rules will, in most cases, only catch the initial scan for vulnerabilities. Once cmd.exe is copied into the scripts directory under a new name, any future requests from the same attacker will probably use the new name and will not be caught by the IDS system.

MS01-026 – "Unicode" revisited

In April 2001, Microsoft released yet another security bulletin related to the Unicode vulnerability. Bulletin MS01-026 revealed that the original fix, running a URL security check after the Unicode decode operation, was defeated by the presence of another decoding operation after the security check. So, if you encoded your slash character, and then encoded the encoded version of the slash you could fool the security check with what looks to be harmless characters.

From the NSFOCUS Security Advisory SA2001-02:

> For example, a character '\' will be encoded to "%5c". And the corresponding code of these 3 characters is:
> '%' = %25
> '5' = %35
> 'c' = %63
>
> encode this 3 characters for another time (sic), we can get many results such as:
> %255c
> %%35c
> %%35%63
> %25%35%63
> …

> Thereby, '..\' can be represented by '..%255c' and '..%%35c', etc. After first decoding '..%255c' is turned into '..%5c'. IIS will take it as a legal character string that can pass security check-up. But after a second decode process, it will be reverted to '..\'. Hence, attacker can use '..\' to carry out directory traversal and run arbitrary program outside of Web directory.

Thus, the "Unicode" bug appears in yet another form. By doubly encoding the '/' or '\' characters, an attacker can yet again foil the CGI URL security check, as the security check will only see a seemingly harmless '%5c' when it checks the URL. However, the presence of an additional decoding operation after the security check then transforms the '%5c' into '\'.

This variant attack isn't limited to just decoding the '\' and '/' characters. You could also encode the '.' character. A first encoding would encode it as '%2e', and you could then encode any one or more of those characters.

Server admins who had only applied the patches for the previous vulnerabilities were once again vulnerable. However, those to took additional precautions, changing permissions on cmd.exe and the like, and those who did not have \inetpub installed on the c:\ drive were not vulnerable to this new attack.

This Advisory/vulnerability is a wonderful example of how important it is to have a broad security policy when it comes to implementing and maintaining your servers. A simple, narrow security policy of "We'll just apply all security patches for the server" leaves you open to attack in these kind of scenarios. If you had implemented another simple security measure, not running your web folders on the same logical drive as your /winnt directory, you actually would not have been vulnerable to either one of the security vulnerabilities discussed in this paper.

Unicode_shell.pl – A Practical Example

To highlight the ease with which this vulnerability can be exploited, and the flexibility the vulnerability offers, we will take a look at one of the exploit scripts available on many security sites. Unicode_Shell.pl, by B-r00t, is a simple perl script that can scan an IIS webserver, determine a specific URL to send to gain access to cmd.exe, and provide the attacker with a virtual command line interface on the target system. The script even provides the functionality to change the URL to use for the attack, as well as scanning several popular Unicode URL strings on the server to account for different scenarios. These URL strings have the forward-slash and backslash characters encoded in Unicode in a variety of ways including:

%c0%af	%c1%9c	%c1%pc	%c0%qf
%c1%8s	%c1%1c	%c1%af	%e0%80%af

As you can see, there are many ways to encode the slash characters in Unicode. This script, which was written before Microsoft bulletin MS01-026, does not include the alternate encoding sequences for that vulnerability, though this functionality is easily

added.

Once a target host is chosen, and an attack URL has been selected, the script creates a copy of cmd.exe in the scripts directory to allow the full functionality of cmd.exe to be utilized (e.g. piping output to text files). At this point the remote attacker has the same abilities as a user logged onto the system has in a cmd.exe window. The attacker can create batch files, read or delete files, modify system settings. An attacker could also use tftp to transfer Trojan programs like Netcat, SubSeven or Back Orifice to the system and use the system as a launching pad for attacks on other systems.

Unicode_shell.pl is just one of many attack scripts available to attackers. Other attack scripts serve a single purpose, such as uploading and executing Netcat on the target system. Another script uses firedaemon and SUD to create a "root kit" for Windows systems. The power and flexibility of this vulnerability mean that the types of attack are only limited by the attacker's imagination.

Below is a sample session from unicode_shell.pl. :

```
 A Unicode HTTP exploit for Microsoft NT IIS WebServers.

 First tries to get IIS Server string.
 Scans for usable Unicode URL in 20 different ways.
 Then allows choice of which URL to use including an URL
of
 your own design eg. After copying cmd.exe to /scripts.
 Commands are executed via your choice of URL on the
target
 server.

 URL can be changed at anytime by typing URL.
 The Webserver can be re-SCANed at anytime by typing
SCAN.
 Program can be QUIT at anytime by typing QUIT.
 HELP prints this ...
 ENJOY !

Host :10.254.0.15

Port :80

Trying to obtain IIS Server string ...
OUTPUT FROM 10.254.0.15.

Server: Microsoft-IIS/5.0
Date: Sat, 04 Aug 2001 16:25:08 GMT
Connection: Keep-Alive
Content-Length: 1270
Content-Type: text/html
Set-Cookie:
ASPSESSIONIDQQGGGSCK=OMFJOPACOBGNABPEOFMDBGCP; path=/
Cache-control: private

OK ... It Seems To Be Microsoft IIS.


Scanning Webserver 10.254.0.15 on port 80 ...
10.254.0.15 IS VULNERABLE TO UNICODE URL NUMBER 1 !!!
```

```
10.254.0.15 is not vulnerable to Unicode URL Number 2.
10.254.0.15 is not vulnerable to Unicode URL Number 3.
10.254.0.15 is not vulnerable to Unicode URL Number 4.
10.254.0.15 is not vulnerable to Unicode URL Number 5.
10.254.0.15 is not vulnerable to Unicode URL Number 6.
10.254.0.15 is not vulnerable to Unicode URL Number 7.
10.254.0.15 IS VULNERABLE TO UNICODE URL NUMBER 8 !!!
10.254.0.15 is not vulnerable to Unicode URL Number 9.
10.254.0.15 IS VULNERABLE TO UNICODE URL NUMBER 10 !!!
10.254.0.15 is not vulnerable to Unicode URL Number 11.
10.254.0.15 is not vulnerable to Unicode URL Number 12.
10.254.0.15 is not vulnerable to Unicode URL Number 13.
10.254.0.15 is not vulnerable to Unicode URL Number 14.
10.254.0.15 is not vulnerable to Unicode URL Number 15.
10.254.0.15 is not vulnerable to Unicode URL Number 16.
10.254.0.15 is not vulnerable to Unicode URL Number 17.
10.254.0.15 is not vulnerable to Unicode URL Number 18.
10.254.0.15 IS VULNERABLE TO UNICODE URL NUMBER 19 !!!
10.254.0.15 is not vulnerable to Unicode URL Number 20.

URL To Use [0 = Other]: 1

URL:
HTTP://10.254.0.15/scripts/..%c0%af../winnt/system32/cmd.
exe?/c+
HELP QUIT URL SCAN Or Command eg dir C:
Command :dir c:
HTTP://10.254.0.15/scripts/..%c0%af../winnt/system32/cmd.
exe?/c+dir+c:
OUTPUT FROM 10.254.0.15.

Server: Microsoft-IIS/5.0
Date: Sat, 04 Aug 2001 16:17:19 GMT
Content-Type: application/octet-stream
Volume in drive C has no label.
Volume Serial Number is 5446-9E02

 Directory of C:\Inetpub\scripts

05/07/2001  11:05a      <DIR>          .
05/07/2001  11:05a      <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)   2,520,857,088 bytes free

HELP QUIT URL SCAN Or Command eg dir C:
```

Although this was just a small taste of what an attacker can do with this vulnerability, it does show just how powerful this vulnerability can be. An attacker who uses this vulnerability has your system completely under his or her control.

Conclusions

The power and flexibility of the Unicode vulnerability make it one of the most popular, and therefore dangerous, vulnerabilities currently used by attackers today. Attackers can very easily create new and varied attacks using this vulnerability, and the

vulnerability is easy enough to exploit that talented attackers can attack your server "on the fly", adjusting their commands issued to the server to adjust for your particular environment.

However, this vulnerability can be easily defeated if a careful system administrator takes a few simple steps, such as moving the web folder root off of the logical drive that holds the system executables. But there are too many systems on the internet whose administrators take a less proactive approach to security and only apply security patches on their system rather than design security into the system from the start. Worse yet are the systems administrators who only apply the Microsoft Service Packs to their systems and take no other action to secure their servers. The patches for this vulnerability (MS00-057 and MS01-026) have not yet been included in any of Microsoft's Service Packs for either NT or Windows 2000 systems, which means that several poorly administered systems are still vulnerable almost a year after the vulnerability was announced.

References

Microsoft Security Bulletin (MS00-057)
Title: Patch Available for "File Permission Canonicalization" Vulnerability
Originally Posted: August 10, 2000
URL: http://www.microsoft.com/technet/security/bulletin/MS00-057.asp

Microsoft Security Bulletin (MS00-078)
Title: Patch Available for "Web Server Folder Traversal" Vulnerability
Originally Posted: October 17, 2000
URL: http://www.microsoft.com/technet/security/bulletin/MS00-078.asp

Microsoft Security Bulletin (MS01-026)
Title: Superfluous Decoding Operation Could Allow Command Execution via IIS
Originally Posted: August 10, 2000
URL: http://www.microsoft.com/technet/security/bulletin/MS01-026.asp

Author: Ron Grove
Title: Details on a hacked NT server (possible kit?)
Originally Posted: February 25, 2001
URL: http://www.securityfocus.com/templates/archive.pike?list=88&mid=165327

Author: Matt Scarborough
Title: Re: Some details in a recent NT hack we encountered
Originally Posted: February 25, 2001
URL: http://www.securityfocus.com/templates/archive.pike?list=75&mid=165451

Author: Steven Shields
Title: "Web Server Folder Traversal" vulnerability (MS00-078)
Originally Posted: February 13, 2001
URL: http://www.sans.org/infosecFAQ/threats/traversal.htm

Author: Ben Wilson
Title: Life Cycle of a Vulnerability using the CGI Script Vulnerability in Microsoft IIS (Internet Information Server, Microsoft's Web Server) as an example
Originally Posted: March 14, 2001
URL: http://www.sans.org/infosecFAQ/win/life_cycle.htm

Author: B-r00t
Title: Unicode_shell.pl
URL: http://packetstormsecurity.org/0101-exploits/unicode_shell.pl

Author: NSFocus Security Team
Title: Microsoft IIS CGI Filename Decode Error Vulnerability
Originally Posted: May 15, 2001
URL: http://www.nsfocus.com/english/homepage/sa01-02.htm

© SANS Institute 2000 - 2005 Author retains full rights.