Determine Your System's Status

We last updated this page on: Nov 06, 2018 07:50

Overview

cPanel's Support department detected the following security issues:

- Compromised RPMs in the OpenSSH binaries.
- Compromised `libkeyutils` directories.
- Root-level compromises.

In these cases, Trojan horses (Trojans) affected files that these directories and binaries contain. We **strongly** recommend that hosting providers and system administrators use this document to determine the status of their systems.

> **Note:**
> The affected OpenSSH Binaries and `libkeyutils` directories produce similar network traffic.

If you experience any issues while you perform these checks or you suspect that someone has compromised your server, contact your hosting provider for assistance.

> **Notes:**
> - To check your system for compromises unrelated to the OpenSSH Binaries and `libkeyutils` directories, use WHM's *Security Advisor* interface (*WHM >> Home >> Security Center >> Security Advisor*).
> - We **strongly** recommend that you read our Tips to Make Your Server More Secure documentation.

Compromised systems

Click the tab below that corresponds to the item you wish to check.
OpenSSH RPMsThe libkeysutils Utilitiesroot-level Compromises

On CentOS and Red Hat® Enterprise Linux® systems, we determined that the following OpenSSH binaries contain the Ebury Trojan:

- The `sshd` binary.
- The `ssh` binary.
- The `ssh-keygen` binary.
- The `ssh-askpass` binary.

This Trojan's code collects authentication credentials for inbound and outbound network connections, as well as the SSH keys that these binaries generate.

Use the following checks to determine whether the Trojan compromised your system's OpenSSH binaries.

Check for malicious processes

To check for malicious processes on your server, run the following command:

```
netstat -plan | grep atd
```

This command does not return output on non-compromised systems. On compromised systems, this command returns output that resembles the following example:

```
unix 2 [ ACC ] STREAM LISTENING 103713 8119/atd @/tmp/dbus-ZP7tFO4xsL
```

Check the RPMs' change logs.

To check an RPM version's change log, run the following command:

> In this example, `openssh-5.3p1-209.el6` represents the OpenSSH RPM binary that you wish to check.

Non-compromised RPMs contain a signature

```
rpm -q --changelog openssh-5.3p1-209.el6 | head -10
```

A compromised RPM's change log contains **no** entries.

Check your system's `yum` logs.

Open your system's `/var/log/yum.log` file with a text editor and search for the OpenSSH binaries. The file's contents resemble the following example:

```
Jun 22 10:33:00 Installed: geronimo-jms-1.1.1-19.el7.noarch
Jun 22 10:33:00 Installed: xml-commons-apis-1.4.01-16.el7.noarch
Jun 22 10:33:00 Installed: xml-commons-resolver-1.2-15.el7.noarch
Jun 22 10:33:00 Installed: xalan-j2-2.7.1-23.el7.noarch
Jun 22 10:33:00 Installed: xerces-j2-2.11.0-17.el7_0.noarch
Jun 22 10:33:01 Installed: avalon-framework-4.3-10.el7.noarch
Jun 22 10:33:01 Installed: tomcat-el-2.2-api-7.0.69-11.el7_3.noarch
Jun 22 10:33:01 Installed: tomcat-lib-7.0.69-11.el7_3.noarch
Jun 22 10:33:02 Installed: tomcat-7.0.69-11.el7_3.noarch
Jun 22 10:33:02 Installed: openssh-6.6.1p1-35.el7_3.x86_64
Jun 22 10:33:02 Installed: libssh2-1.4.3-10.el7_2.1.x86_64
Jun 22 10:33:02 Installed: openssh-server-6.6.1p1-35.el7_3.x86_64
Jun 22 10:33:02 Installed: cpanel-perl-524-Net-OpenSSH-0.74-1.cp1162.noarch
Jun 22 10:33:02 Installed: openssh-clients-6.6.1p1-35.el7_3.x86_64
```

If your system's `/var/log/yum.log` file does not contain the OpenSSH RPMs, the server may exist in a compromised state.
To check your `libkeysutils` utility for the Ebury Trojan, perform the following steps:

Important:

- We recommend that you perform each of the following steps to fully determine your server's status.
- Your server's architecture determines the output that you receive when you run the commands in this section.
  - 32-bit systems use `lib`. 64-bit systems use the `/lib64` library. The `/lib64` library may also contain a `lib` directory with 32-bit libraries for compatibility purposes. To find your server's architecture, run the `arch` command:

```
[user@host ~]$ arch
```

The output will resemble one of the following examples:

64-bit server

```
x86_64
```

32-bit server

```
i386
```

1

Verify the keyutils-libs package.

Changes should **not** occur on your system without your consent. To check for changes on your system, run either of the following commands:

```
rpm -V keyutils-libs
netstat -plan | grep atd
```

This command does not return output on non-compromised systems. On compromised systems, this command returns the following output:

```
....L...    /lib64/libkeyutils.so.1
```

> **Note:**
> Ebury control system operators often patch the MD5 sums of the `libkeyutils1` package to match the old packages, so an MD5 check may not reveal compromised packages.

Check the files' installation source.

If the command in Step 1 returns output, verify whether an RPM provided the files. To do this, run the following command:

```
rpm -qf /lib64/libkeyutils.so.1
```

> **Note:**
> In this example, `/lib64/libkeyutils.so.1` represents the file to verify.

On non-compromised systems, the output resembles the following example:

```
keyutils-libs-1.4-5.el6.i686
```

On compromised systems, the output resembles the following example:

```
file /lib64/tls/libkeyutils.so.1.5 is not owned by any package
```

Verify the file that links to the `libkeyutils.so.1` file.

To see which file links to the `libkeyutils.so.1` file, run the following command:

```
ls -l /lib64/libkeyutils.so.1
```

On non-compromised systems, the output will resemble the following example:

```
lrwxrwxrwx 1 root root 18 Feb 20 12:15 /lib64/libkeyutils.so.1 ->
libkeyutils.so.1.3*
```

On compromised systems, the output will return one or more of the following files:

```
libkeyutils.so.1.9
libkeyutils.so.1.3.2
libkeyutils-1.2.so.2
```

Check the strings for all `libkeyutils.*` libraries for items that relate to networking.

The default `libkeyutils.so.1.3` file should **not** contain the following strings:

```
connect
socket
inet_ntoa
gethostbyname
```

To check your system for these strings, run the following command:

```
strings /lib64/libkeyutils.so.1.3 | egrep
'connect|socket|inet_ntoa|gethostbyname'
```

This command does not return output on non-compromised systems. On compromised systems, this command returns the following output:

```
connect
socket
inet_ntoa
gethostbyname
```

Check `sshd` processes.

On compromised servers, `sshd` processes use shared memory segments.

To check whether any `sshd` processes currently use shared memory segments, run the following command:

```
ipcs -mp
```

On a non-compromised server, the output will resemble the following example:

```
------ Shared Memory Creator/Last-op --------
shmid      owner      cpid       lpid
```

On a compromised server, the output will resemble the following example:

```
------ Shared Memory Creator/Last-op --------
shmid      owner      cpid       lpid
1769472    root       1975       1975
2129921    root       2931       2940
1736706    root       1965       1965
2162691    root       2931       2940
2195460    root       2931       2940
2228229    postgres   4011       6813
```

If any programs use shared memory segments, run the `ps` command, for example:

```
ps aux | grep 1975
```

This command checks whether any of the items in the `cpid` and `lpid` columns correspond to the `sshd` processes. The output will resemble the following example:

```
root     1975  0.0  0.0  64080  1172 ?        Ss   Feb17
0:00 /usr/sbin/sshd
```

Monitor outbound UDP port 53 traffic.

We recommend that you use a network data capture tool, such as the `tcpdump` utility, to monitor outbound User Datagram Protocol (UDP) on port 53. This utility returns DNS traffic between your server and the local resolvers that reside in your `/etc/resolv.conf` file.

To monitor outbound UDP on port 53, run the following command:

```
[root@host ~]# tcpdump -Annvvs 1500 -i any udp and dst port 53
```

On a non-compromised server, the output will resemble the following example:

```
22:18:22.038264 ARP, Request who-has 10.147.0.62 tell 10.147.0.64, length
46
22:18:22.244856 ARP, Request who-has 10.147.0.64 tell 10.147.0.61, length
46
22:18:22.245149 ARP, Request who-has 10.147.0.61 tell 10.147.0.64, length
46
22:18:22.888851 b4:99:ba:02:18:66 > Broadcast, ethertype Unknown
(0xcafe), length 90:
        0x0000:  0500 0100 0900 0000 0100 ffff 0c00 0002  ..... ..........
        0x0010:  4c00 0000 0000 0000 8300 0080 0000 0000  L...............
```

On a compromised server, the output will resemble the following example:

```
07:54:48.233159 IP (tos 0x0, ttl  64, id 31281, offset 0, flags [DF],
proto: UDP (17), length: 91) 1.2.3.4.43089 > 72.156.139.154.53:
[bad udp cksum d7a9!]  4619+ A?
6196g8f43a4facd3561de4gec736fb.5.5.5.5. (63)
```

Important:
The example of the packet above shows that the cPanel server at `1.2.3.4` sends a UDP packet on **port 53** to the host at `72.156.139.154`. You should notice the following false query from the example above:

```
6196g8f43a4facd3561de4gec736fb.5.5.5.5
```

The string `6196g8f43a4facd3561de4gec736fb` represents an encoded password, and `5.5.5.5` represents the IP address that you used to log in to the server. This output indicates that the malicious host at `72.156.139.154` captured the following information:

- The IP address that you used to log in to the server.
- The login password.

Verify that each library that sshd links against belongs to a known package.

To check the libraries that the `sshd` daemon links against, run the following command

```
ldd /usr/sbin/sshd
```

The output will resemble the following example:

```
libnspr4.so => /lib64/libnspr4.so (0x00007fcb04a30000)
 libfreebl3.so => /lib64/libfreebl3.so (0x00007fcb0482d000)
 libkrb5support.so.0 => /lib64/libkrb5support.so.0 (0x00007fcb0461d000)
 libkeyutils.so.1 => /lib64/libkeyutils.so.1 (0x00007fcb04419000)
 libattr.so.1 => /lib64/libattr.so.1 (0x00007fcb04213000)
 libelf.so.1 => /lib64/libelf.so.1 (0x00007fcb03ffb000)
 libbz2.so.1 => /lib64/libbz2.so.1 (0x00007fcb03dea000)
```

To verify that these libraries belong to a valid package, run the following command:

```
rpm -qf /lib64/libfipscheck.so.1
```

The output will resemble the following example:

```
fipscheck-lib-1.2.0-7.el6.x86_64
```

Perform an LD_DEBUG check

To check the libraries with the LD_DEBUG program, which searches for Indicators of Compromise (IOC), run the following command

```
LD_DEBUG=symbols /bin/true 2>&1 | egrep
'/lib(keyutils|ns[25]|pw[35]|s[bl]r)\.'
```

The program will return any indicators of a compromise.

Perform an objdump check

The objdump command reports information about object files. You can use this command to determine if a needed file displays suspicious characteristics or behavior

To check the libraries with the LD_DEBUG program, run the following command, where `/path/to/libkeyutils.so.1` represents the full path to the suspected file:

```
objdump -x /path/to/libkeyutils.so.1 | grep NEEDED | grep -v -F -e libdl.so
-e libc.so
```

The program will return the phrase "`NEEDED`" if any indicators of a compromise exist.
A `root` compromise represents a security breach that occurrs at the `root` level. This exposes the entire server to theft and damage.

If cPanel Support detects that your server contains a root-level compromise, we will let you know. However, please note that the cPanel Support Team are not security consultants and we provide minimal support for servers with root-level compromises. At most, we will try to determine if attackers have used our software (cPanel & WHM) to gain unauthorized root access to your server.

So how does cPanel Support determine if your server contains a root-level compromise? Compromises vary in complexity and form, but we can easily detect certain rootkits.

Check for suspicious library files

For example, the `/lib64/libpw5.so` file is part of the Ebury Rootkit, and clean systems do not usually contain this file. That file may use other names which the We Live Security analysis of the Ebury Rootkit document lists.

Llibrary files normally use less than 10Kb. Any library file that uses over 10Kb strongly indicates a compromise.

To check for the existence of the `/lib64/libpw5.so` library file, run the following command:

```
ls /lib64/libpw5.so
```

This command returns the following output when the file does not exist:

```
/bin/ls: cannot access /lib64/libpw5.so: No such file or directory
```

On compromised systems, this command returns output that resembles the following example:

```
/lib64/libpw5.so
```

Additionally, the system uses yum or rpm to install most library files. If we query the RPM database for the `/lib64/libpw5.so` file and it returns a "`Not Owned By Any Package`" error, then the server may be compromised.

> **Important:**
> New variants of the Rootkit may falsify RPM package management. We recommend that you verify the file's hash with the instructions in the following section.

```
# rpm -qf /lib64/libpw5.so
file /lib64/libpw5.so is not owned by any package
```

Check the file hash.

Run the `sha256` hash of the file through the VirusTotal website to check it against pre-existing scans of the same file.

For example, to generate the sha256 hash for the `/lib64/libpw5.so` file, run the following command:

```
# sha256sum /lib64/libpw5.so
sha256sum: 970b49c16eebd558ac8984643f3763e76a52c9be4118f9e5830b8f5c406414fc
```

Then, navigate to the `https://www.virustotal.com/en/file/hash` website, where `hash` represents the hash of the file. In this example, you would navigate to the following website:

```
https://www.virustotal.com/en/file/970b49c16eebd558ac8984643f3763e76a52c9b
e4118f9e5830b8f5c406414fc/analysis/
```

The website should return similar results to the following output:

```
SHA256: 970b49c16eebd558ac8984643f3763e76a52c9be4118f9e5830b8f5c406414fc
File name:  libpw5.so
Detection ratio:    3 / 58
```

These results show that at least 3 out of 5 antivirus systems detect this file as a compromise.

> **Note:**
> The VirusTotal website does **not** contain results from every potentially compromised file. We **strongly** recommend that you consult a qualified security specialist.

Check for intertwined binaries

Check binaries on your server to detect whether they intertwine with the suspicious file.

For example, the following command detects whether binaries intertwine with the `/lib64/libpw5.so` file.

> **Note:**
> `# lsof | grep /lib64/libpw5.so`

The command will produce results similar to the following output:

```
auditd     1577               root  mem       REG                8,3
34536   16257536 /lib64/libpw5.so
sshd       2126               root  mem       REG                8,3
34536   16257536 /lib64/libpw5.so
exim       2622           mailnull  mem       REG                8,3
34536   16257536 /lib64/libpw5.so
pure-ftpd  2821               root  mem       REG                8,3
34536   16257536 /lib64/libpw5.so
pure-auth  2823               root  mem       REG                8,3
34536   16257536 /lib64/libpw5.so
cpsrvd     2887               root  mem       REG                8,3
34536   16257536 /lib64/libpw5.so
httpd      3063             nobody  mem       REG                8,3
34536   16257536 /lib64/libpw5.so
pop3-logi  3093           dovenull  mem       REG                8,3
34536   16257536 /lib64/libpw5.so
imap-logi  3094           dovenull  mem       REG                8,3
34536   16257536 /lib64/libpw5.so
lmtp       3120               root  mem       REG                8,3
34536   16257536 /lib64/libpw5.so
named     20066              named  mem       REG                8,3
34536   16257536 /lib64/libpw5.so
dnsadmin  21983               root  mem       REG                8,3
34536   16257536 /lib64/libpw5.so
```

In this example, the first line shows that the `/lib64/libpw5.so` file intertwines with the `auditd` program. The `auditd` program, or Linux Auditing System, writes audit records to disk.

If you run the `lsof -p 1577` command, which checks the PID of the `auditd` program, you will see results similar to the following:

```
COMMAND  PID USER   FD   TYPE             DEVICE SIZE/OFF     NODE NAME
auditd  1577 root  cwd    DIR                8,3     4096        2 /
auditd  1577 root  rtd    DIR                8,3     4096        2 /
auditd  1577 root  txt    REG                8,3   104544 41681130
/sbin/auditd
auditd  1577 root  mem    REG                8,3    88600 16252994
/lib64/libz.so.1.2.3
auditd  1577 root  mem    REG                8,3    20024 16253063
/lib64/libdl-2.12.so
auditd  1577 root  mem    REG                8,3  1924768 16252941
/lib64/libc-2.12.so
auditd  1577 root  mem    REG                8,3   596864 16253075
/lib64/libm-2.12.so
auditd  1577 root  mem    REG                8,3    44472 16253350
/lib64/librt-2.12.so
auditd  1577 root  mem    REG                8,3   143280 16252965
/lib64/libpthread-2.12.so
auditd  1577 root  mem    REG                8,3   145864 16252997
/lib64/libaudit.so.1.0.0
auditd  1577 root  mem    REG                8,3   113904 16253078
/lib64/libnsl-2.12.so
auditd  1577 root  mem    REG                8,3    40792 16253035
/lib64/libwrap.so.0.7.6
auditd  1577 root  mem    REG                8,3    34536 16257536
/lib64/libpw5.so
auditd  1577 root  mem    REG                8,3  1971488 50333097
/usr/lib64/libcrypto.so.1.0.1e
auditd  1577 root  mem    REG                8,3   159312 16257525
/lib64/ld-2.12.so
auditd  1577 root    0u   CHR                1,3      0t0     3857
/dev/null
auditd  1577 root    1u   CHR                1,3      0t0     3857
/dev/null
auditd  1577 root    2u   CHR                1,3      0t0     3857
/dev/null
auditd  1577 root    3u  unix 0xffff8804384f1c00      0t0    14145
@/tmp/dbus-JFgUwQ7Nx3
```

The last line represents a socket connection in a temporary directory, which is highly suspicious. If we check for that name with the `netstat` com mand, we see that it listens for connections:

```
# netstat -nap | grep "@/tmp/dbus-JFgUwQ7Nx3"
unix  2      [ ACC ]     STREAM     LISTENING     14145  1577/auditd
@/tmp/dbus-JFgUwQ7Nx3
```

No reason exists for the `auditd` program to maintain a socket connection to the outside world. Likely in this case, hackers modified this binary enough to search for SSH keys or root passwords, which the binary then sends to a Command and Control server for later use.

FAQ

How does a root compromise differ from a site compromise?

A site compromise occurs on a specific website and malicious attackers can only steal information from that website, not from any others. A root compromise exposes the entire server to theft, such as SSH keys and passwords. It can also cause enough damage to the server so that it can no longer boot.

But what about symlink hacks that can hack multiple websites?

Most experts do not consider symlink hacks themselves to be root compromises, even though they look like they have hacked multiple accounts. The server itself is usually not in any danger.

For more information on how to prevent symlink hacks, read our Symlink Race Condition Protection documentation.

I've seen symlinks that can grab the `/etc/passwd` file. Isn't that a root compromise?

No. The `/etc/passwd` file must be readable, but this does not represent a root compromise. The system stores password hashes in the `/etc/shadow` file, which you cannot simply view with a symlink hack. A further definition of a root-level compromise is one where an unauthorized user gains access to it. You may not lose much from a particular attack, but any unauthorized access has the potential for further breaches.

Every rootkit has at lease 2 purposes:

1. Hides the attacker.
2. Grants access to the attacker.

A root-level compromise exposes the **entire** server, and you should consider **everything** a loss. You can no longer trust any data, any configuration information, and probably any connectivity information and passwords. An attacker will likely want to continue to access to the server, and they will try to configure the system as to perform normally. This makes rootkits difficult to detect, but it is not impossible to detect them.

Additional documentation

Suggested documentation For cPanel users For WHM users For developers

- Determine Your System's Status
- How to Set or Unset RPM Management
- RPM Installation Failures
- How to Build and Install Custom RPMs
- How to Set Up a Third-Party RPM Repository with GPG Signatures
    - An In-Depth Analysis of Linux Ebury
    - Windigo, Not Windigone
    - Windigo/Ebury Update
    - Windigo Whitepaper
- How to Use cPanel API Tokens
- Security
- Man-in-the-Middle Attacks
- How to Configure Microsoft Windows 7 to use TLS Version 1.2
- Important Notices

- Determine Your System's Status
- How to Set or Unset RPM Management
- RPM Installation Failures
- How to Build and Install Custom RPMs
- How to Set Up a Third-Party RPM Repository with GPG Signatures

- Guide to Standardized Hooks - RPM::Versions Functions
- WHM API 1 Functions - get_rpm_version_data
- WHM API 1 Functions - list_rpms
- WHM API 1 Functions - edit_rpm_version
- WHM API 1 Functions - delete_rpm_version