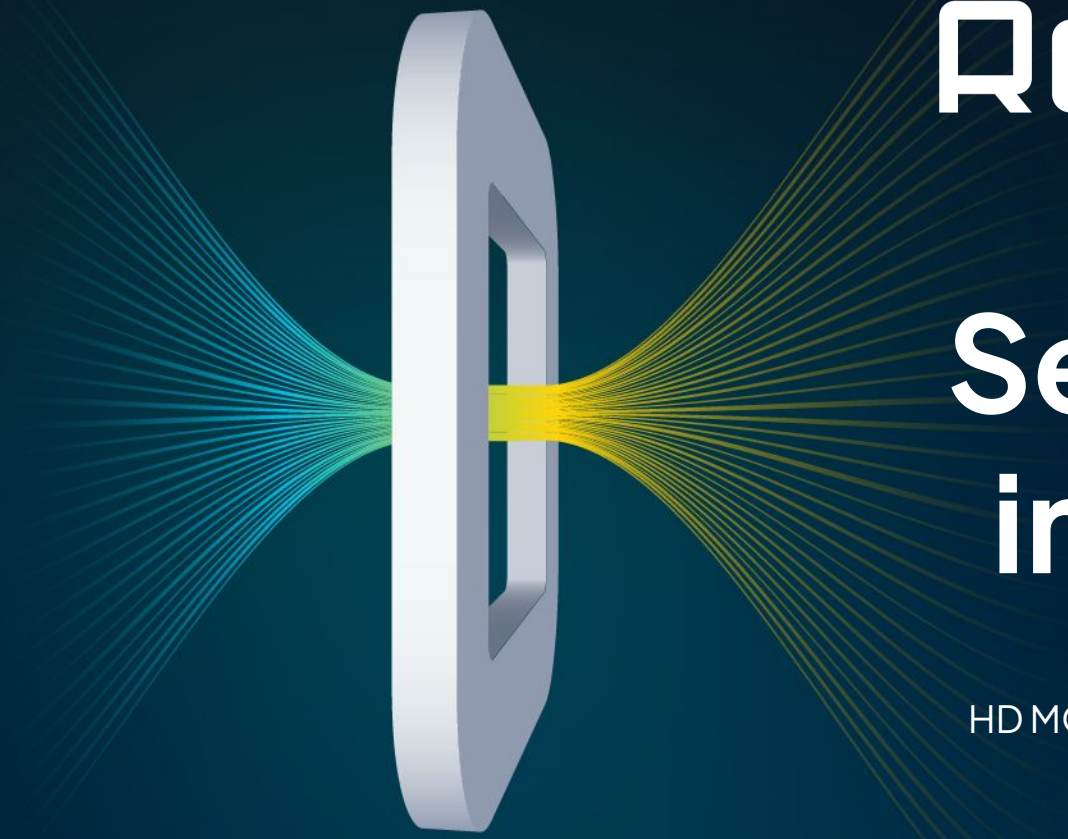
black hat USA 2024

runZero RESEARCH

BLACK HAT BRIEFINGS



Secure Shells in Shambles

HD MOORE | ROB KING | AUGUST 7, 2024

Agenda

runZero

This is a talk about the evolution of the Secure Shell (SSH)

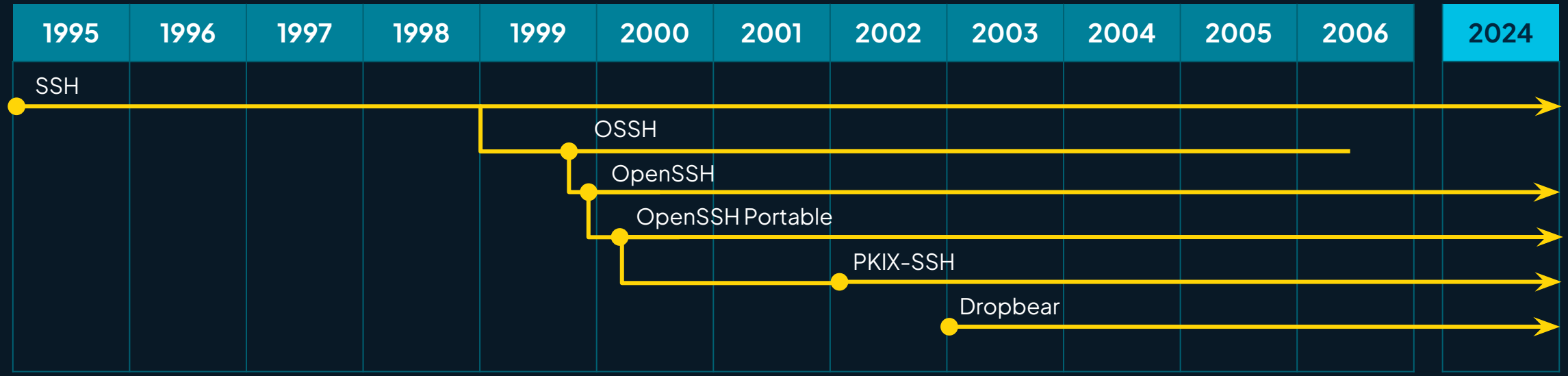
- → An overview of the SSH ecosystem
- → What's changed & what hasn't
- → New & interesting attacks
- → OpenSSH fragmentation
- → Introducing **SSHamble**
- → Defending SSH

In the beginning was SSH

runZero

Tatu Ylönen created SSH v1 in 1995 as freeware

- → Continued development as the proprietary SSH.com
- → Björn Grönvall forked Ylönen's free SSH v1.2.12 as OSSH
- → OpenBSD forked OSSH into OpenSSH in 1999



SSH is mostly OpenSSH & Dropbear

runZero

OpenSSH	20,200,340
Dropbear sshd	5,482,314
Linksys WRT45G modified dropbear sshd	46,214
lancom sshd	43,574
SCS sshd	8,215
HP Integrated Lights-Out mpSSH	7,493
WeOnlyDo sshd	6,458
ZyXEL ZyWALL sshd	3,417
NetScreen shhd	1,854
DrayTek Vigor 2820n ADSL router sshd	1,848
CoreFTP sshd	1,700

Not-OpenSSH/Dropbear are important

Firewall, networking, & storage

- → Cisco, NetScreen, Adtran, ComWare, Lancom

OT/ICS equipment

- → Siemens, NetPower, Mocana, CradlePoint, Digi

Sensitive applications

- → MOVEIT, CrushFTP, GlobalScape,JSCAPE
- → BitVis, GoAnywhere, ConfD
- → Gerrit, Forgejo, Gitlab

SHODAN https://www.shodan.io/search/facet?query=shodan.module%3A%22ssh%22&facet=product

Other implementations

runZero

Standalone product examples

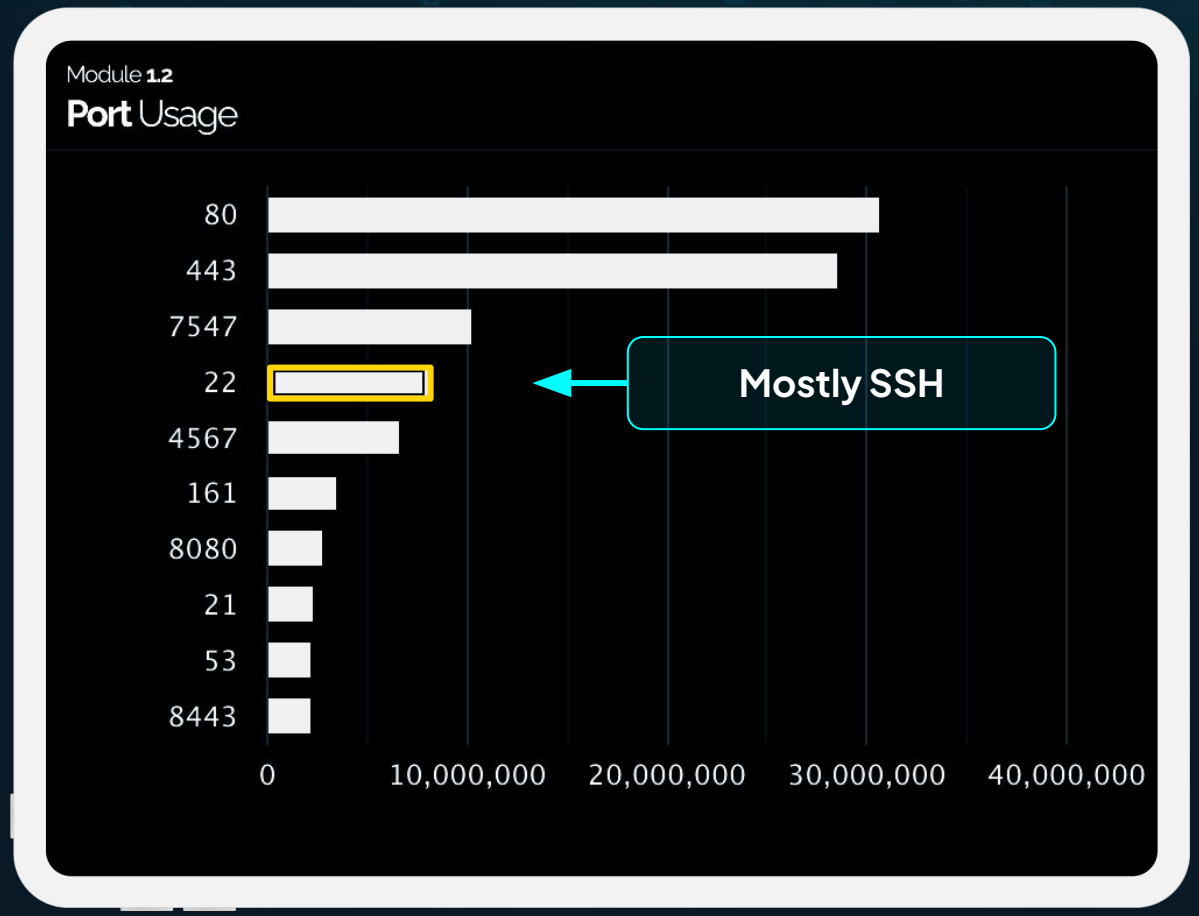
- PKIX-SSH — popular in networking equipment, forked from OpenSSH
- WolfSSH — small implementation popular in embedded systems
- lsh — an old implementation that predates OpenSSH Portable

SSH library examples

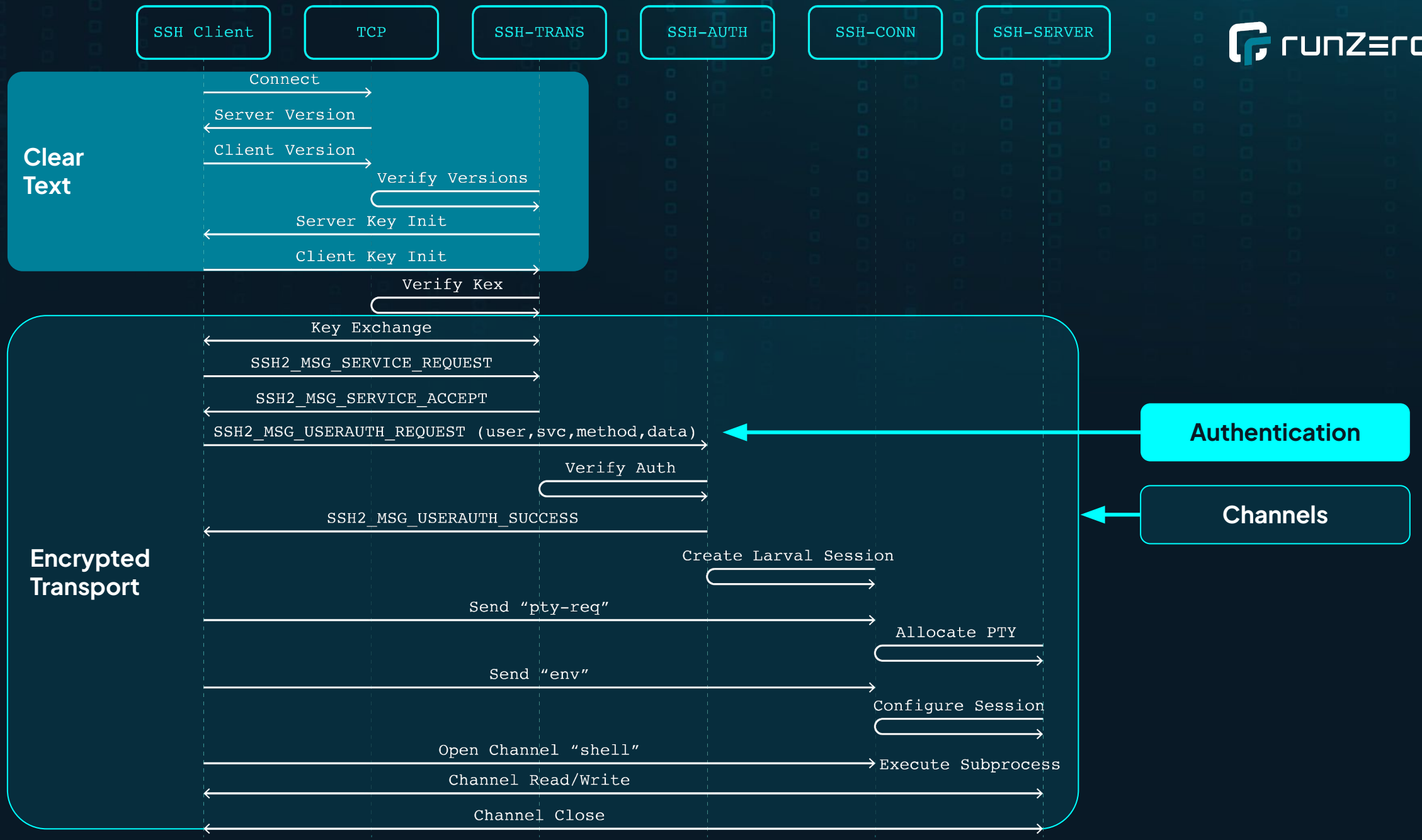
- libssh — open source, bindings for lots of languages
- Go x/crypto/ssh — a pure Go implementation
- Apache MINA — a Java implementation
- Paramiko — SSH in Python

SSH is everywhere

- Second-most common remote admin service behind HTTP
- Enabled by default in clouds
- Part of every major OS
- Embedded & servers
- Even mobile!



SHODAN https://exposure.shodan.io/#/US


SSH Client
TCP
SSH-TRANS
SSH-AUTH
SSH-CONN
SSH-SERVER
Clear Text
Connect
Server Version
Client Version
Verify Versions
Server Key Init
Client Key Init
Verify Kex
Key Exchange
SSH2_MSG_SERVICE_REQUEST
SSH2_MSG_SERVICE_ACCEPT
SSH2_MSG_USERAUTH_REQUEST (user,svc,method,data)
Verify Auth
SSH2_MSG_USERAUTH_SUCCESS
Encrypted Transport
Create Larval Session
Send "pty-req"
Allocate PTY
Send "env"
Configure Session
Open Channel "shell"
Execute Subprocess
Channel Read/Write
Channel Close
Authentication
Channels

SSH provides transport & authentication

runZero

Version exchange & kex init in the clear

- Version: SSH-2.0 OpenSSH-9.8p1 deb13u3
- Ciphers, MACs, Compressions, Languages, etc

Key exchange to negotiate secure transport

- Diffie-Hellman & friends pinned with server host key(s)
- Algorithm picked by kex init agreement

Authentication using one or more methods

- Passwords, public keys, kerberos, & more
- PK uses the session ID for proof signing

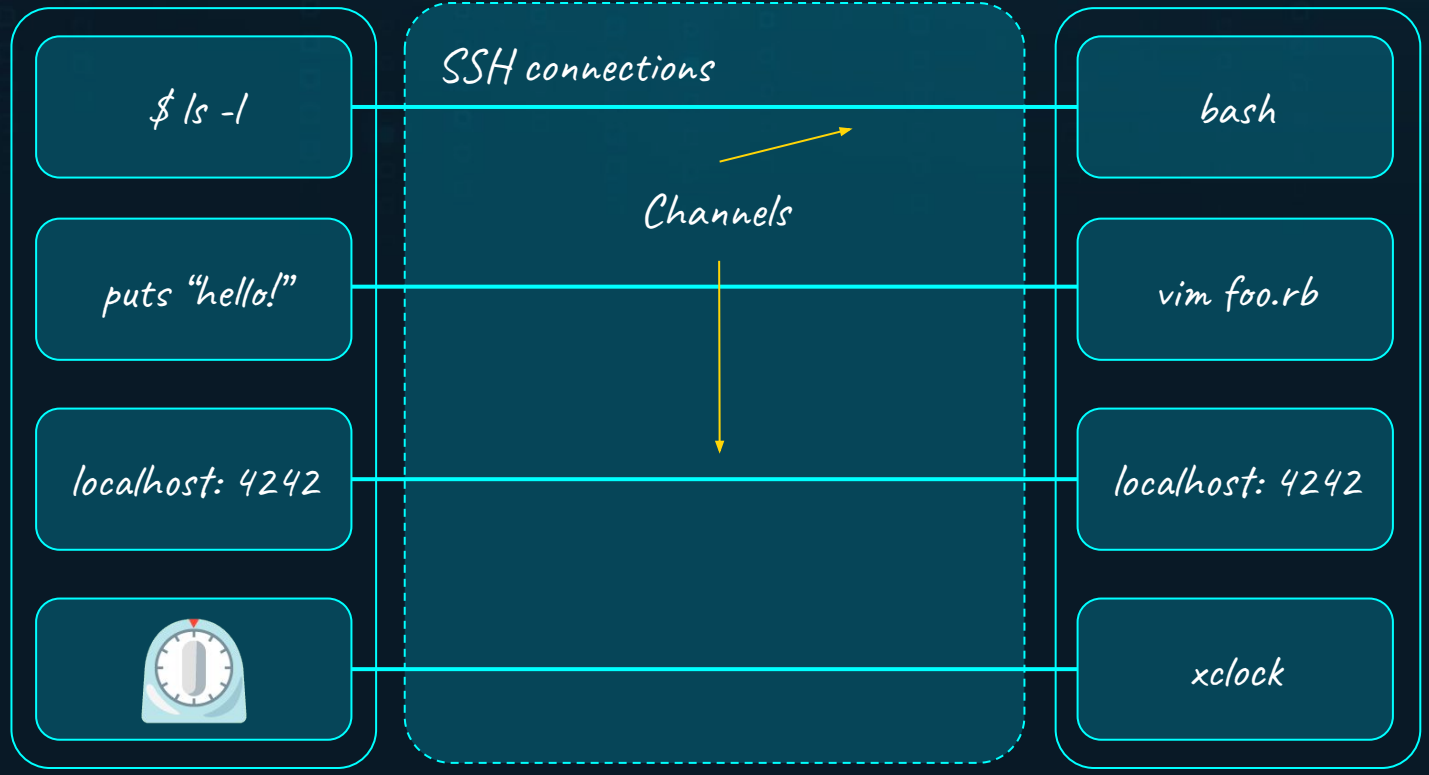
Similar to TLS

Channels, subsystems, & shells, oh my!

runZero

SSH multiplexes multiple channels (concurrently)

- Interactive shells
- Command execution
- File transfer (SCP, SFTP)
- TCP forwarding
- Unix socket forwarding
- X11 display forwarding
- Agent forwarding



SSH is the other secure transport

runZero

An alternative to TLS, but not exactly the same

- Server key management can be, but usually isn't CA-based
- Authentication is a core stage of the protocol
- Multiplexer & session commands are unique
- SSH uses the first algorithm sent by the client & supported by the server

Compliance schemes gloss over SSH

- Vendors point to strong cipher/mac + authentication similar to TLS
- SSH specifics are often missing, assume best practices
- Key management is the biggest gap

What's New?

More protocol extensions

runZero

ping	Ping & pong
server-sig-algs	Support for more algorithms
publickey-hostbound-v00	Host-bound public keys
tun	Layer 2 & 3 tunneling
hostkeys/hostkeys-prove	Host key rotation
aes128–gcm,hmac-sha1–etm, …	New cipher, kex, & MACs

SSHFP: Verify server host keys via DNS

runZero

DNS record format defined in RFC 4255

- Key Algorithm + Hash Type + Fingerprint
  - 4 [ED25519] / 2 [SHA256] / 0A2B3C [SHA256 hash]
- Enforce client-side with -o VerifyHostKeyDNS=yes
- Enumerate via dig or ssh-keyscan
  - `dig -t SSH example.com`
  - `ssh-keyscan -D example.com`

Low adoption as of late 2021*

- Enabled for 1 in every 10,000 domains tested
- Only 50% use DNSSEC

** See "Neef, S., Wisiol, N. (2022). Oh SSH-it, What's My Fingerprint? A Large-Scale Analysis of SSH Host Key Fingerprint Verification Records in the DNS"*

MFA for SSH: Interactive OTP

runZero

Traditional SSH MFA is via PAM plugins

After Password

```
$ ssh dev@192.168.67.2
(dev@192.168.67.2) Password:
(dev@192.168.67.2) Verification code:
```

Before Password

```
$ ssh dev@192.168.67.3
 https://api-abc1234.duosecurity.com
 /frame/portal/v4/enroll?code=012…
```

- → Uses **challenge-response** or **keyboard-interactive*** mode
- → Google Auth, Duo Security, QQ.com, Qomolo, & more

* **keyboard-interactive** usually just means **password**, but it is also used for interactive OTP.

MFA for SSH: FIDO2 resident keys

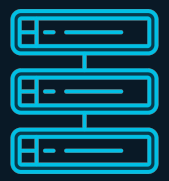
runZero

Use a token-aware SSH agent

- https://github.com/FiloSottile/yubikey-agent
- https://github.com/maxgoedjen/secretive

Use the new “sk” key types

- `ssh-keygen -t ed25519-sk -O resident -O verify-required`
- `ssh-keygen -K`



SSH Server (optional)

- `PubkeyAuthOptions verify-required`

Centralized SSH authentication

runZero

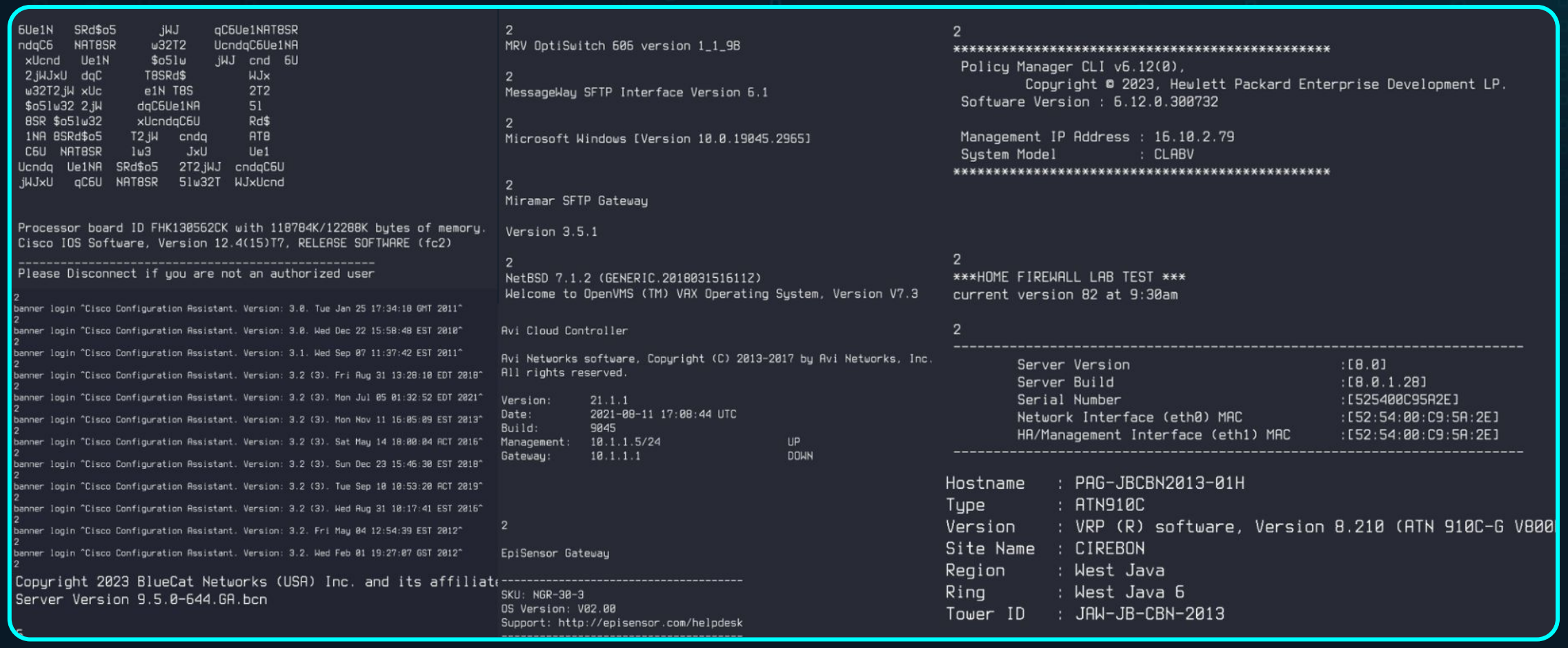
Certificates with short-expiration signed SSH keys

- Authenticate to an IDP, get a signed SSH key
- Use the signed key like a normal private key
- The gold standard for managed SSH

Projects & products

- Opera SSH Key Authority (SKA)
- HashiCorp Vault SSH Certificate Secret Engine
- Tectia UKM, Teleport, UserFi, SpanKey, Delinea, & more!

Useful pre-authentication banners



SSH key types, exchanges, extensions

runZero


```
"version": "SSH-2.0-OpenSSH_9.7p1 Debian-5",
"kex": {
  "kexAlgos": [
    "sntrup761x25519-sha512@openssh.com",
    "curve25519-sha256",
    "curve25519-sha256@libssh.org",
    "ecdh-sha2-nistp256",
    "ecdh-sha2-nistp384",
    "ecdh-sha2-nistp521",
    "diffie-hellman-group-exchange-sha256",
    "diffie-hellman-group16-sha512",
    "diffie-hellman-group18-sha512",
    "diffie-hellman-group14-sha256",
    "ext-info-s",
    "kex-strict-s-v00@openssh.com"
  ],
  "hostKeyAlgos": [
    "rsa-sha2-512",
    "rsa-sha2-256",
    "ecdsa-sha2-nistp256",
    "ssh-ed25519"
  ],
  "cipherC2S": [
    "chacha20-poly1305@openssh.com",
    "aes128-ctr",
    "aes192-ctr",
    "aes256-ctr",
    "aes128-gcm@openssh.com",
    "aes256-gcm@openssh.com"
  ],
```

```
{
  "publickey-hostbound@openssh.com": "0",
  "server-sig-algs": "ssh-ed25519,sk-ssh-ed25519@openssh.com,ssh-rsa,rsa-sha2-256,rsa-sha2-512,ssh-dss,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,sk-ecdsa-sha2-nistp256@openssh.com,webauthn-sk-ecdsa-sha2-nistp256@openssh.com"
}
{
  "server-sig-algs": "ssh-ed25519,sk-ssh-ed25519@openssh.com,ssh-rsa,rsa-sha2-256,rsa-sha2-512,ssh-dss,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,sk-ecdsa-sha2-nistp256@openssh.com,webauthn-sk-ecdsa-sha2-nistp256@openssh.com"
}
{
  "ping@openssh.com": "0",
  "publickey-hostbound@openssh.com": "0",
  "server-sig-algs": "ssh-ed25519,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,sk-ssh-ed25519@openssh.com,sk-ecdsa-sha2-nistp256@openssh.com,rsa-sha2-512,rsa-sha2-256"
}
{
  "server-sig-algs": "rsa-sha2-256,rsa-sha2-512"
}
{
  "server-sig-algs": "ssh-ed25519,ssh-rsa,rsa-sha2-256,rsa-sha2-512,ssh-dss,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521"
}
```

OpenSSH's new PerSourcePenalties

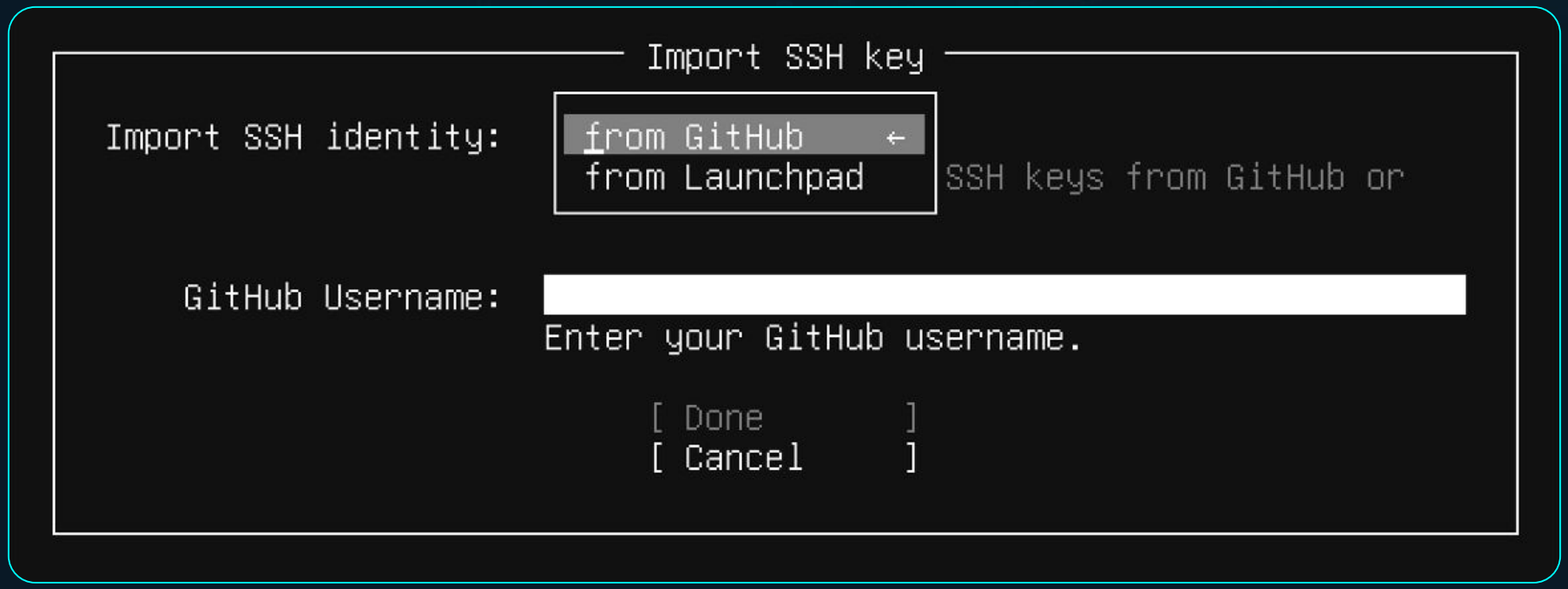
runZero

PerSourcePenalties

> *Controls penalties for various conditions that may represent attacks on sshd(8). If a penalty is enforced against a client then its source address and any others in the same network, as defined by PerSourceNetBlockSize, will be refused connection for a period.*
>
> *A penalty doesn't affect concurrent connections in progress, but multiple penalties from the same source from concurrent connections will accumulate up to a maximum. Conversely, penalties are not applied until a minimum threshold time has been accumulated.*
>
> *Penalties are enabled by default with the default settings listed below but may disabled using the no keyword. The defaults may be overridden by specifying one or more of the keywords below, separated by whitespace. All keywords accept arguments, e.g. "crash:2m".*

SSH keys as public identities

- Public keys used to being mostly private
- GitHub & Launchpad changed that



runZero

```
ssh whoami.filippo.io
```

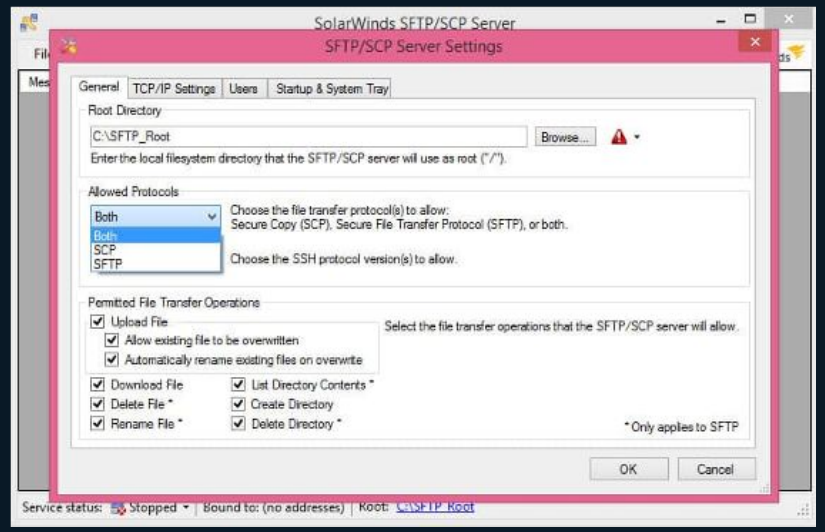
```
+-------------------------------------------------------------------+
|                                                                   |
|              _o/ Hello HD Moore!                                  |
|                                                                   |
|                                                                   |
|  Did you know that ssh sends all your public keys to any server   |
|  it tries to authenticate to?                                     |
|                                                                   |
|  We matched them to the keys of your GitHub account,              |
|  @hdm, which are available via the GraphQL API                    |
|  and at https://github.com/hdm.keys                               |
|                                                                   |
|  -- Filippo (https://filippo.io)                                  |
|                                                                   |
|                                                                   |
|  P.S. The source of this server is at                             |
|  https://github.com/FiloSottile/whoami.filippo.io                 |
|                                                                   |
+-------------------------------------------------------------------+
```

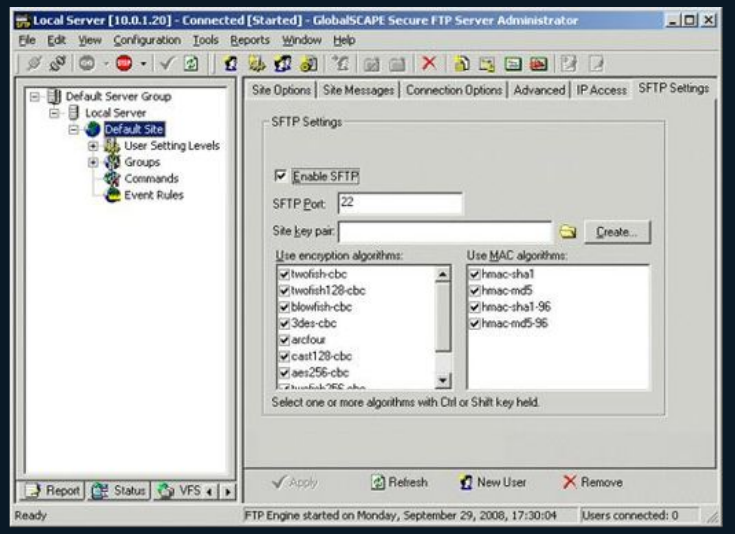
SFTP as a *de facto* standard for MFT

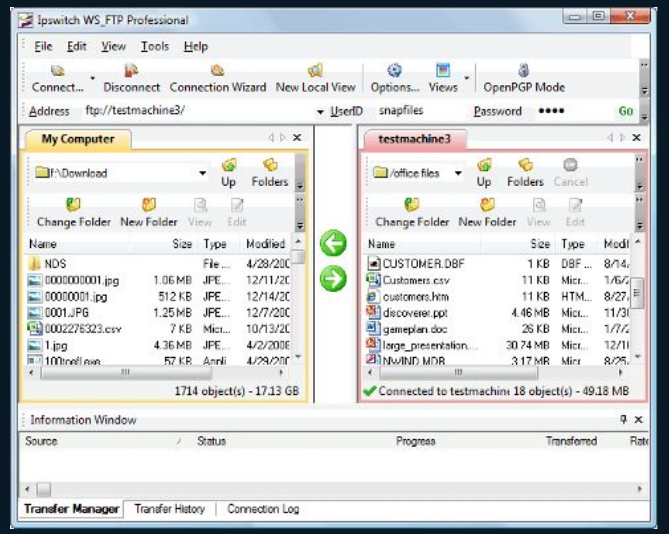
runZero

Commercial MFT products support SCP/SFTP

- Many are based on existing third-party SSH libraries
- Axway, GlobalScape, CuteFTP, Cerberus, Bitvise
- SolarWinds, JSCAPE, FileZilla, Kiteworks, WS_FTP







Return of the terminal

runZero

Libraries for Go & Rust have created a TUI renaissance

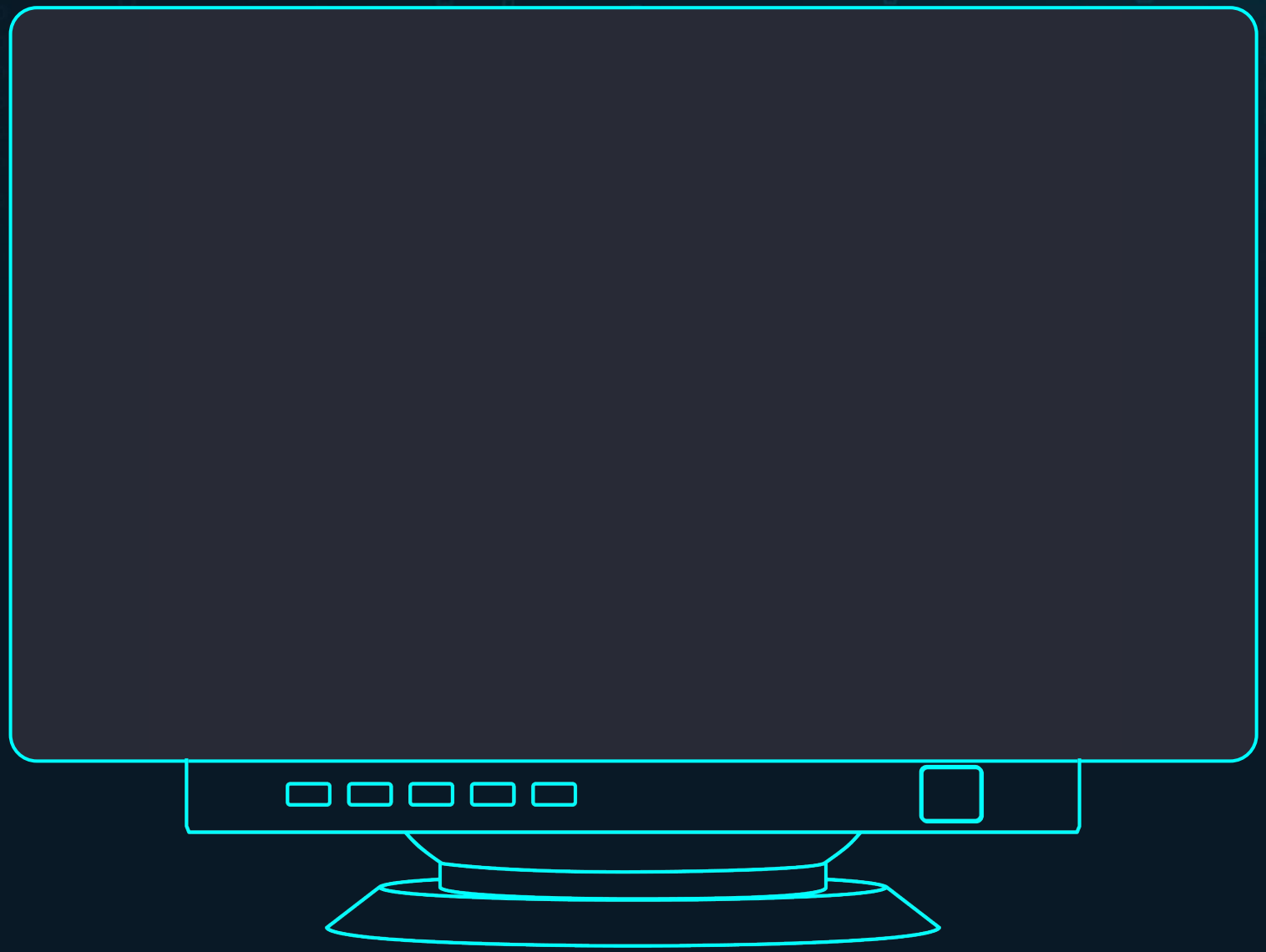
- → Pretty interfaces delivered right to your screen via SSH
- → Treat SSH almost like TLS with optional authentication

SSH libraries are used to power source code forges

- → Go-based GOGS, Gitea, Forgejo, & soft-serve
- → Apache Mina supports Gerrit
- → Azure DevOps Server (VS TFS)

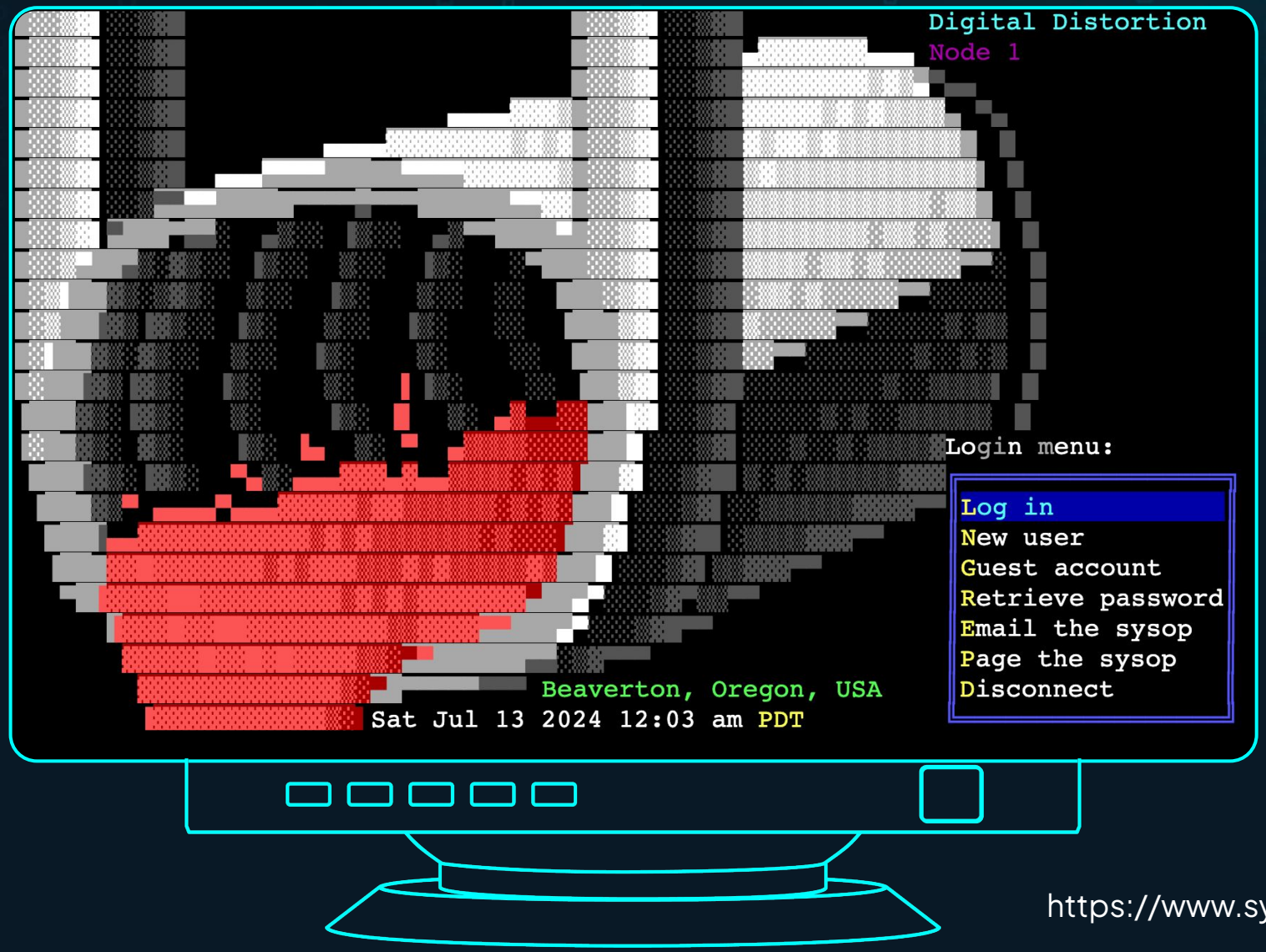
$ ssh starwarstel.net

runZero



$ ssh user@synchronet

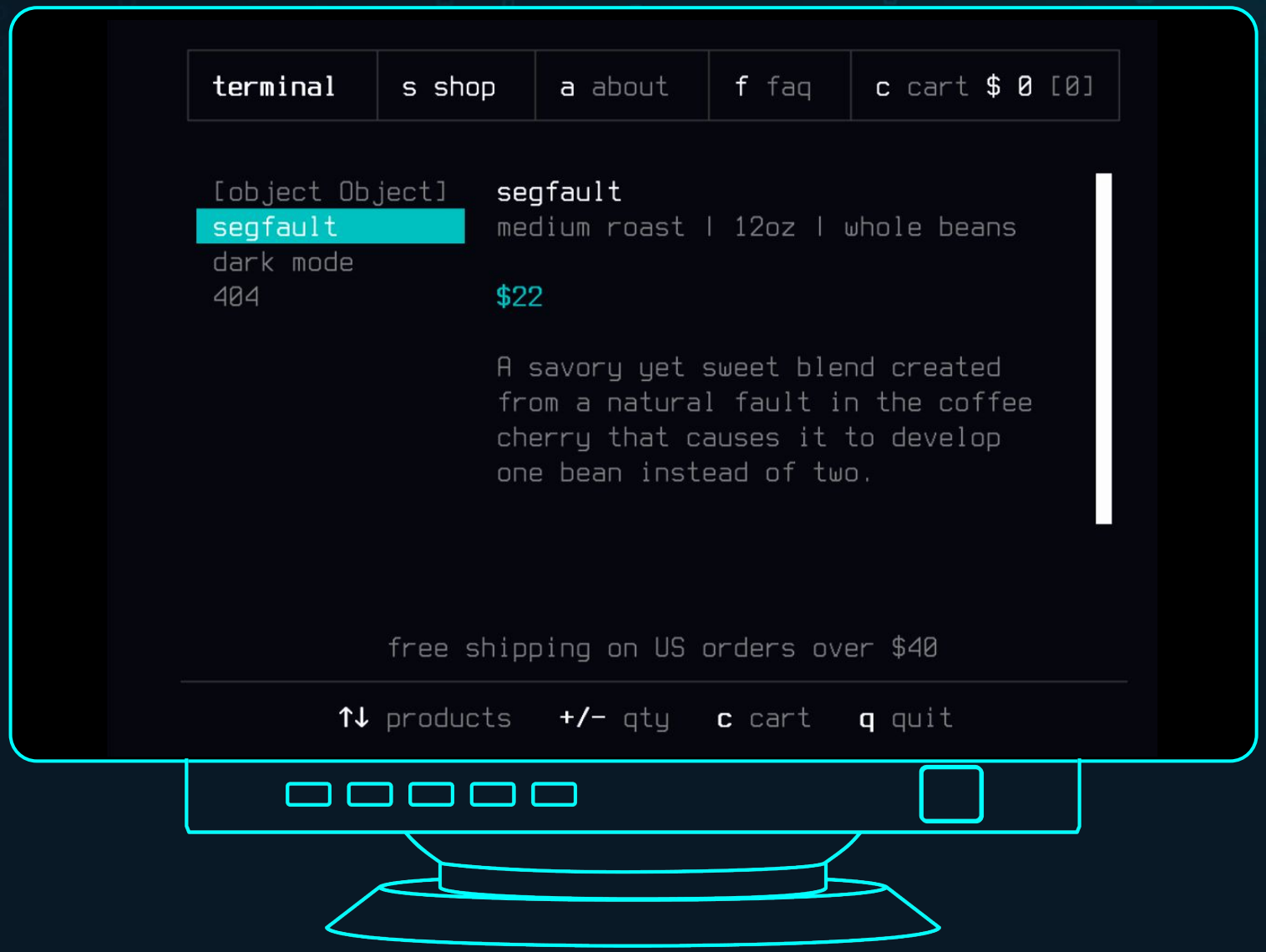
runZero



https://www.synchro.net/sbbslist.html

$ ssh terminal.shop

runZero



Recent Exposures

Terrapin Attack

runZero

Breaking SSH Channel Integrity by Sequence Number Manipulation

Fabian Bäumer
Research Assistant, Ruhr University Bochum

black hat USA 2024

Thursday, August 8 @
11:20am-12:00pm
Islander FG, Level 0

CVE-2023-48795

XZ Utils backdoor

runZero

A multi-year campaign started in 2021 and triggered in 2024

- “Jia Tan” persona was likely the product of a state actor
- Nearly-perfect Nobody-But-Us backdoor in SSH
- Backdoor targeted SSH via systemd patches
- Limited to Debian/RHEL-based distros

Caught at the last possible moment by Andres Freund

- Noticed that sshd was using more CPU than it should
- Backdoor made it into rolling releases only

CVE-2024-3094

RegreSSHion

runZero

Incredible work by the Qualys Threat Research Unit

- Regression of a signal re-entrance vulnerability
- Unauthenticated remote root code execution
- Tough to exploit due to ASLR & timing

CVE-2024-6387

Related issue discovered by Solar Designer

- Specific to Red Hat builds of OpenSSH
- Limited to the non-root privsep user

CVE-2024-6409

The patch was hidden in the PerSourcePenalties feature, released a month prior to the disclosure.

MOVEit & IPWorks SSH

runZero

Another MOVEit vulnerability, but this time in SSH

- watchTowr Labs reversed the MOVEit patch for CVE-2024-3094
- The attacker's unauthenticated public key blob is opened as a file
- File path supports UNC and was used for authentication
- Root cause was the third-party IPWorks library
- Threaded a dozen needles to bypass auth

CVE-2024-5806

What's the Same?

Unauthenticated information exposure

runZero

TCP/IP → **Server Version** → **Kex Init Extensions** → **Server Banner** → **Authentication**

TCP window size & scaling factors can determine the OS & kernel versions.

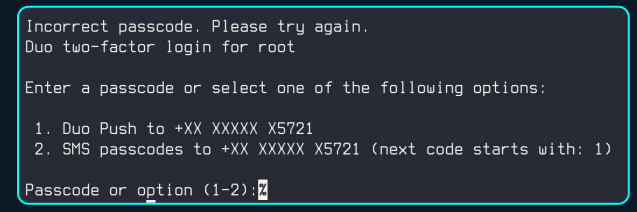
Protocol version, implementation, & package version.

Ciphers, MACs, key exchange protocols, compression methods, & server-side extensions.

Pre-authentication "banner" can be extensive, especially with network equipment.

Authentication method list, public key testing, failed auth limits, & interactive questions & prompts.

Platform	Version	SSH banner	v4 tcp.win	v4 MSS	MSS Multiplier	v4 Window Scale
CentOS Linux	7.1	SSH-2.0-OpenSSH_6.6.1	14480	1460	10	7
CentOS Linux	7.2	SSH-2.0-OpenSSH_6.6.1	28960	1460	20	7
CentOS Linux	7.3		28960	1460	20	7
CentOS Linux	7.4	SSH-2.0-OpenSSH_7.4	28960	1460	20	7
CentOS Linux	7.5		28960	1460	20	7
Oracle Linux Server	7.7		28960	1460	20	7
CentOS Linux	7.9		28960	1460	20	7
Oracle Linux Server	7.9		28960	1460	20	7
Scientific Linux	7.9		28960	1460	20	7
CentOS Linux	8.0	SSH-2.0-OpenSSH_7.8	28960	1460	20	7
Oracle Linux Server	8.0		28960	1460	20	7



A large post-auth attack surface

runZero

Restricted shell environments are difficult to secure

- Multiplexed channels
- Connection forwarding
- Environment manipulation
- Subsystems (SFTP, etc)
- X11 forwarding
- PTY requests
- Client-sent signals
- Window size changes
- Break commands
- Agent auth requests

Default exposure to brute force attacks

runZero

Admins are generally left to figure it out on their own

- Fail2Ban & PAM lockouts can help, but incomplete
- `PerSourcePenalties` will help, but not yet widely deployed

Horrific amount of wasted CPU due to constant attacks

- A real impact on embedded device performance
- Still not as terrible as blockchains or AI

Public key authentication is still weird

runZero

Attacker can verify public keys without the private key

- Servers reply with PK_OK for valid public keys
- Clients then send the public key + signature
- Leads to information leaks

Public key auth is flexible, but is easy to get wrong

- Dynamic PK authentication via `AuthorizedKeysCommand`
- CA user key management & revocations are finicky

Host key management is error prone

Host key duplication is incredibly common

- → Vendors accidentally hard-code firmware & VMs
- → Cloud providers still get this wrong with images
- → VMware hosts often set host key in gold image

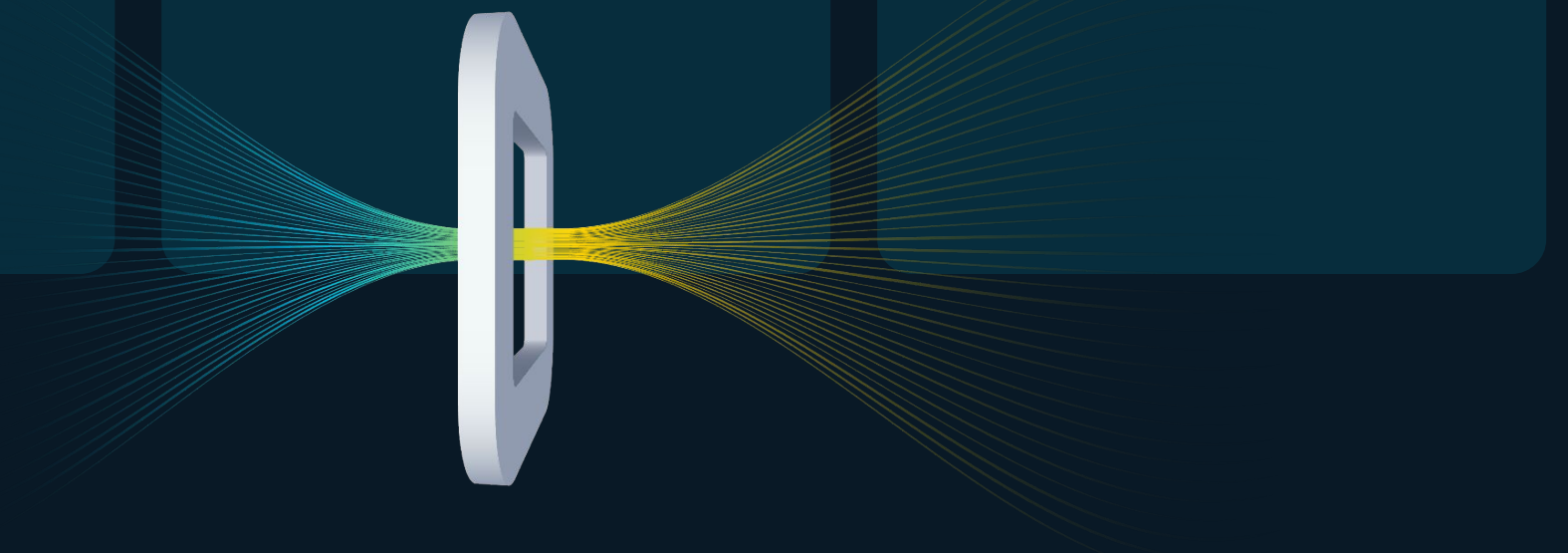
Host keys are rarely changed due to challenges

- → GitHub exposed their main RSA key in 2023
- → Rotation broke automation & upset users
- → Compare to modern TLS rotations
- → CAs can help, but tricky at scale

SSH is still (used as) a transport layer

runZero

SSH as a generic secure transport layer	SFTP & SCP are a popular way to move files	Port forwarding & traffic tunneling
→ git, rsync, systemctl, docker, duplicati, ssh-fs	→ sftp-only shells, tons of commercial tools	→ vendor-appliances & light VPNs



New Meets Old

(Public Key Authentication)

Public key authentication is two-stage

runZero

An SSH client can confirm if a public key is valid for a given user

- → Metasploit support since 2012, but still not widely known
- → The security impact is minimal?

```
/* XXX fake reply & always send PK_OK ? */
/*
* XXX this allows testing whether a user is allowed
* to login: if you happen to have a valid pubkey this
* message is sent. the message is NEVER sent at all
* if a user is not allowed to login. is this an
* issue? -markus
*/
```


OpenSSH Source (9.8p1)

Link a user & key to a specific server

runZero

Servers

A list of IP addresses or hostnames running SSH.

Scanners
- → nmap
- → zmap
- → masscan

Databases
- → Shodan
- → Censys
- → Fofa.info

Public Keys

A list of public keys possibly linked to the target.

GitHub

Canonical Launchpad

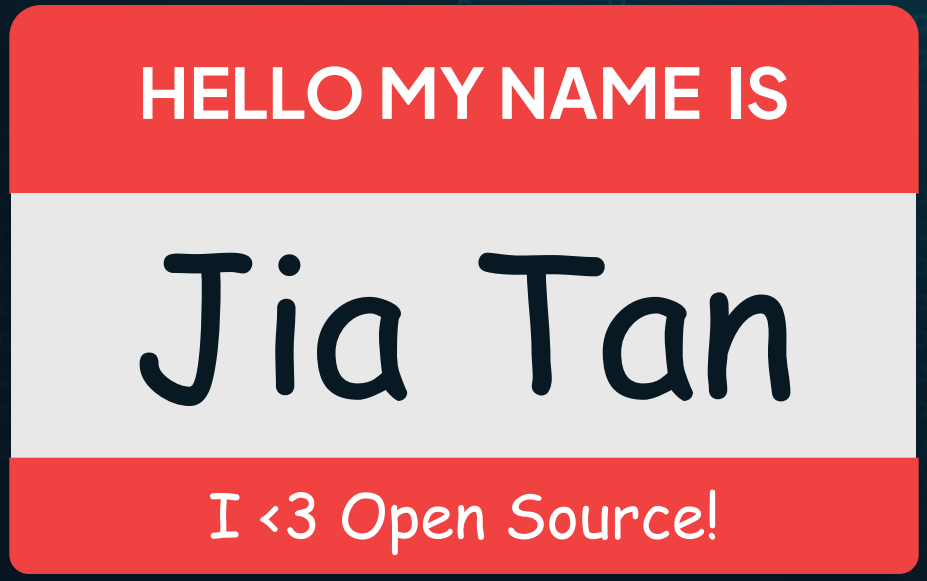
BadKeys

Usernames

A list of usernames likely used by the target.

Defaults
- → root
- → ec2–user
- → ubuntu

Specific
- → Public key “comments”
- → Common handles
- → Email prefixes



runZero

```
$ curl https://github.com/JiaT75.keys

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDHVp3Bvg/ALC61dsGehbvoqic49D4SfoiiPURSEec3/phZdAfR1hD6QSNTHLY3QDT
b0994ZwOFi05YpUM6/qwBUAbroS64/Mp55qDBlark5v83LcTq7a29VUH3Xvu7sAgdYda16a2KnmU5lhETvBfxuS+tpGin9r
aSp+B+z0PIpr9EmEeQgKtgKRQBiMWMtw7jBxm5INk54SmePNDva3f4ml08/Z4JM76dJ7DBQGrLUqZGsRFOZclMb3YOE7DjP
GQQ37TzGvKwLaGvRuocA8oW5zp07+uQldP2LIbt0V99eyXrgD7WLc/sdzWeefoNltcgcV/KEg9ivD02qWFDBzAKMcJuLMhq
xXIo64KZuVjWRrflgKCk5wZt0XPZ30MFqbBvjhn8zG7bIQJORmn/j6QSyHewu4Rre7uGxAuzee2PPSaSQ51dKgbdn3B3Uuw
N8KeIO54W1VYWip+GlG2tXHZAdJOgPPaM72OAqFQBta2MzcHi3/m2HgUNBttYhSUtaeX8myfiRcnC7APhZMOuU9rrHdti2K
D6IVArtBiorZbs8iFlzUPmdYVdeFP7EtW6EWgZSLV7rN2r2+CNVJeTrX9zA+mnRjhjq4ffgRUoQikY876kY+1YiEERm7LRB
MkKIzM4ZsBk7VQwImSGReyfwEht9tedU5mf5pkrbL8VSMrqQQ==

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIFiXcmAAjTBp5kM2AUTJdAEB7DHyYuY8am8FIMROD3FG
```

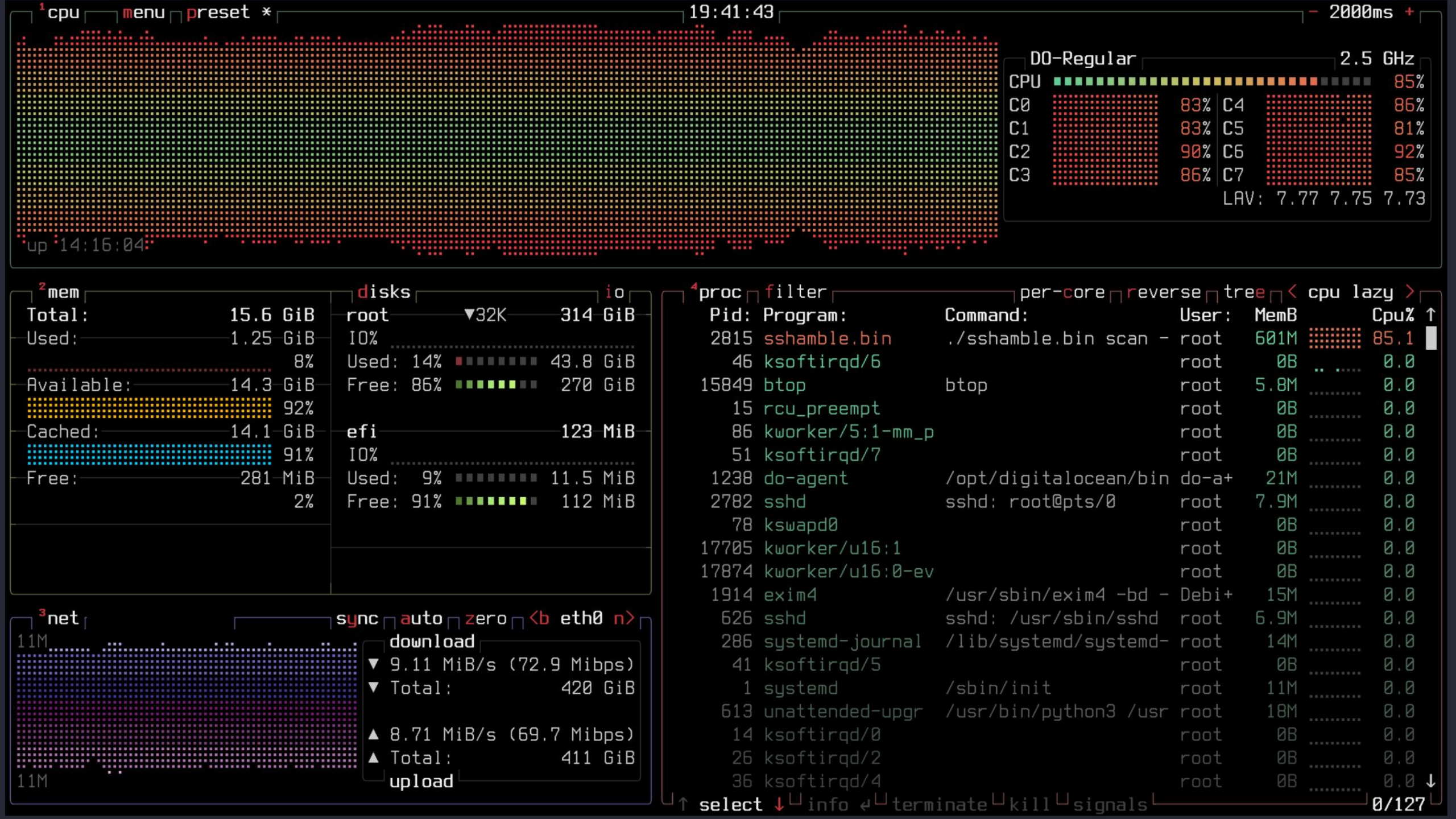
Hunting for Jia Tan across the internet

runZero

After the XZ backdoor was exposed, we went hunting

- → Copied Jia Tan's SSH public keys from GitHub
- → Scanned all of IPv4 for SSH with zmap
- → Created `SSHamble` to half-auth scan
- → Ran `SSHamble` on all SSH hits

We got results!


1 cpu menu preset *
19:41:43
- 2000ms +
up 14:16:04
DO-Regular 2.5 GHz
CPU 85%
C0 83% C4 86%
C1 83% C5 81%
C2 90% C6 92%
C3 86% C7 85%
LAV: 7.77 7.75 7.73
2 mem
Total: 15.6 GiB
Used: 1.25 GiB
8%
Available: 14.3 GiB
92%
Cached: 14.1 GiB
91%
Free: 281 MiB
2%
disks io
root ▼32K 314 GiB
IO%
Used: 14% 43.8 GiB
Free: 86% 270 GiB
efi 123 MiB
IO%
Used: 9% 11.5 MiB
Free: 91% 112 MiB
4 proc filter per-core reverse tree < cpu lazy >
Pid: Program: Command: User: MemB Cpu% ↑
2815 sshamble.bin ./sshamble.bin scan - root 601M 85.1
46 ksoftirqd/6 root 0B 0.0
15849 btop btop root 5.8M 0.0
15 rcu_preempt root 0B 0.0
86 kworker/5:1-mm_p root 0B 0.0
51 ksoftirqd/7 root 0B 0.0
1238 do-agent /opt/digitalocean/bin do-a+ 21M 0.0
2782 sshd sshd: root@pts/0 root 7.9M 0.0
78 kswapd0 root 0B 0.0
17705 kworker/u16:1 root 0B 0.0
17874 kworker/u16:0-ev root 0B 0.0
1914 exim4 /usr/sbin/exim4 -bd - Debi+ 15M 0.0
626 sshd sshd: /usr/sbin/sshd root 6.9M 0.0
286 systemd-journal /lib/systemd/systemd- root 14M 0.0
41 ksoftirqd/5 root 0B 0.0
1 systemd /sbin/init root 11M 0.0
613 unattended-upgr /usr/bin/python3 /usr root 18M 0.0
14 ksoftirqd/0 root 0B 0.0
26 ksoftirqd/2 root 0B 0.0
36 ksoftirqd/4 root 0B 0.0 ↓
↑ select ↓ info ↵ terminate k kill s signals
0/127
3 net sync auto zero <b eth0 n>
11M
11M
download
▼ 9.11 MiB/s (72.9 Mibps)
▼ Total: 420 GiB
▲ 8.71 MiB/s (69.7 Mibps)
▲ Total: 411 GiB
upload

The ~~friends~~ shells we found along the way

runZero

And every single result was a false positive for Jia Tan

- → Tons of honeypots & misbehaved servers
- → Reworked the tools & tried again
- → Still no Jia Tan :(

We found thousands of unauthenticated shells instead

- → Some honeypots, but mostly real bugs
- → This work led to this talk!

Dear Law Enforcement,

- → Our scans resulted in Jia's public key hash & our IP is in everyone's logs
- → Please don't arrest us!

Speeding up public key testing

runZero

SSH servers implement MaxAuthTries

- → OpenSSH defaults to 5 & counts pubkey tests
- → This is why having >4 keys in your agent breaks
- → Not all servers count pubkey tests as failed...

Rapid testing with a single connection

runZero

10% of all public SSH servers do not rate limit key testing

→ Dropbear is the most common, but many others

GlobalScape EFT	Maverick SSHD	LANCOM	Adtran
BitVise WinSSHD	GoAnywhere	Arris	Crestron
CrushFTPd	mod_sftpd	Medallia	+ Many More!

Testing millions of public keys fast

runZero

```
% wc -l github-2018.keys
 4,673,197 data/github.keys
```

```
% nc 192.168.68.2 22
SSH-2.0-dropbear_2022.83
```

```
% sshamble scan --checks pubkey-hunt \
--pubkey-hunt-conn-limit 1000000 --pubkey-hunt-file github-2018.keys \
-u root 192.168.68.2
192.168.68.2:22 pubkey-hunt is running with 4673197 test keys
192.168.68.2:22 pubkey-hunt completed 4673190/4673197 keys in 7m37s (10544/s)
192.168.68.2:22 pubkey-hunt accepted hunted half-auth for root with key ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDipNPRHvHknF6WLl7oEPoxxH7k13iKA/14yiWwOwHAUFg+1tl….


dropbear[2921]: Exit before auth from <192.168.68.1:50311>: Exited normally
```

single connection

Compare vs OpenSSH MaxAuthLimit=5

runZero

```
% wc -l github-2018.keys
 4,673,197 data/github.keys
```

```
% nc 192.168.68.2 2222
SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u3
```

```
% sshamble scan --checks pubkey-hunt \
--pubkey-hunt-conn-limit 1000000 --pubkey-hunt-file github-2018.keys \
-u root 192.168.68.2 -p 2222
192.168.68.2:2222 pubkey-hunt is running with 4673197 test keys
192.168.68.2:2222 pubkey-hunt completed 4673190/4673197 keys in 9h50m4s (132/s)
192.168.68.2:2222 pubkey-hunt accepted hunted half-auth for root with key ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDipNPRHvHknF6WLl7oEPoxxH7k13iKA/14yiWwOwHAUFg+1tl….


sshd[6530]: Connection closed by authenticating user root 192.168.68.1 [preauth]
```

single connection

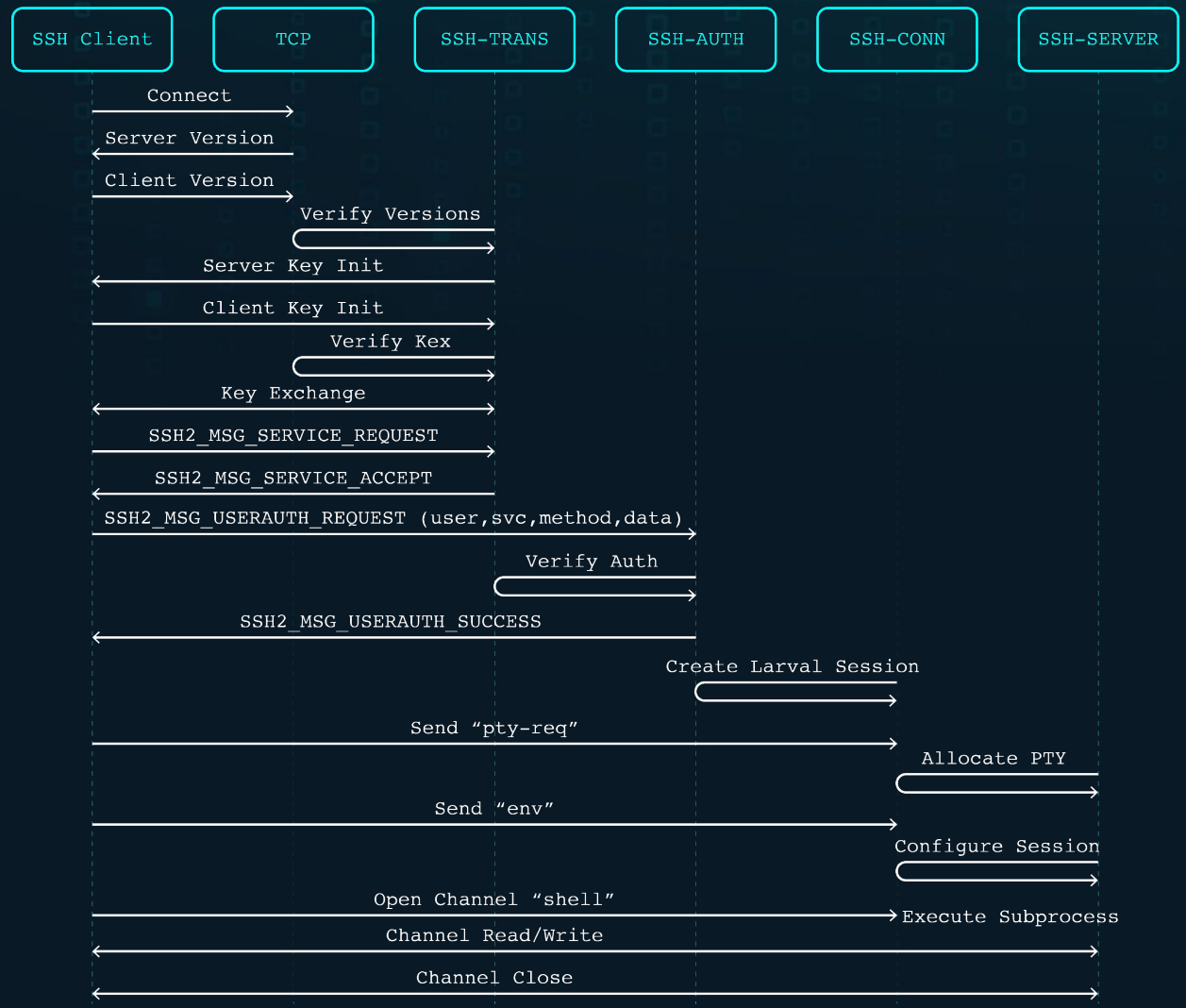
New Meets Old

(Authentication Bypass)

Secure shell uses a strict state engine

- Accepted client message types change as the connection moves through each state
- OpenSSH & Dropbear remap the table of command handlers on each state change
- Message IDs are clamped to specific allowed ranges by session state

```
SSH2_MSG_TRANSPORT_MIN                  1
SSH2_MSG_TRANSPORT_MAX                  49
SSH2_MSG_USERAUTH_MIN                   0
SSH2_MSG_USERAUTH_MAX                   79
SSH2_MSG_USERAUTH_PER_METHOD_MIN            60
SSH2_MSG_USERAUTH_PER_METHOD_MAX            79
SSH2_MSG_CONNECTION_MIN                 80
SSH2_MSG_CONNECTION_MAX                 127
SSH2_MSG_RESERVED_MIN                   128
SSH2_MSG_RESERVED_MAX                   191
SSH2_MSG_LOCAL_MIN                      192
SSH2_MSG_LOCAL_MAX                      255
SSH2_MSG_MIN                            1
SSH2_MSG_MAX                            255
```



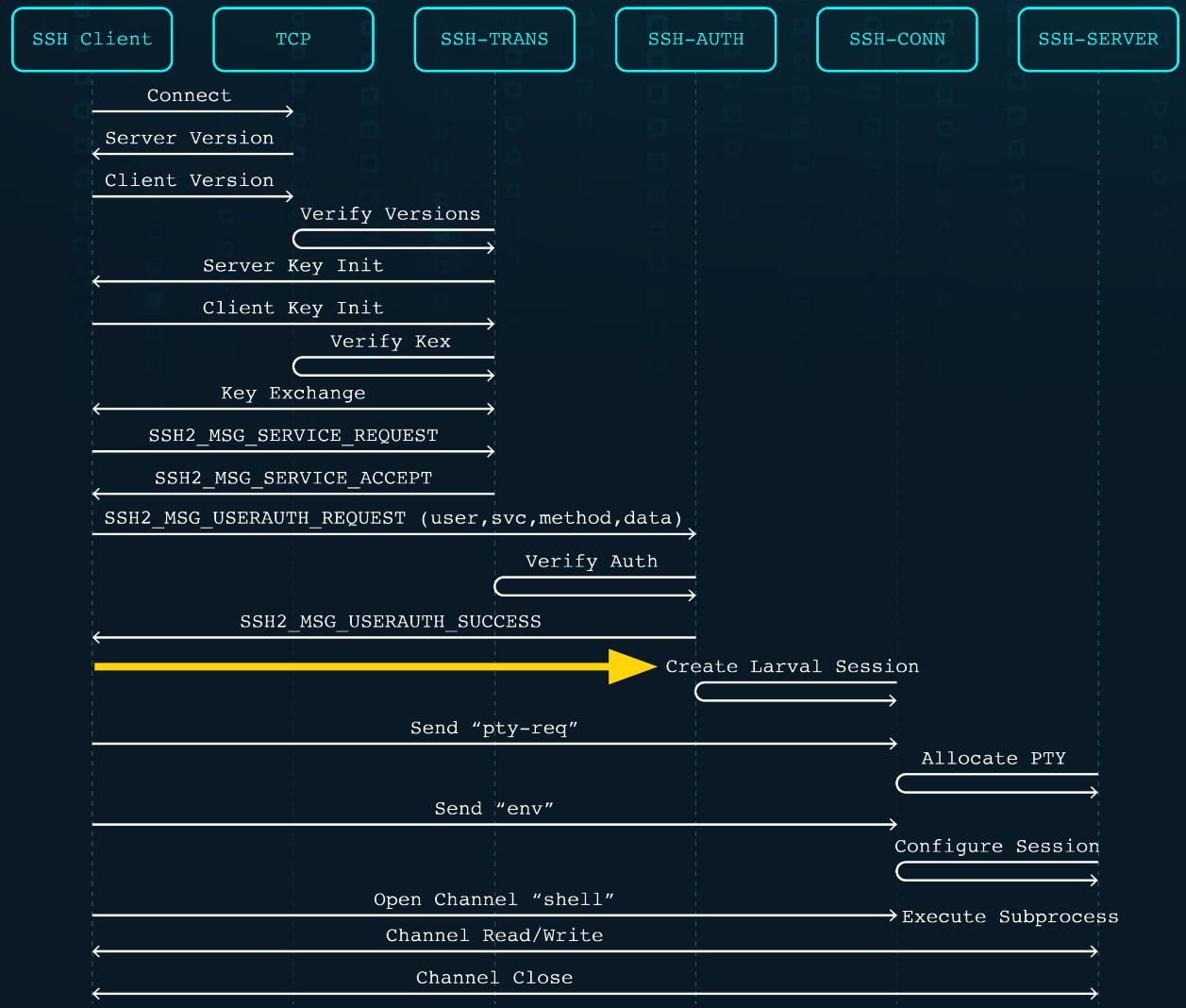
State transitions gone wrong (historic)

runZero

CVE-2018-10933

A bug in libssh where the server trusted a client-sent USERAUTH_SUCCESS message.

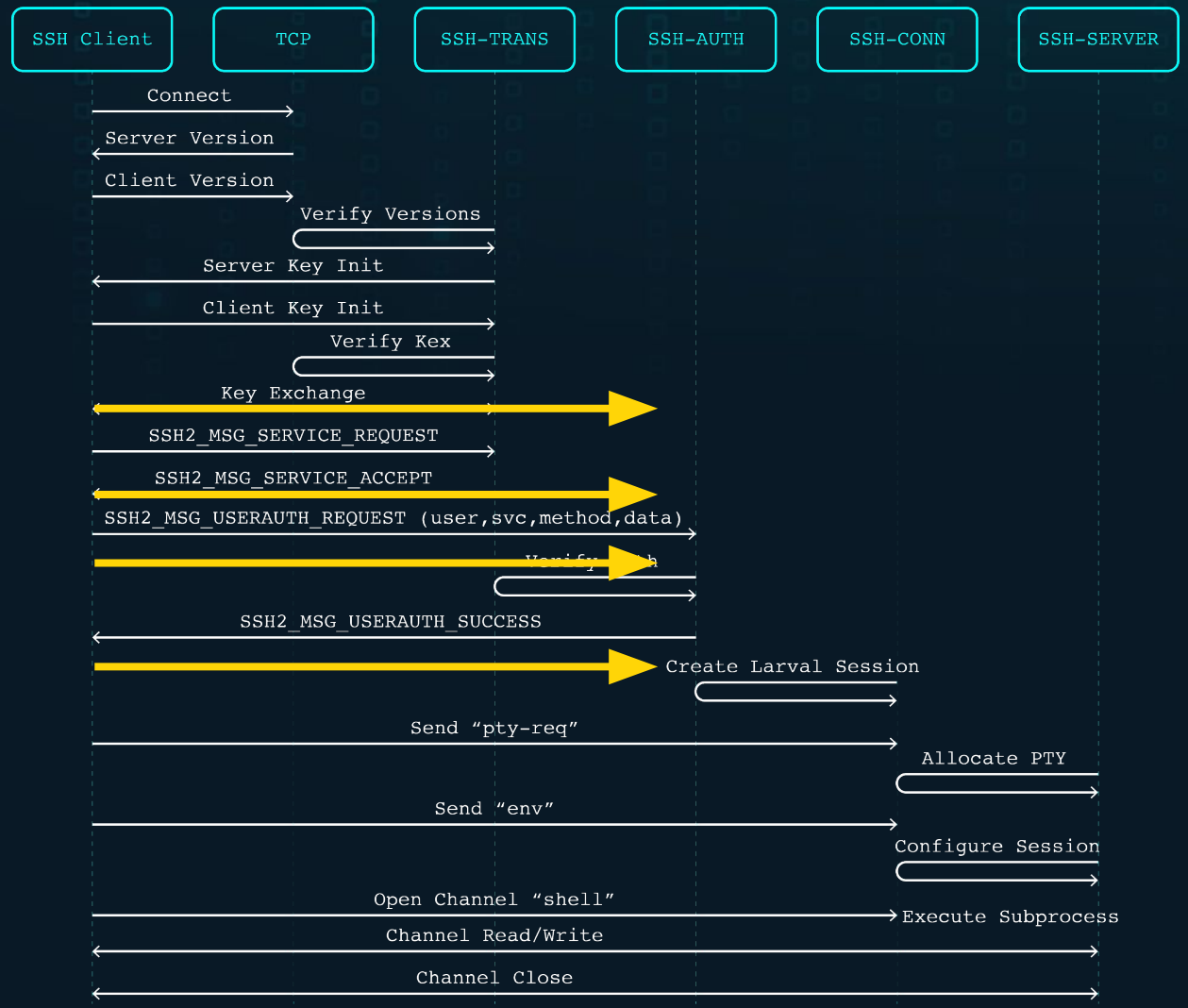
Metasploit support!



State transitions gone wrong (new)

What happens if we ask for a session at every possible state transition?

Free shells!

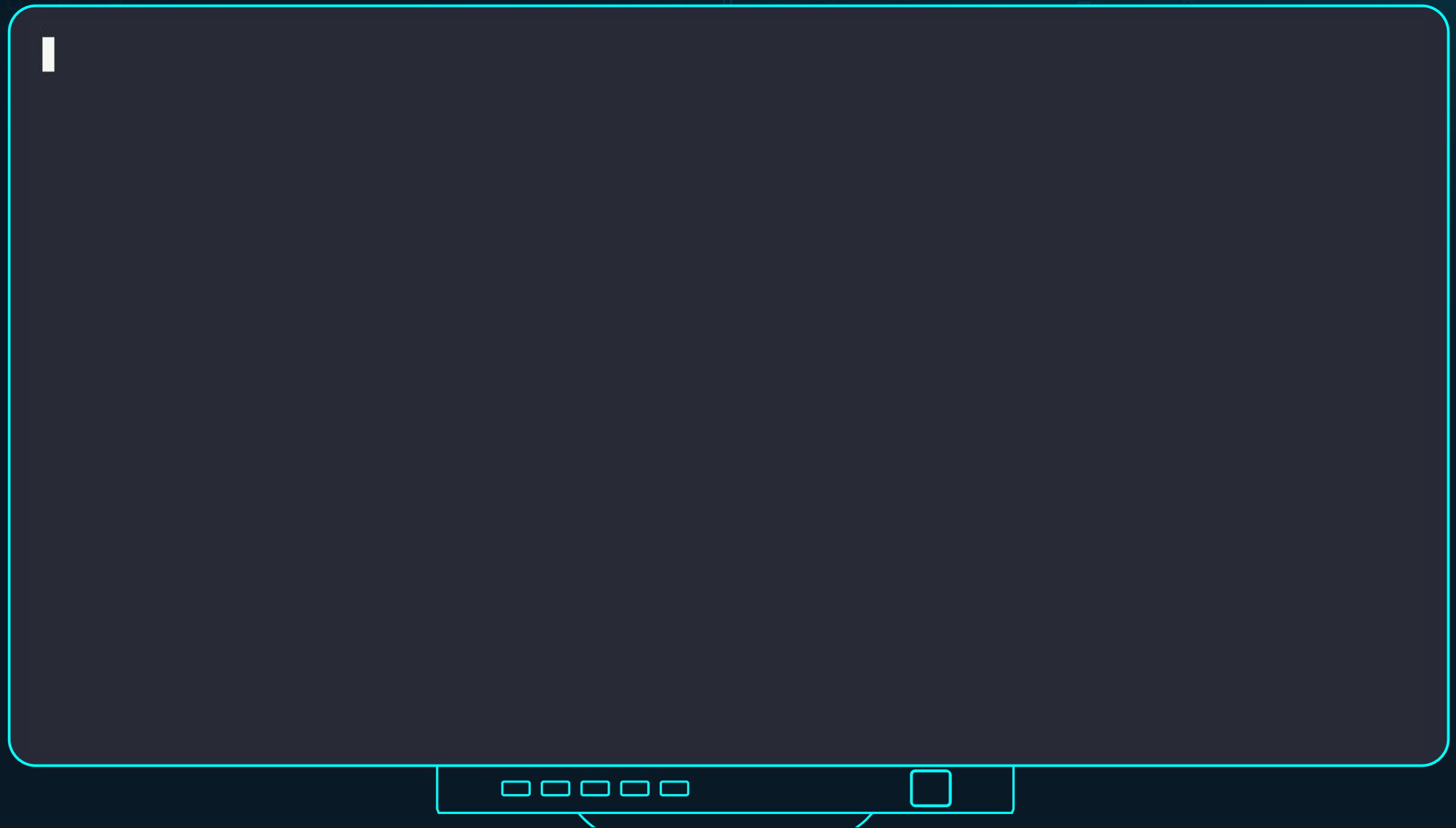


State transition vulnerabilities

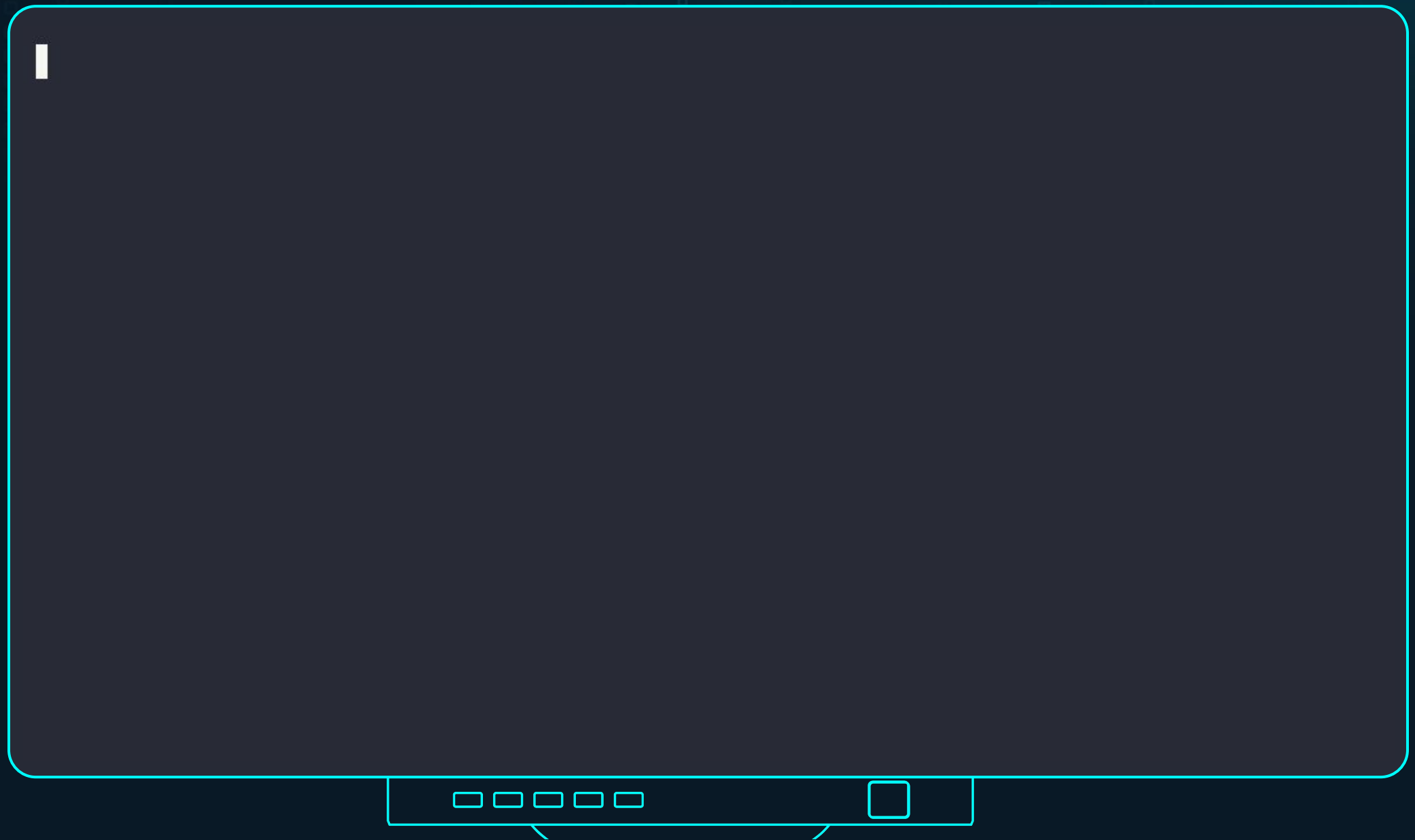
runZero

Product	Impact	Details
Digi TransPort WR Gateways	Remote CLI as SUPER	Authentication bypass due to uninitialized variable. Updates available for WR11, WR21, WR31, WR44R, WR44RR included in version 8.6.0.4. The Digi International product security team was great to work with (via Bugcrowd).
Realtek ADSL Routers	Remote CLI access as admin	Authentication bypass via skipping ssh-userauth. White-labeled by Netis, Neterbit, and many other vendors. Observed in firmware as recent as 2023.
Panasonic Ethernet Switches	Remote CLI access as admin	Authentication bypass via skipping auth “none” after the ssh-userauth sequence. Models include PN28080K, PN28240i, and likely others.

Neterbit NSL-224 authentication bypass



Digi TransPort authentication bypass



Post-session authentication is a bad idea

runZero

Various products allow ***none*** authentication & then implement interactive login in the session.

Dangerous due to the extensive post-auth attack surface of SSH.

Post-session capabilities	
shell	exec
pty-req	x11-req
subsystem	env
break	signal
agent-auth-req	window-change

Post-session authentication

runZero

```
root@              password:


Copyright (c) 2021 SonicWall, Inc.



Using username 'root'.
Password:
```

```
Please login:
```

```
Copyright (c) 2002 - 2013 Juniper Networks, Inc. All rights reserved.



Username:
```

Ruckus Wireless AP command injection

runZero

SSH auth *none* drops to an interactive login session

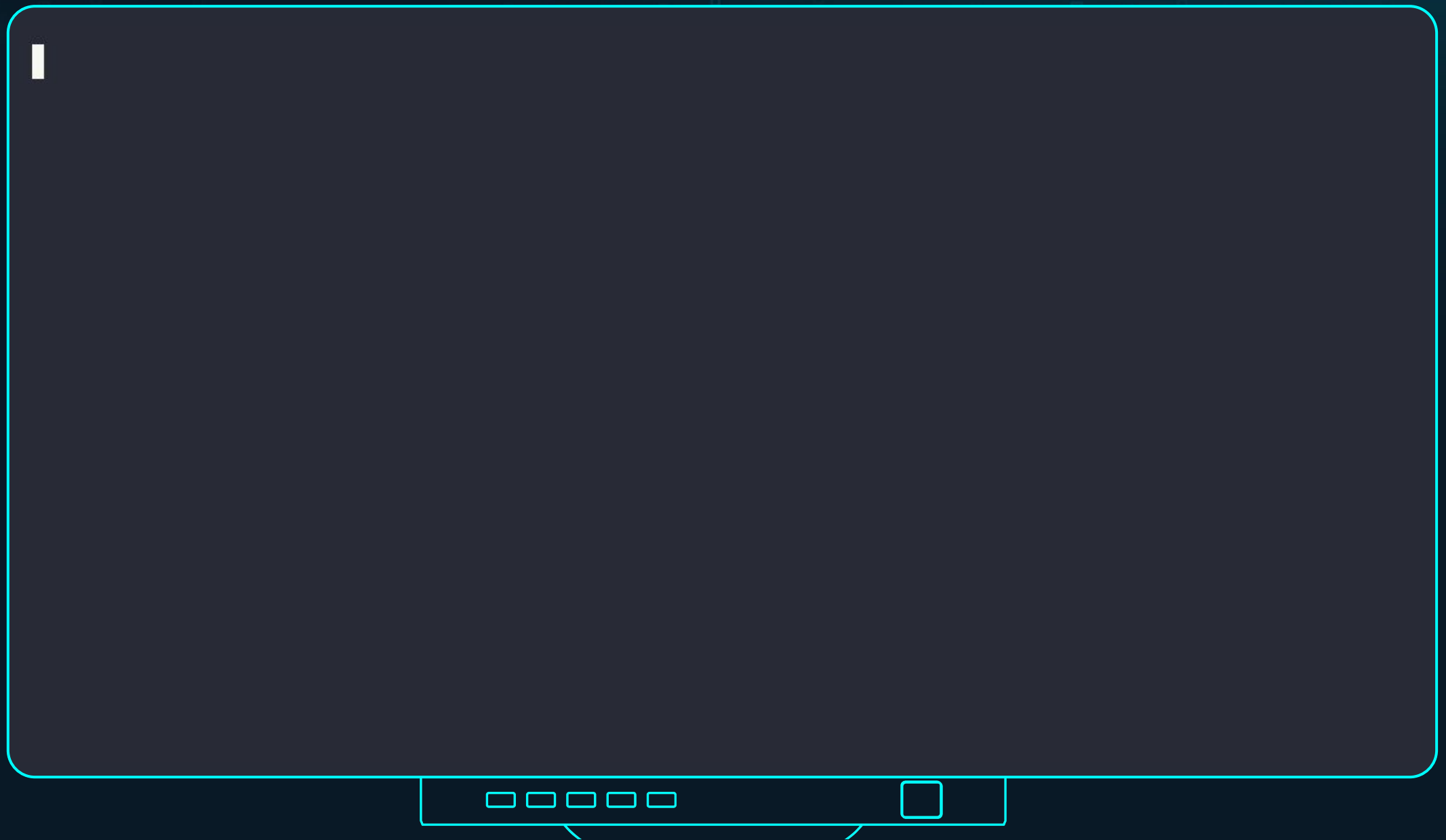
- The password input is passed into a shell without escapes

```
echo -n "$(echo pa55w0rd 1>&2)" | sha256sum
```

Fixed in firmware versions v5.2.1 (stable) & 6.2.1 (tech)

- Trivial root & still ~900 exposed on the internet
- No CVE, no security mention in the release notes
- Why did this bug live so long?

Ruckus Wireless AP command injection



Signal handling varies by service

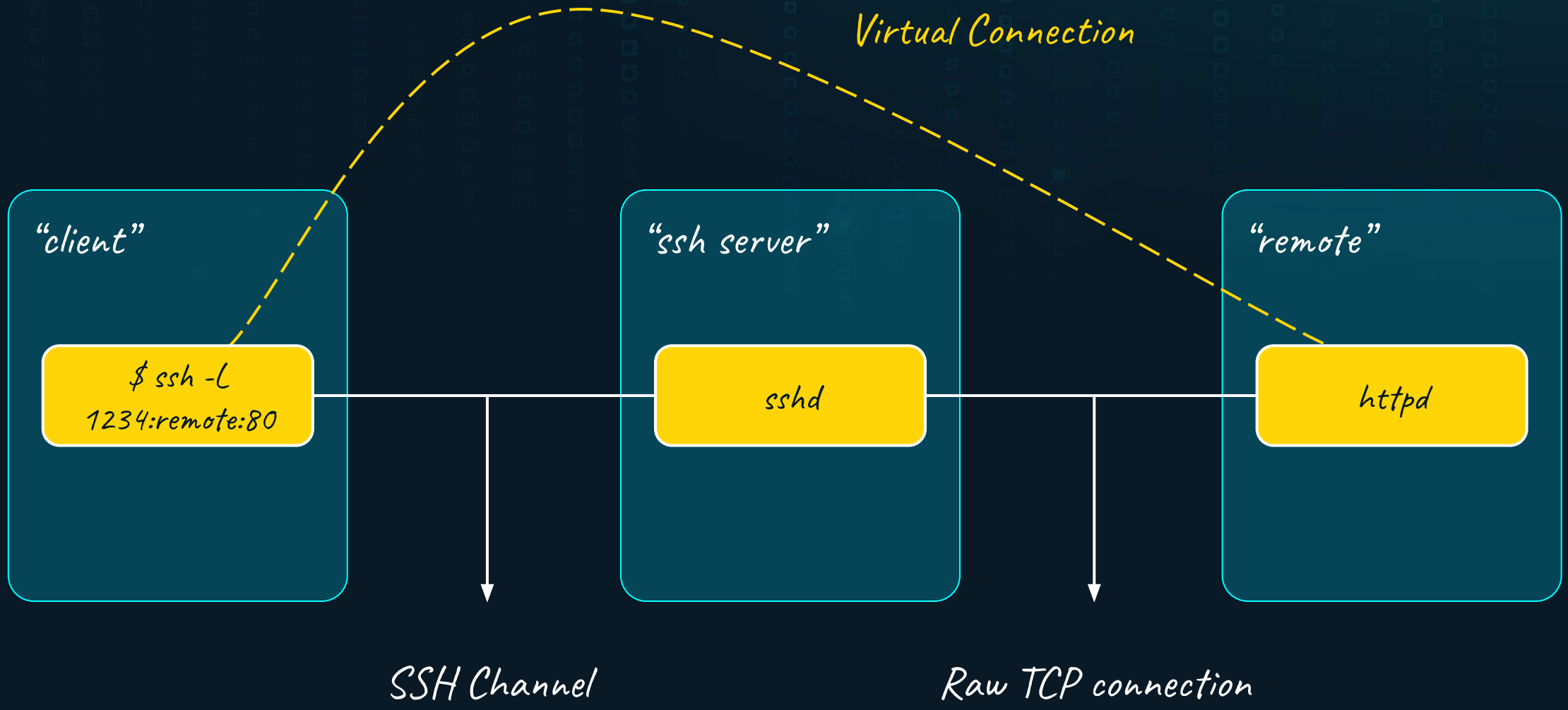
- OpenSSH restricts signals to relatively safe options
- Dropbear allows just about anything, even SEGV
- Signal-based attacks seem promising

```
Login:

sshamble> signal SEGV

Aiee, segfault! You should probably report this as a bug to the developer
```

Fun with Forwarding

SSH connection forwarding

runZero



Forwarding in restricted shells

runZero

Inadvertent forwarding in SSH is a common issue

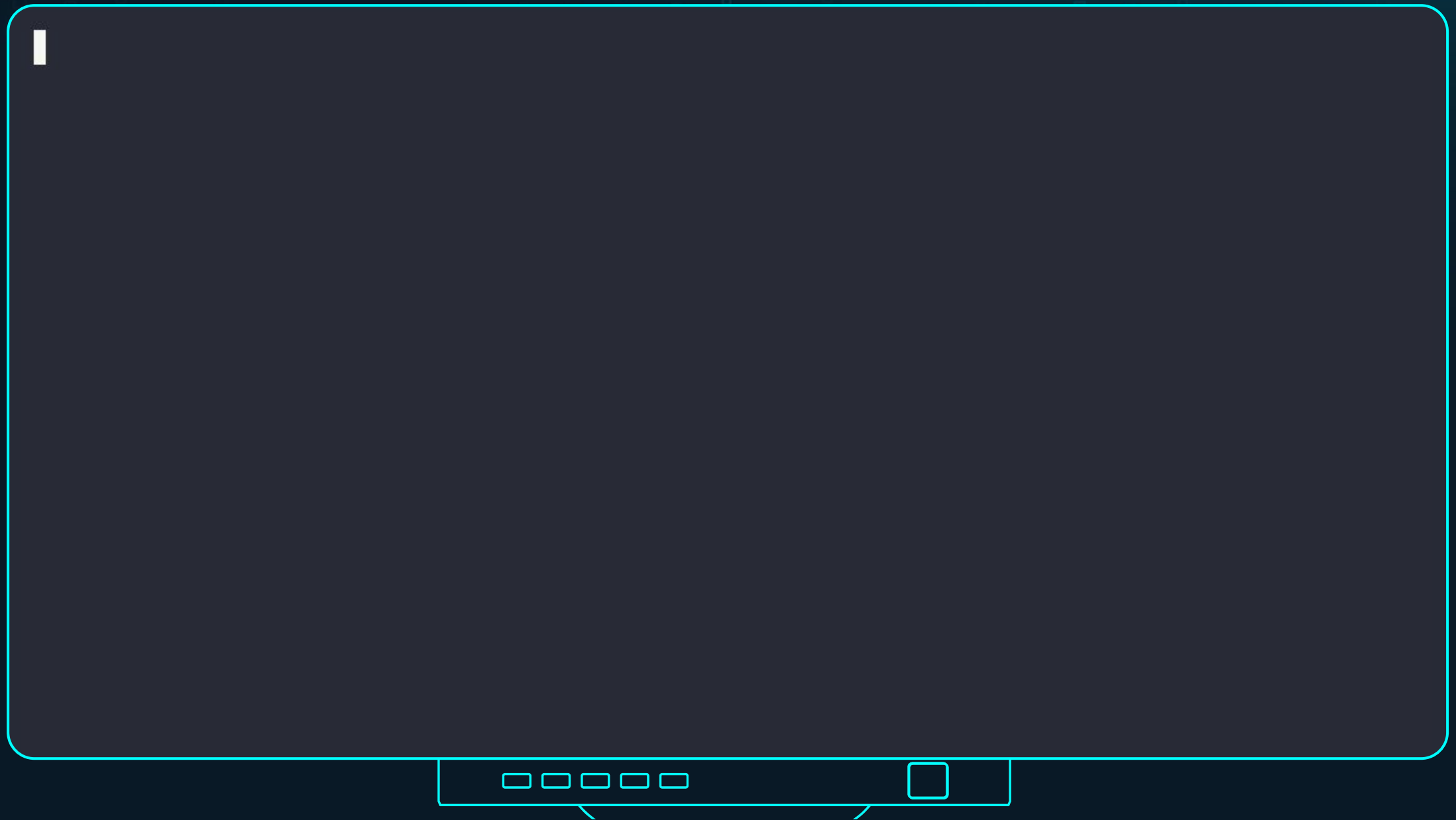
- → Network devices, virtual machines, & appliances
- → Can enable other attacks & bypass restrictions
- → Exposes localhost-bound daemons

Post-auth login enables unauthenticated attackers

- → Not super common, but we found some anyways
- → Requires testing a few destinations to evade ACLs

ION Networks Service Access Point

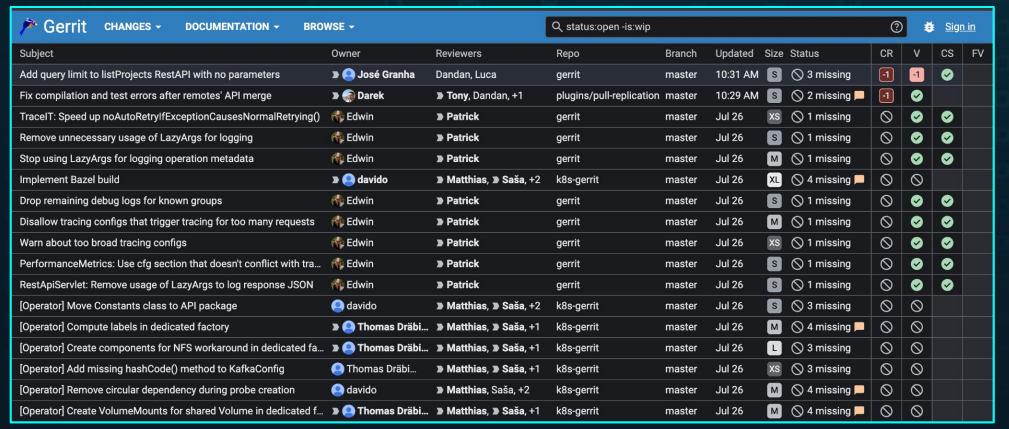
runZero

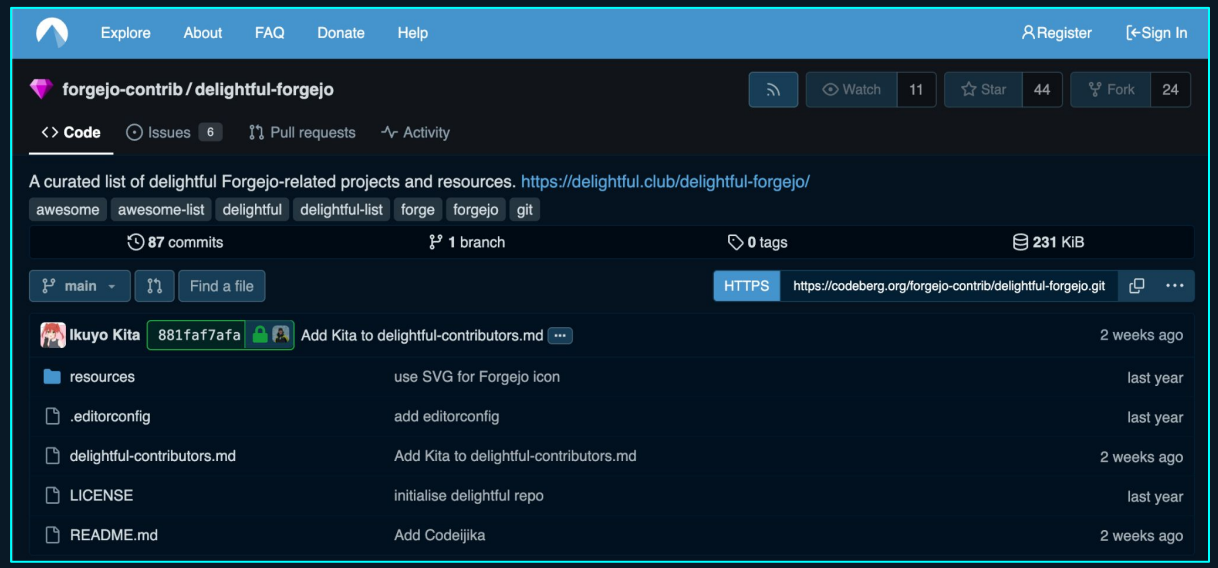


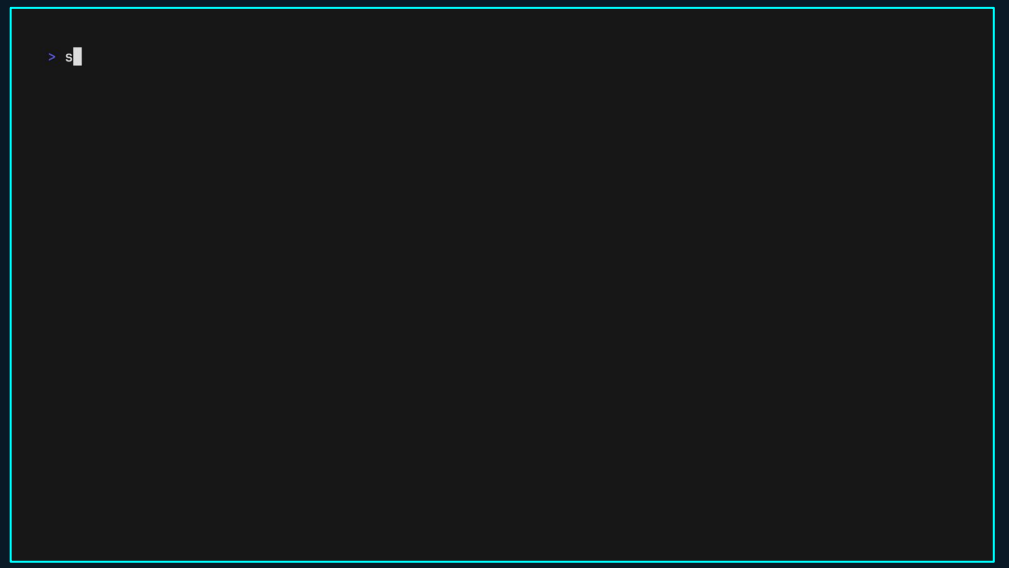
Checkout Git

Git-based code forges support SSH

- Services like GitHub, Gitlab, Bitbucket
- Projects like GOGS, Gitea, Forgejo, Gerrit
- Libraries like charmbracelet/ssh & Mina







Gitlab, Gitea, & Forgejo

runZero

- Environment control limited to **GIT_PROTOCOL**
- Git only parses the **version** parameter
- Usually safe, but bugs still exist
  - Go < 1.19.3 via CVE-2022-41716

```
GIT_PROTOCOL=version=2:\x00PATH=C:\Users\gitlab\repositories\rob
```

GOGS “env” command injection

runZero

GOGS was the first Go-based git forge

- Supports SSH “env”, but gets it terribly wrong

```
ExecCmd("env", fmt.Sprintf("%s=%s", env.Name, env.Value))
```

This does nothing, "env" doesn't set the parent env

- GOGS supports self-registration & ***env*** often supports **-S**
- Exploit with env `-SA=B touch /tmp/fun`
- No patch available, consider alternatives

* Independently discovered by Sonar Source (reported 2 days before us): CVE-2024-39930

SSH libraries & *env*: Apache Mina

runZero

Apache Mina is a Java package for SSH clients & servers

- Passes "env" variables to caller with no restrictions
- Callers (like Gerrit) **do** limit the environment
- JGit & friends don't spawn subprocesses

SSH libraries & *env*: Soft Serve

runZero

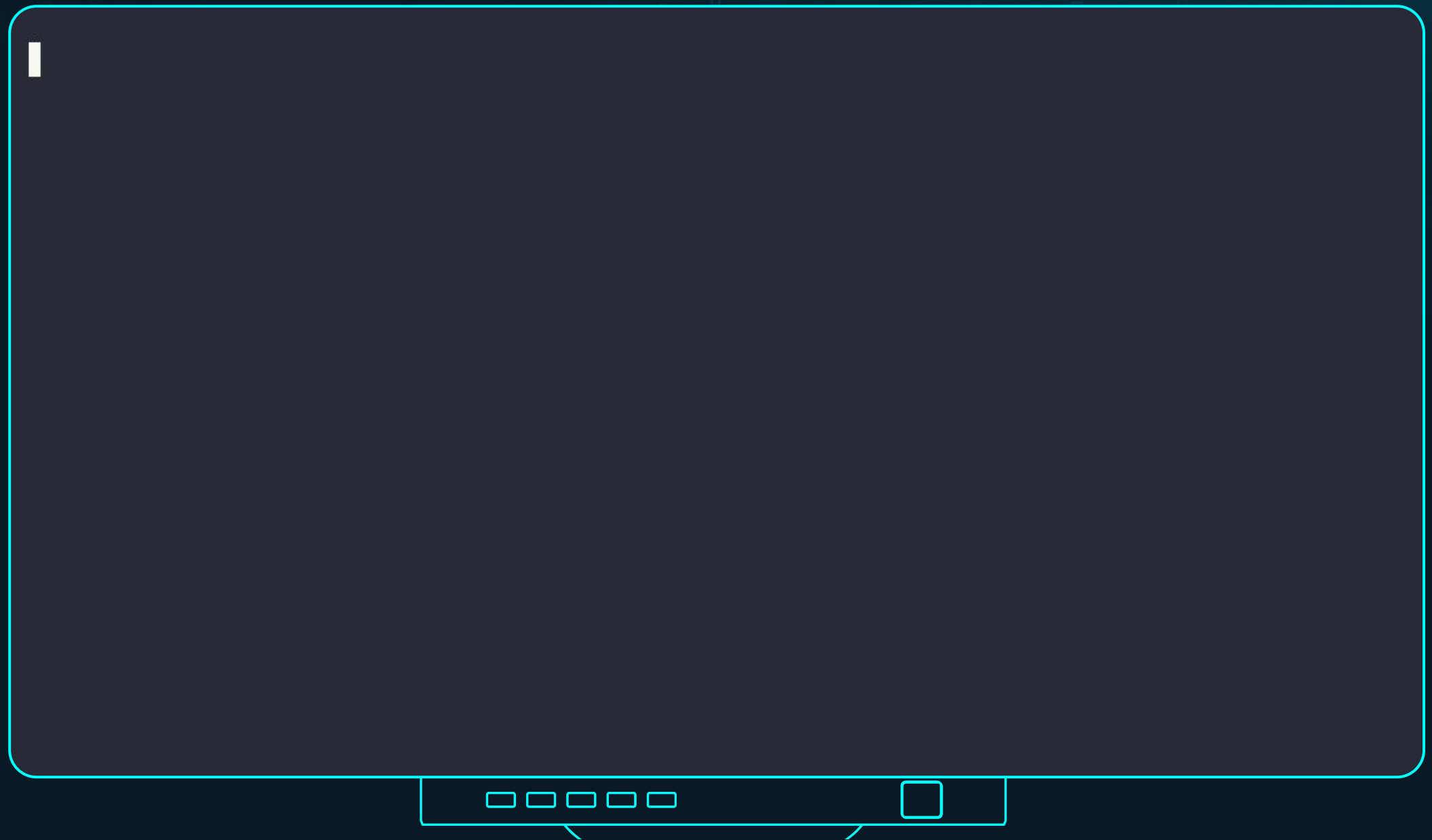
Soft Serve is a feature-full Git forge that provides a beautiful CLI

- → Uses charmbracelet/ssh (a gliderlabs/ssh fork)
- → Accepts all environment variables
- → Soft Serve passes these to Git
- → Combination is a remote shell

CVE-2024-41956

Remote Code Execution in Soft Serve

runZero



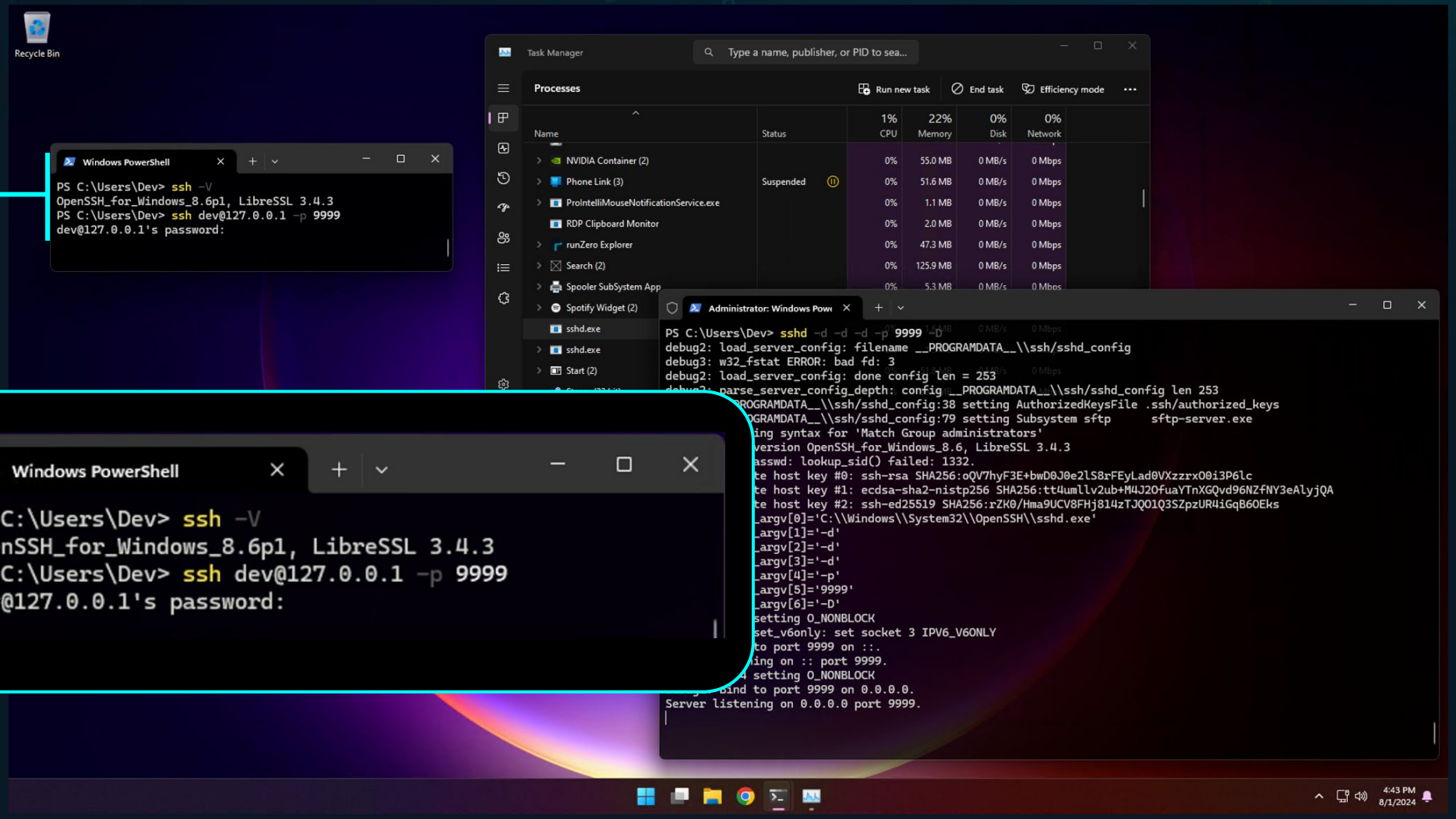
OpenSSH Fragmentation

OpenSSH divergence by platform

runZero

Name	Divergence	Notes
Apple macOS	Light	Changes are limited to macOS compatibility, support for the Keychain, the macOS PKCS helper, & endpoint event logging support.
Debian/Ubuntu Linux	Moderate	Systemd support & much more (36+ patches)
Red Hat Linux	Moderate	Systemd support & much more (~60 patches)
PKI-X SSH	Major	Forked in 2002 for X509 support, commonly found in networking gear and FIPS-compliant network appliances. Generally follows OpenSSH changes, but not exactly.
Microsoft Windows	Extreme	Over 350 files changed. Replaces fork with subprocesses, removes chroot support & log sanitization. Logs to Windows Events. Sends telemetry containing SSH-encrypted values. Password authentication uses Lsa* functions. Still hasn't fixed Terrapin. Not affected by regreSSHion.

OpenSSH for Windows



OpenSSH for Windows Telemetry

runZero

- OpenSSH for Windows sends detailed usage data to Microsoft
- Client & server versions, kex init parameters, auth methods

```
void send_ssh_version_telemetry (const char* ssh_version,
    const char* peer_version, const char* remote_protocol_error)
{
    TraceLoggingRegister (g_hProvider1);
    TraceLoggingWrite (
        g_hProvider1,
        "Startup",
        TelemetryPrivacyDataTag (PDT_ProductAndServiceUsage),
        TraceLoggingKeyword (MICROSOFT_KEYWORD_MEASURES),
        TraceLoggingString (ssh_version, "ourVersion"),
        TraceLoggingString (remote_protocol_error, "remoteProtocolError"),
        TraceLoggingString (peer_version, "peerVersion")
    );
    TraceLoggingUnregister (g_hProvider1);
}
```

compat/timingsafe_bcmp.c

runZero

```
int timingsafe_bcmp(const void *b1, const void *b2, size_t n) {
 const unsigned char *p1 = b1, *p2 = b2;
 int ret = 0;
 for (; n > 0; n--) {
   ret |= *p1++ ^ *p2++;
 }
 return (ret != 0);
}
```

A solid bit of code from DJM

- → Timing-safe
- → Efficient
- → Secure

compat/timingsafe_bcmp.c for Windows

runZero

```
int timingsafe_bcmp(const void *b1, const void *b2, size_t n) {
 const unsigned char *p1 = b1, *p2 = b2;
 int ret = 0;
 for (; n > 0; n--) {
#ifdef WINDOWS
   if (*p1 == '\r' && *(p1 + 1) == '\n' && *p2 == '\n')
     p1++;
#endif // WINDOWS
   ret |= *p1++ ^ *p2++;
 }
 return (ret != 0);
}
```

compat/timingsafe_bcmp.c for Windows

runZero

```
int timingsafe_bcmp(const void *b1, const void *b2, size_t n) {
 const unsigned char *p1 = b1, *p2 = b2;
 int ret = 0;
 for (; n > 0; n--) {
#ifdef WINDOWS
   if (*p1 == '\r' && *(p1 + 1) == '\n' && *p2 == '\n')
     p1++;
#endif // WINDOWS
   ret |= *p1++ ^ *p2++;
 }
 return (ret != 0);
}
```

Two lines, but so many bugs!

- → Not timing-safe
- → 1-byte OOB per \r
- → Unequal byte match

A critical function within OpenSSH

runZero

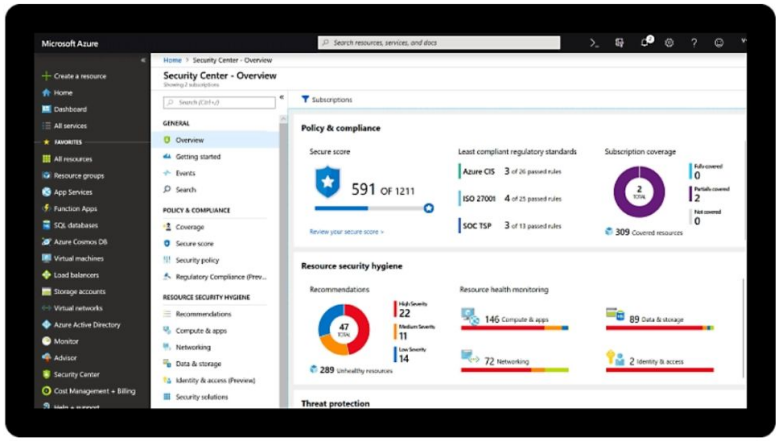
- MAC check on every SSH packet
- RSA signature verification
- SSH certificate comparison
- X11 cookie comparison
- chachapoly_crypt() MAC
- SSHFP DNS record checks
- SSH agent validation
- WebAuthn SK checks
- SSH keygen verification
- … & much more!

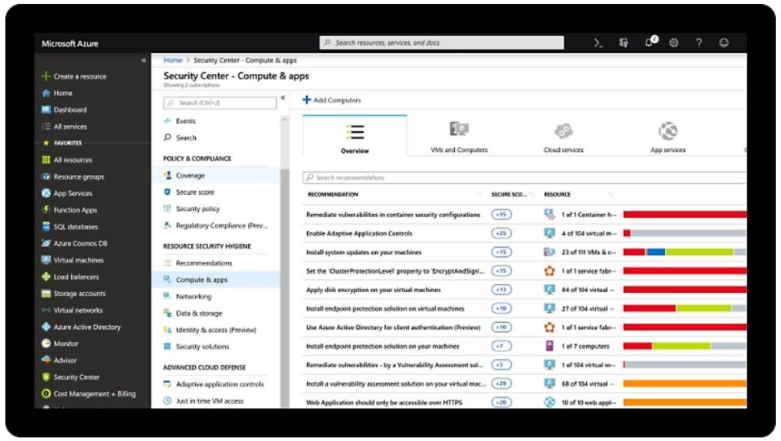
One of the most sensitive functions, but what can we do with it?

- Attacker has limited influence on the first argument
- Requires brute force to trigger in the MAC check
- Not obviously exploitable :(

Comprehensive security and compliance, built in

- Microsoft invests more than **$1 billion annually** on cybersecurity research and development.
- We employ more than **3,500 security experts** who are dedicated to data security and privacy.





Learn more about security on Azure

https://azure.microsoft.com/en-us/products/devops/server

Microsoft Security Response Center

runZero

> *Thank you again for submitting this issue to Microsoft. Although your report is valid, currently, MSRC prioritizes vulnerabilities that are assessed as "Important" or "Critical" severities for immediate servicing. After careful investigation, this case does not meet MSRC's current bar for immediate servicing because currently it appears to be theoretical due to no control over the first argument to the function & would require a brute force style attack to obtain a single byte of data. If you can prove remote reachability or the ability to leak information remotely, then please submit a new report & we are happy to investigate this further!*

Introducing SSHamble

runZero

- A research tool for SSH implementations
- Interesting attacks against authentication
- Post-session authentication attacks
- Pre-authentication state transitions
- Post-session enumeration
- Easy timing analysis

https://SSHamble.com

Built-in checks

runZero

bypass	auth=none	skip=auth	auth=success
	method=null	method=empty	skip=pubkey-any
publickey	pubkey-any	pubkey-any-half	user-key
	half-auth-limit	pubkey-hunt	—
password	pass-any	pass-empty	pass-null
	pass-user	pass-change-empty	pass-change-null
keyboard	kbd-any	kbd-empty	kbd-null
	kbd-user	—	—
gss-api	gss-any	—	—
userenum	timing-none	timing-pass	timing-pubkey
vulns	vuln-tcp-forward	vuln-generic-env	vuln-softserve-env
	vuln-gogs-env	vuln-ruckus-password-escape	—

Getting started

runZero

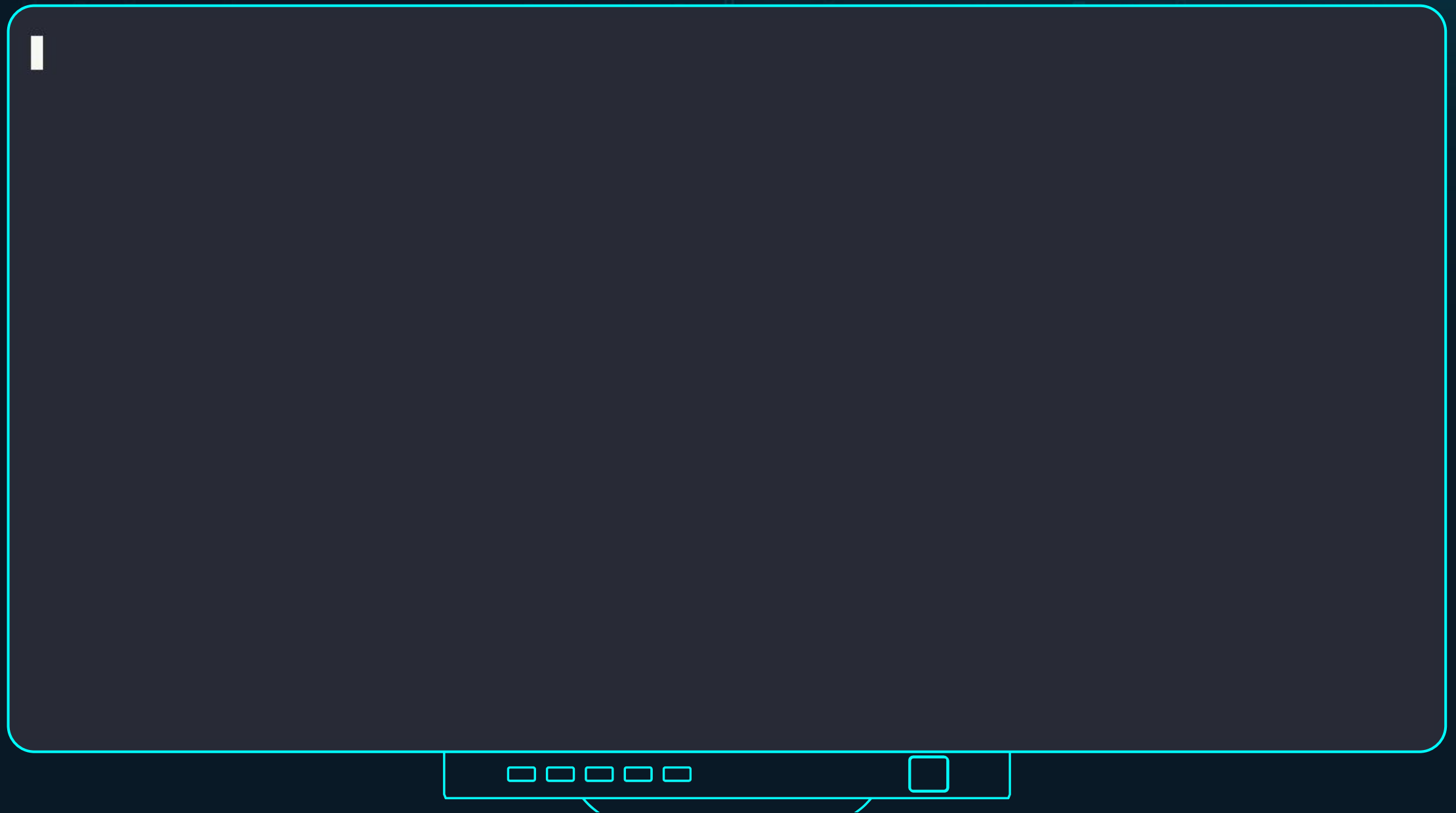
```
Start a network scan
$ sshamble scan -o results.json 192.168.0.0/24

Analyze the results
$ sshamble analyze -o output results.json

Specify ports, usernames, passwords, public keys, private keys, and more
$ sshamble scan -o results.json 192.168.0.0/24 \
    --users root,admin,4DGift,jenkins \
    --password-file copilot.txt \
    -p 22,2222 \
    --pubkey-hunt-file admin-keys.pub \

Open an interactive shell for sessions
$ sshamble scan -o results.json 192.168.0.0/24 \
    --interact first --interact-auto “pty,env LD_DEBUG=all,shell”
```

The interactive shell

runZero

```
Enter the sshamble shell with `^E`. Commands:

    exit                        - Exit the session (aliases 'quit' or '.')
    help                        - Show this help text (alias '?')
    env       a=1 b=2           - Set the specified environment variables (-w for wait mode)
    pty                         - Request a pty on the remote session (-w for wait mode)
    shell                       - Request the default shell on the session
    exec      cmd arg1 arg2     - Request non-interactive command on the session
    signal    sig1 sig2         - Send one or more signals to the subprocess
    tcp       host port         - Make a test connection to a TCP host & port
    unix      path              - Make a test connection to a Unix stream socket
    break     milliseconds      - Send a 'break' request to the service
    req       cmd arg1 arg2     - Send a custom SSH request to the service
    sub       subsystem         - Request a specific subsystem
    send      string            - Send string to the session
    sendb     string            - Send string to the session one byte at a time

sshamble>
```

Happy scanning!



Defending SSH

Client recommendations

runZero

Use public key authentication exclusively

- Separate GitHub/Launchpad keys from server administration keys
- Store your private key on a hardware token
- Switch to Ed25519 if you haven't already

If you use ssh agent forwarding, restrict destinations

- https://www.openssh.com/agent-restrict.html

Adjust configuration for LTS distro SSH clients

- Update ssh_config for OpenSSH 9.8+ Ciphers/MACs/KeyAlgs

Server recommendations (general)

runZero

Centralize SSH hostkey management

- → Collect server hostkeys & provide clients pre-approved known_hosts

Use public key authentication exclusively

- → Limit public key types to Ed25519 & RSA >= 2048

Limit resource usage by attackers

- → Enable `PerSourcePenalties` & set `PerSourceNetBlockSize`
- → Consider lowering `MaxStartups` & `MaxAuthTries`
- → Disable forwarding (TCP, Unix, Agent, X11) unless required

Adjust configuration for LTS distro SSH servers

- → Update sshd_config for OpenSSH 9.8+ Ciphers/MACs/KeyAlgs

Server recommendations (CA)

runZero

Configure a CA for server hostkeys

- → Create a CA, sign, & distribute hostkeys to each of your servers
- → Set `known_hosts` for clients: `@cert-authority *.domain.tld <CA.pub>`
- → CA hostkeys are backwards compatible (fallback to known_hosts)

Configure a CA for signing user keys

- → Sign user public keys with short-term expirations (using your tool of choice)
- → `ssh-keygen -s userCA -I user@example.com -n username -V +1h userkey.pub`

Consider mandating token-stored private keys

- → Enforce verification on servers with `PubkeyAuthOptions`
- → Require PIN with `verify-required` (vs `touch-required`)

Vendor recommendations

runZero

Build with OpenSSH wherever possible

- → Leverage OpenSSH 9.8p1+ for tons of great defensive features
- → Integrate with system authentication vs post-session

Ship clean firmware without static credentials

- → Prior to imaging, purge all host keys, known_hosts, & authorized_keys
- → Disable password authentication (or restrict to serial or console tty)

General hardening

- → Disable empty password auth & limit which users can authenticate
- → Disable all types of forwarding, set `ForceCommand` for shells

Conclusions

runZero

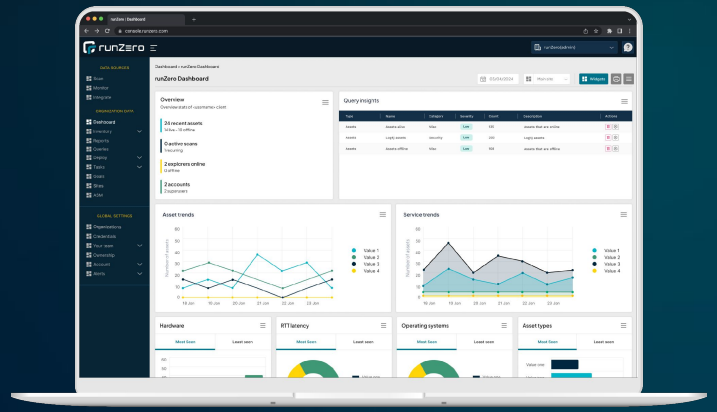
1. The secure shell is more critical than ever
2. Public key authentication is still leaky
3. OpenSSH is still your safest choice
4. Tons of issues in the periphery

black hat USA 2024

runZero

Thank you.

HD MOORE | ROB KING | AUGUST 7, 2024



runZero RESEARCH

runZero.com

research@runZero.com

SSHamble.com

References

- https://github.com/ssh-mitm/ssh-mitm
- https://ssh-comparison.quendi.de/comparison/hostkey.html
- https://words.filippo.io/ssh-whoami-filippo-io/
- https://github.com/badkeys/badkeys
- Metasploit: ssh_identify_pubkeys (2012)
- regreSSHion: https://www.qualys.com/2024/07/01/cve-2024-6387/regresshion.txt
- Terrapin: https://terrapin-attack.com/
- https://labs.watchtowr.com/auth-bypass-in-un-limited-scenarios-progress-moveit-transfer-cve-2024-5806/
- https://boehs.org/node/everything-i-know-about-the-xz-backdoor
- http://thetarpit.org/2018/shithub-2018-06
- https://helda.helsinki.fi/server/api/core/bitstreams/471f0ffe-2626-4d12-8725-2147232d849f/content
- https://github.blog/2023-03-23-we-updated-our-rsa-ssh-host-key/
- Kannisto, J., Harju, J. (2017). The Time Will Tell on You: Exploring Information Leaks in SSH Public Key Authentication. In: Yan, Z., Molva, R., Mazurczyk, W., Kantola, R. (eds) Network and System Security. NSS 2017. Lecture Notes in Computer Science(), vol 10394. Springer, Cham. https://doi.org/10.1007/978-3-319-64701-2_22
- West, J.C., Moore, T. (2022). Longitudinal Study of Internet-Facing OpenSSH Update Patterns. In: Hohlfeld, O., Moura, G., Pelsser, C. (eds) Passive and Active Measurement. PAM 2022. Lecture Notes in Computer Science, vol 13210. Springer, Cham. https://doi.org/10.1007/978-3-030-98785-5_30
- Neef, S. (2022). Source & result datasets for "Oh SSH-it, what's my fingerprint? A Large-Scale Analysis of SSH Host Key Fingerprint Verification Records in the DNS" [Data set]. Zenodo. https://doi.org/10.5281/zenodo.6993096