Headless Host Scanning

Jan Schaumann <jschauma@netmeister.org>

99CE 1DC7 770A C5A8 09A6 0DCD 66CE 4FE9 6F6B D3D7

@jschauma
Life Lessons: Pen testing can't solve everything.
Distance to Yahoo! ad in feet: 404
Construction cost: $357 million
Capacity: 41,915
Ballpark numbers

Distance to Yahoo! ad in feet: 30927

Construction cost: $1.5 billion

Capacity: 52,325
Life Lessons: Size matters
It all began some time in January of 2007...
Initial Problem Statement

“Hey, we need to know which hosts can’t deal with the new Daylight Saving Time.”
A Systems Administrator's Toolbox
A simple tool

/bin/sh

perl

python
#!/bin/sh
while read host; do
  ssh $host "env TZ=PST8PDT \ 
    perl -e '@t=localtime 1173682800; exit 1 - pop @t'"
done
Problem Statement

“Please run this script on these hosts.”
2nd Pass

#!/bin/sh
while read host; do
  scp script.sh $host:/tmp/
  ssh $host "/tmp/script.sh"
done
Mensch, nutz' doch ein Konfigurationsmanagementtool!
Configuration Management

Diagram:
- **New**
  - **Build**
- **Clean**
  - **Initialize**
- **Configured**
  - **Update**
  - **Entropy**
  - **Debug**
  - **Retire**
- **Unknown**
- **Off**
Configuration Management

Good at:
- keeping state
- ensuring flat text file properties

Not so good at:
- deploying applications
- executing (one-time) scripts

Terrible at:
- reporting output of individual commands or scripts
Life Lessons: Sturgeon's Law (1st adage)

"Nothing is always absolutely so."
Mensch, nutz' doch Deine Logaggregationsinfrastruktur!
Log Aggregation

- makes assumptions about message format
- frequently unreliable and/or asynchronous
- adds additional complexity
3rd Pass

#!/bin/sh
while read host; do
cat script.sh | ssh $host "bash -s"
done
3rd Pass

#!/bin/sh
while read host; do
    ssh $host "bash -s" < script.sh
done
“Run this set of arbitrary commands on our hosts. Yes, on all of them.”
Wait a second – *all* hosts?

```bash
$ wtf host
wtf: I don't know what host means!
$
```

- a computer connected to a network
- a virtual machine
- a container
  - jails, zones, ...
  - chroot
Sturgeon's Law (1st adage)

"Hosts" on the network

"Hosts" in asset database

"Hosts" in DNS
#!/bin/sh
while read host; do
  ssh $host "bash -s" < script.sh
done
Life Lessons: Causality (Duh!)

$ w
10:45AM up 51 days, 22:19, 11 users, load averages: 976.23, 618.17, 1020.25
$ time ls
     13.354 real   0.001 user   0.002 sys
$
Horizontal Scalability

Scanmaster

Scanslaves

Scanslaves

Targethosts
SSH Complications: `password prompts`

Interactive connections?

Password:

`$USER@hostname's password:`
SSH Complications: *password prompts*

Interactive connections begat password injectors.
SSH Complications: **password prompts**

```bash
proc respond { pw } {
    global pws
    if {![info exists pws($pw)]} {
        send_user " (autopw) "
        stty -echo
        expect_user -re "(.+)\n"
        stty echo
        set pws($pw) [
            exec perl -ne "print pack(q{u}, \$_)" << $expect_out(1,string)
        ]
    } else {
        send_user " (autopw'ed)"
    }
    log_user 0
    send -- "[exec perl -ne "print unpack(q{u}, \$_)" << $pws($pw)]\n"
    log_user 1
}

eval spawn -noecho $argv

expect {
    "password:" {
        respond system
        exp_continue
    }
}
```
SSH Complications: *assword prompts*

Key Authentication to the rescue!
Horizontal Scalability

Scanmaster

Scanslaves

Targethosts
SSH Complications: **Password prompts**

Key Authentication to the rescue?

- private keys need to be present on all hosts connecting to targets
- you need to provide passphrase for private keys at scan start

Enter passphrase for `/home/${USER}/.ssh/id_rsa`:
Enter passphrase for `/home/${USER}/.ssh/id_dsa`:
Enter passphrase for `/home/${USER}/.ssh/scankey`:
SSH Complications: *password prompts*

Key Authentication also uses password injectors.
SSH Complications: password prompts

SSH Agent Forwarding to the rescue!

- private keys stored in agent on single "safe" machine
- agent forwarding enabled to scan slaves
- everything's peachy!
SSH Agent Forwarding

Master

Slave

Targethosts

proxy agent
SSH Agent Forwarding

Master

Slave

Targethosts

proxy
agent
SSH Agent Forwarding

Master
SSH Agent Forwarding

11/19/12

Headless Host Scanning -- @jschauma
Scalability issues

Scanmaster

Scanslaves

Targethosts

Scanslaves

Scanslaves
Miscellaneous Scalability Issues

- doing hostname lookups for each connection is expensive
  - use IP addresses
- parsing `known_hosts` with several 100K entries is expensive
  - use `UserKnownHostsFile=/dev/null`
  - use `GlobalKnownHostsFile=/dev/null`
- hosts change ssh keys "frequently"
  - we need `StrictHostKeyChecking=no` :-(
- collecting even only a few lines from every single host can fill up your disk quickly
- ssh connections can hang "indefinitely" (=> `tkill(1)` cron scheduled killing of processes)
Yay, Open Source!

I just open-sourced Yahoo!'s "scanmaster" scripts: http://is.gd/kRLYyv http://is.gd/sdNNvP - enjoy! (Bug reports and fixes welcome.) #ssh
Scanmaster Scripts

- checkhosts(1)
  - run given script on all hosts in input via ssh(1) (in background)
- scanhosts(1)
  - invoke checkhosts(1) against split input (again in background)
  - defaults to 150 invocations of checkhosts(1) in parallel
- scanslave(1)
  - starts ssh-agent(1) and adds keys
  - adds tkill(1) crontab
  - invoke scanhosts(1) against split input
- scanmaster(1)
  - splits input into number of scanslaves chunks
  - ssh to each scanslave and kicks off scanslave(1) in screen session
  - aggregates and post-processes results
Scanmaster Scripts

$ wc -l *
   330 checkhosts
   259 scanhosts
   325 scanmaster
   179 scanslave
1093 total

$ file *
checkhosts: Bourne shell script text executable
scanhosts: Bourne shell script text executable
scanmaster: Bourne shell script text executable
scanslave: Bourne shell script text executable
$
"Every item on the menu at Taco Bell is just a different configuration of roughly eight ingredients. With this simple periodic table of meat and produce, the company pulled down $1.9 billion last year."

Miscellaneous Security Issues

- ssh-agent exposes our keys to target hosts
  - use ForwardAgent=no, ClearAllForwardings=yes

- ssh-agent on scanslaves exposes our privkeys to anybody with super-user privileges on the scanslaves
  - use a scan-only key with from= and various no- restrictions

- password injector will respond to anything asking for your password
  - uhm... don't use password auth?

- use of a "real" user is stupid
Going topless. T.O.P.L.E.S.S. ✓

Headless User Requirement #1:
- selective access (via ssh keys)

Headless User Requirement #2:
- a restricted shell

Headless User Requirement #3:
- an un-restricted shell

Headless User Requirement #2 and #3:
- allow arbitrary commands, provided they were sanctioned by somebody we trust
Windows PowerShell
Windows PowerShell Execution Policy

- **Restricted**
  - No scripts can be run. Windows PowerShell can be used only in interactive mode.

- **Unrestricted**
  - No restrictions; all Windows PowerShell scripts can be run.

- **RemoteSigned**
  - Downloaded scripts must be signed by a trusted publisher before they can be run.

- **AllSigned**
  - Only scripts signed by a trusted publisher can be run.
The (TacoBell) Sigsh

NAME

sigsh -- a signature verifying shell

SYNOPSIS

sigsh [-c certs] [-x] [-p prog]

DESCRIPTION

sigsh is a non-interactive, signature requiring and verifying command interpreter. More accurately, it is a signature verification wrapper around a given shell. It reads input in PKCS#7 format from standard in, verifies the signature and, if the signature matches, pipes the decoded input into the command interpreter.
Yay, Open Source!

Doing paperwork to open source 501 lines of code, comments and docs wrapping a single openssl command feels silly. #publicdomain'd be nice.
The (TacoBell) Sigsh

```bash
$ find . \( -type f -a ! -path '*.svn/*' \) | xargs wc -l
   50 ./test/sigsh.test.pl
   36 ./certs/Makefile
   5 ./certs/README
121 ./doc/sigsh.1
  17 ./doc/Makefile
176 ./src/sigsh.sh
   12 ./README
   29 ./LICENSE
 446 total
$
```
Yay, Open Source!

@jschauma
Jan Schaumann

I just open sourced Yahoo!'s "sigsh", a signature verifying command-interpreter.
http://is.gd/rE1HYP http://is.gd/FS6U6o

9 Feb via very-simple-tweet  Favorite  Reply  Delete
The (TacoBell) Sigsh

$ egrep -v "^[ ]*#" sigsh.sh | wc -l
   93

$
The (TacoBell) Sigsh

EXAMPLES

The following examples illustrate possible usage of this tool. To execute the commands in the file 'script.bash':

```
openssl smime -sign -nodetach -signer mycert.pem \
    -inkey mykey.pem -in script.bash -outform pem | sigsh
```

To execute the perl code contained in the signed PKCS#7 file 'code.pem':

```
sigsh -p /usr/bin/perl <code.pem
```
The Good, the Bad and the Ugly
The Good

- private cert need not be present on scanslave hosts
- signing party need not be executing party
- if you grant passwordless sudo to the headless user, you can do *
- certs can expire => we have built-in time-bomb powers!
- multiple people can have signing-power independent of each other

- we can scan over XXXK "hosts" in approximately 15 minutes
- we have the flexibility to collect all sorts of interesting data
Scanning over 400 hosts/second

[...]
16:09:14: 0 / 30 done - waiting...
16:10:16: 0 / 30 done - waiting...
16:11:18: 1 / 30 done - waiting...
16:12:20: 6 / 30 done - waiting...
16:13:22: 20 / 30 done - waiting...
16:14:24: 27 / 30 done - waiting...
16:15:25: 29 / 30 done - waiting...
16:16:27: 29 / 30 done - waiting...
16:17:30: 30 / 30 done
16:17:30: Scanned XXXXXX hosts in XX minutes and 32 seconds.
Aggregating data...
The Bad

- this is still a TacoBell sigsh, not a powershell

- we have not solved the ssh-key access problem
  (though a leaked ssh-key cannot allow arbitrary command execution, only execution of previously approved commands)

- multiple people can have signing-power independent of each other
Life Lessons: Pareto's Principle

The "trivial many"

80% of time expended

20% of results

The "vital few"

80% of results

20% of time
Life Lessons: Milton Friedman was right.

There's nothing so permanent as a temporary government program solution.
The Ugly

- Only ~80% of our hosts have sigsh => we need to combine headless with "headful" scans. Thus, we currently:
  - scan headlessly (80%, in <15 minutes)
  - scan with regular user and dedicated scankey (11%)
  - scan with regular user and password, twice (7%)
  - scan with regular user and old password
  - scan with regular user and older password
  - scan with regular user and even older password

- We still (have to) trust the binaries on the remote host.
  - no veriexec, TPE, ...
Life Lessons

- Pen testing can't solve everything.
- Size matters.
- "Nothing is always absolutely so." (Sturgeon's Law)
  - "90% of everything is crud." (2nd adage; freebie)
- Causality (Duh!)
- Mmm, mmm TacoBell!
- 80% of the effects come from 20% of the causes. (Pareto Principle)
- There's nothing so permanent as a temporary solution.

- The worst form of on-demand data collection except all the others that have been tried.
Information

- http://www.netmeister.org/apps/scanmaster/
- http://www.netmeister.org/apps/sigsh/
- jschauma@netmeister.org
- @jschauma

This talk in 140 chars or less:

*Scanning 100s of K hosts headlessly using a signature verifying shell is awful. Still better than most alternatives. That is all.*