BLESS: Better Security and Ops for SSH Access

Bryan D. Payne, Director of Product Security
June 2017

NETFLIX


Hacked By #GOP
Warning :
We've already warned you, and this is just a beginning.
We continue till our request be met.
We've obtained all your internal data including your secrets and top
If you don't obey us, we'll release data shown below to the world.
Determine what will you do till November the 24th, 11:00 PM(GMT
Data Link :
https://www.sonypicturesstockfootage.com/SPEData.zip
http://dmiplaewh36.spe.sony.com/SPEData.zip
http://www.ntcnt.ru/SPEData.zip
http://www.thammasatpress.com/SPEData.zip
http://moodle.universidadebematech.com.br/SPEData.zip

```
list2.txt:Sony - Workday sFTP 20130606 private key.ppk
list2.txt:Sony - Workday sFTP private key.ppk
```

".ppk" (private key files):

```
list2.txt:id_dsa.ppk
list2.txt:audible_magic_sftp_private_key.ppk
list2.txt:id_dsa.ppk
list2.txt:audible_magic_sftp_private_key.ppk
list2.txt:sonykey.ppk
list2.txt:drondilla.ppk
list2.txt:id_dsa.ppk
list2.txt:rjacinto.ppk
list2.txt:id_dsa.ppk
list2.txt:audible_magic_sftp_private_key.ppk
list2.txt:rloperena.ppk
list2.txt:lcheng.ppk
list2.txt:private.ppk
list2.txt:kdedo.ppk
list2.txt:AkamaiPrivateKey.ppk
list2.txt:dnelson.ppk
list2.txt:tderose.ppk
list2.txt:tderose.ppk
list2.txt:private_key.ppk
list2.txt:Sony - Workday sFTP 20130606 private key.ppk
list2.txt:Sony - Workday sFTP 20130606 private key.ppk
list2.txt:ADP SSH Private Key - Old.ppk
list2.txt:ADP SSH Private Key.ppk
```

Large Database of Device Certificates, SSH Keys Published

Written by **Dennis Fisher** on September 9, 2016 in **Authentication**, **Device Security**

Let's say you're a manufacturer of embedded device, maybe routers or wireless access points. Cool. And let's also say that you want to offer encrypted connections to those devices. Great. So you grab a server certificate online, throw it in the device's firmware and ship it. Not cool at all.

Note: This thread has been updated with an official response from Sangoma Technologies, Inc.

This could only potentially affect you if your "*organization has one or more deployments where you previously provided Sangoma with either SSH or Web GUI credentials, so that our support team would have easier access to your systems, when you request our help in future support calls*"

It has nothing to do with commercial module purchases nor commercial module usage.

If you bought a commercial module from Sangoma/Schmooze there is nothing we store to get in to the system the module is purchased from/for. The only information this response is in response to is the information listed in the statement linked to above.

Learning from the Expedia Heist

When your IT admin is the root cause of a security breach

But—centralized management is also the most effective means of lateral movement for an adversary. APT groups love domain admin or a shared SSH credential, like a fleetwide `root` SSH key, shared local admin password, or maybe a AWS access key with `AdminAccess` permissions.

Post by Ryan McGeehan

Security

16

Hackers break into FreeBSD with stolen SSH key

Not many servers dead

By John Leyden 20 Nov 2012 at 09:58

SHARE ▼

Hackers broke into two FreeBSD project servers using an SSH authentication key* and login credentials that appear to have been stolen from a developer, it has emerged.

Stolen SSH keys used for attacks

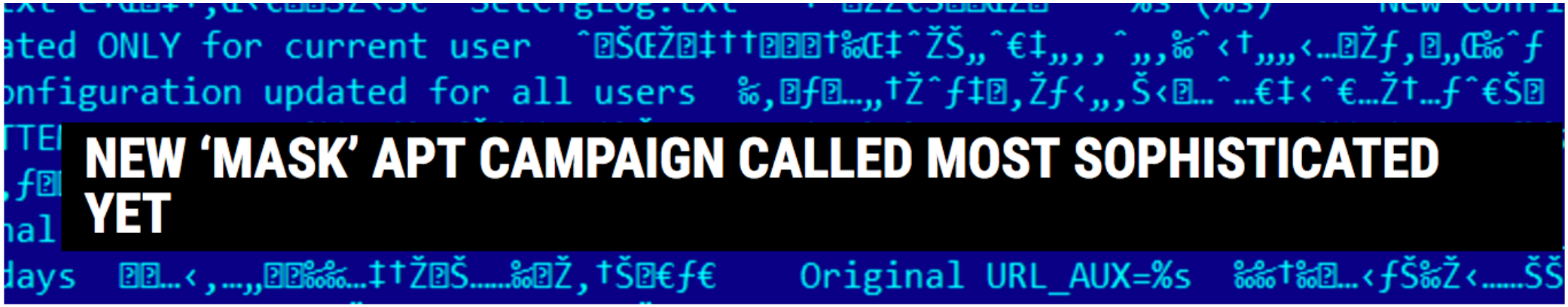
By Shaun Nichols
Aug 28 2008
4:01PM

Linux keys harvested by hackers.

Security experts are warning of a new series of Linux attacks that use stolen Secure Shell (SSH) keys.

The SSH protocol is used as a system for securely communicating between networked machines. The system was first designed as a replacement for the less-secure Telnet protocol.

The attack is part of a malware rootkit known as Phalanx2. According to an advisory from the US Computer Emergency Response Team (US-CERT,) the rootkit is a derivation of an older piece of malware and stores itself in a directory known as " /etc/khubd.p2/" which can only be accessed through the "cd" command.

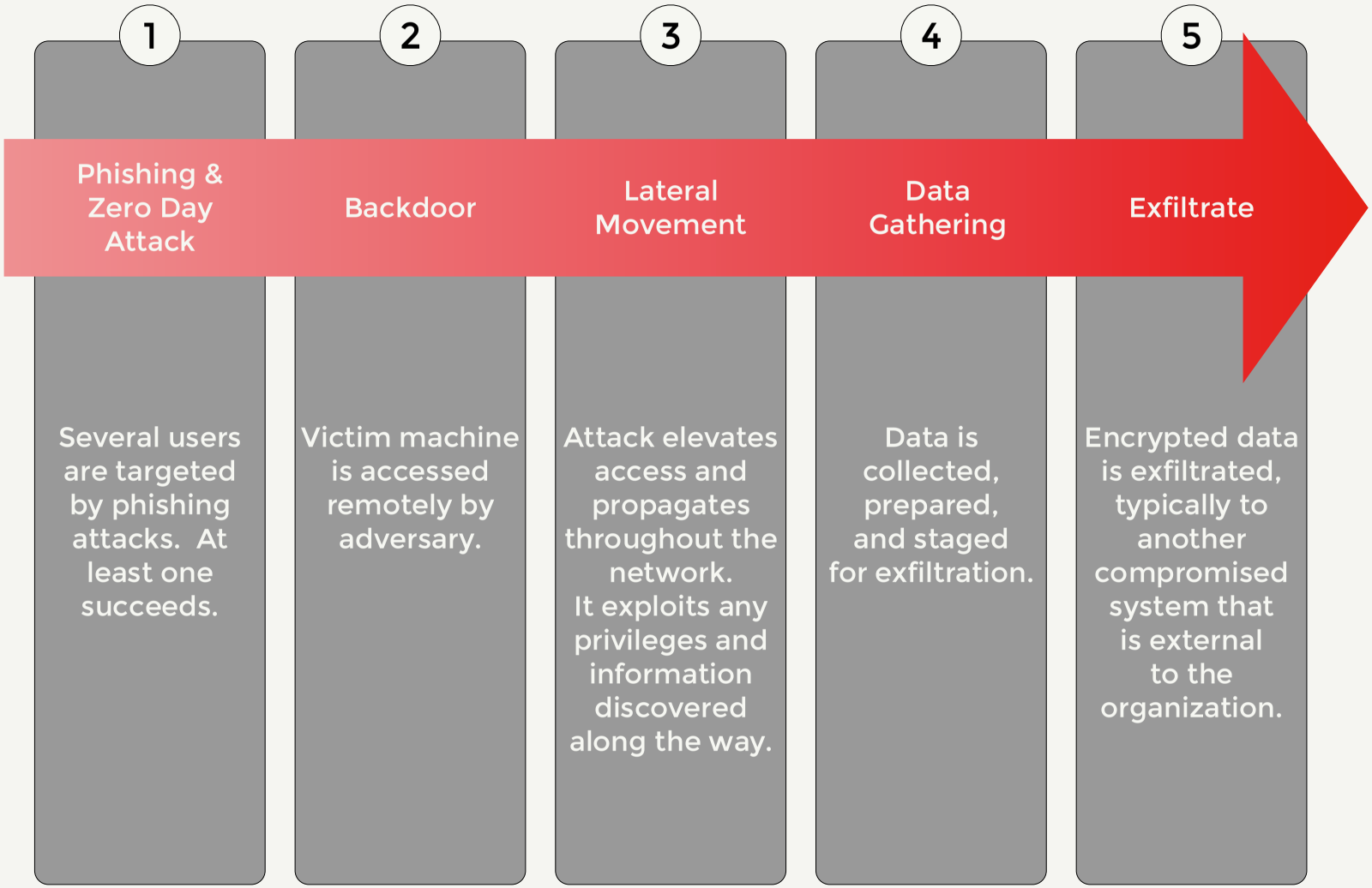


NEW 'MASK' APT CAMPAIGN CALLED MOST SOPHISTICATED YET

by Dennis Fisher Follow @dennisf

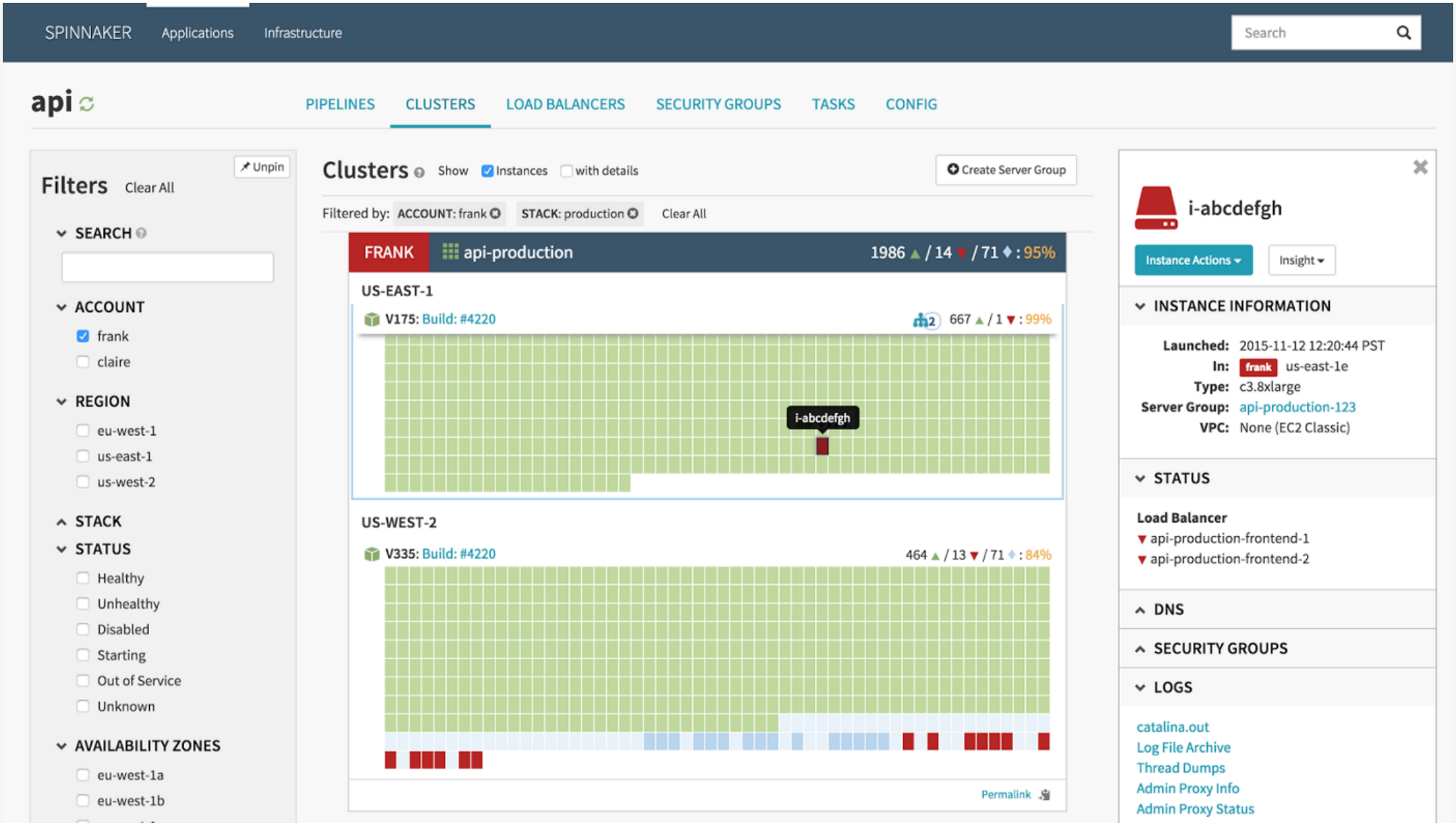
February 10, 2014 , 1:03 pm

PUNTA CANA–A group of high-level, nation-state attackers has been targeting government agencies, embassies, diplomatic offices and energy companies with a cyber-espionage campaign for more than five years that researchers say is the most sophisticated APT operation they've seen to date. The attack, dubbed the Mask, or "Careto" (Spanish for "Ugly Face" or "Mask") includes a number of unique components and functionality and the group behind it has been stealing sensitive data such as encryption and SSH keys and wiping and deleting other data on targeted machines.



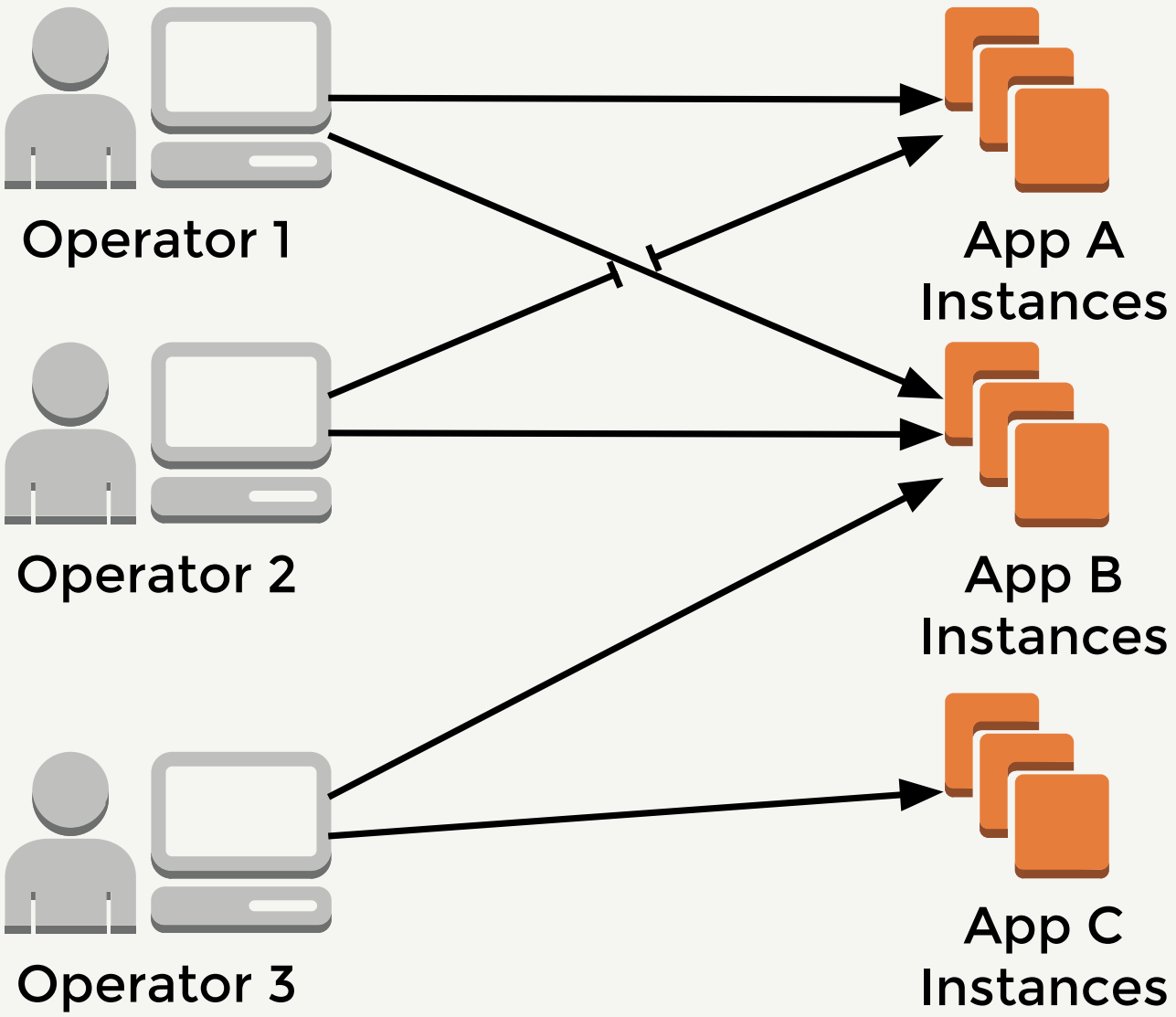
Adapted from https://blogs.rsa.com/anatomy-of-an-attack/.

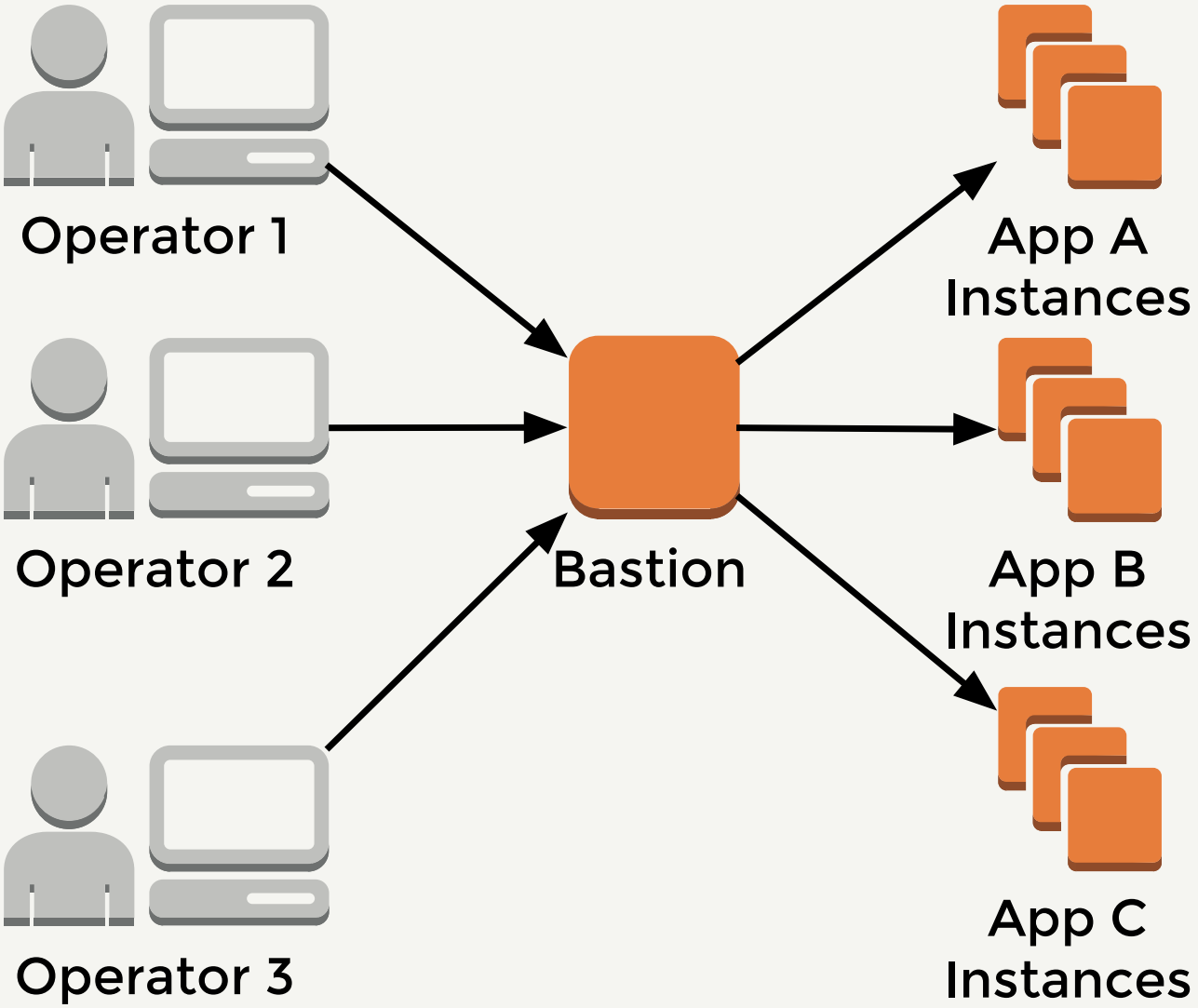
What's the Problem?


SPINNAKER
Applications
Infrastructure
Search
api
PIPELINES
CLUSTERS
LOAD BALANCERS
SECURITY GROUPS
TASKS
CONFIG
Filters
Clear All
Unpin
SEARCH
ACCOUNT
frank
claire
REGION
eu-west-1
us-east-1
us-west-2
STACK
STATUS
Healthy
Unhealthy
Disabled
Starting
Out of Service
Unknown
AVAILABILITY ZONES
eu-west-1a
eu-west-1b
Clusters
Show
Instances
with details
Create Server Group
Filtered by:
ACCOUNT: frank
STACK: production
Clear All
FRANK
api-production
1986 ▲ / 14 ▼ / 71 ♦ : 95%
US-EAST-1
V175: Build: #4220
667 ▲ / 1 ▼ : 99%
i-abcdefgh
US-WEST-2
V335: Build: #4220
464 ▲ / 13 ▼ / 71 ♦ : 84%
Permalink
i-abcdefgh
Instance Actions
Insight
INSTANCE INFORMATION
Launched: 2015-11-12 12:20:44 PST
In: frank us-east-1e
Type: c3.8xlarge
Server Group: api-production-123
VPC: None (EC2 Classic)
STATUS
Load Balancer
▼ api-production-frontend-1
▼ api-production-frontend-2
DNS
SECURITY GROUPS
LOGS
catalina.out
Log File Archive
Thread Dumps
Admin Proxy Info
Admin Proxy Status

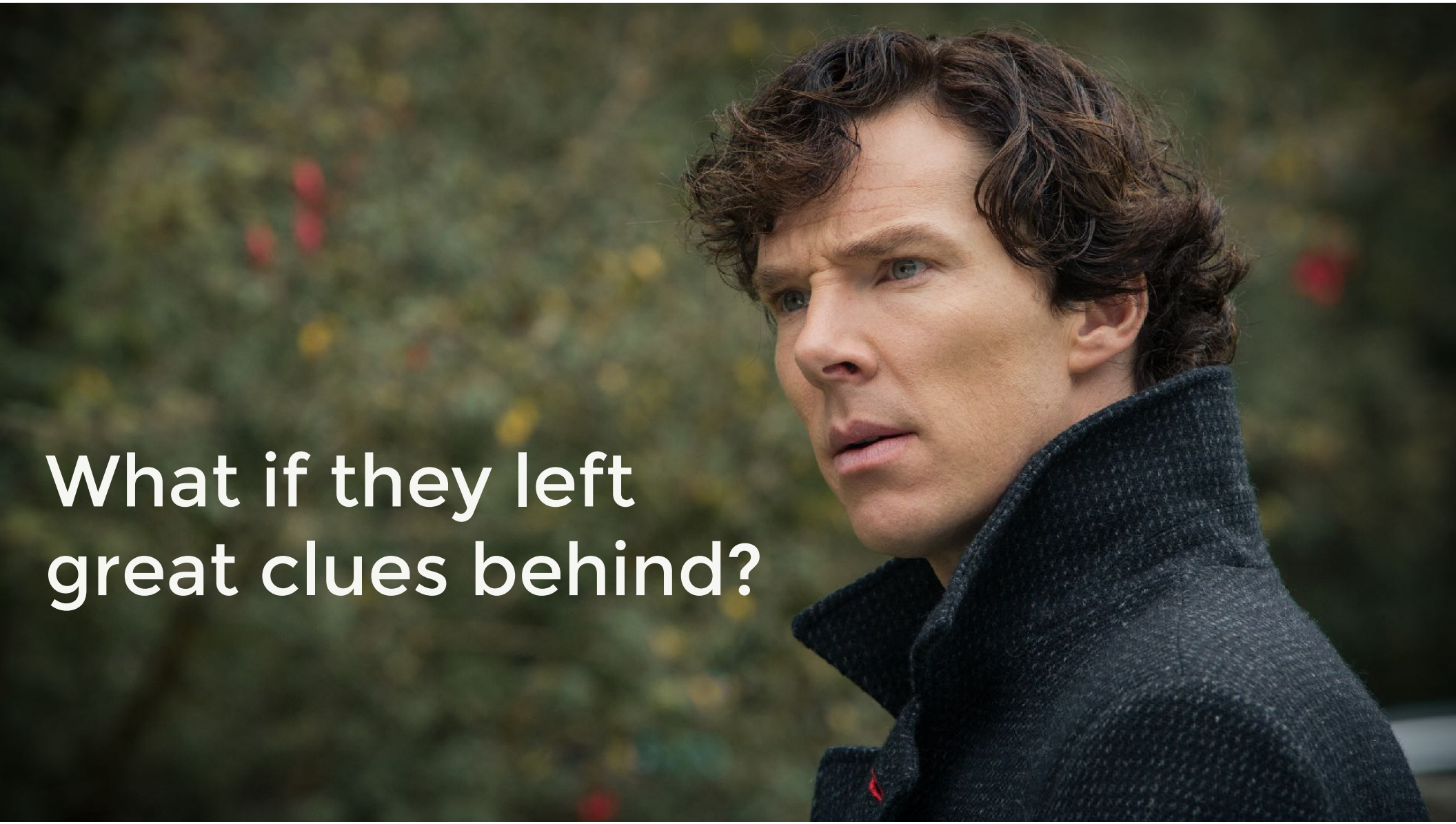
LDAP


LDAP


Operator 1
Operator 2
Operator 3
App A Instances
App B Instances
App C Instances


Operator 1
Operator 2
Operator 3
Bastion
App A Instances
App B Instances
App C Instances

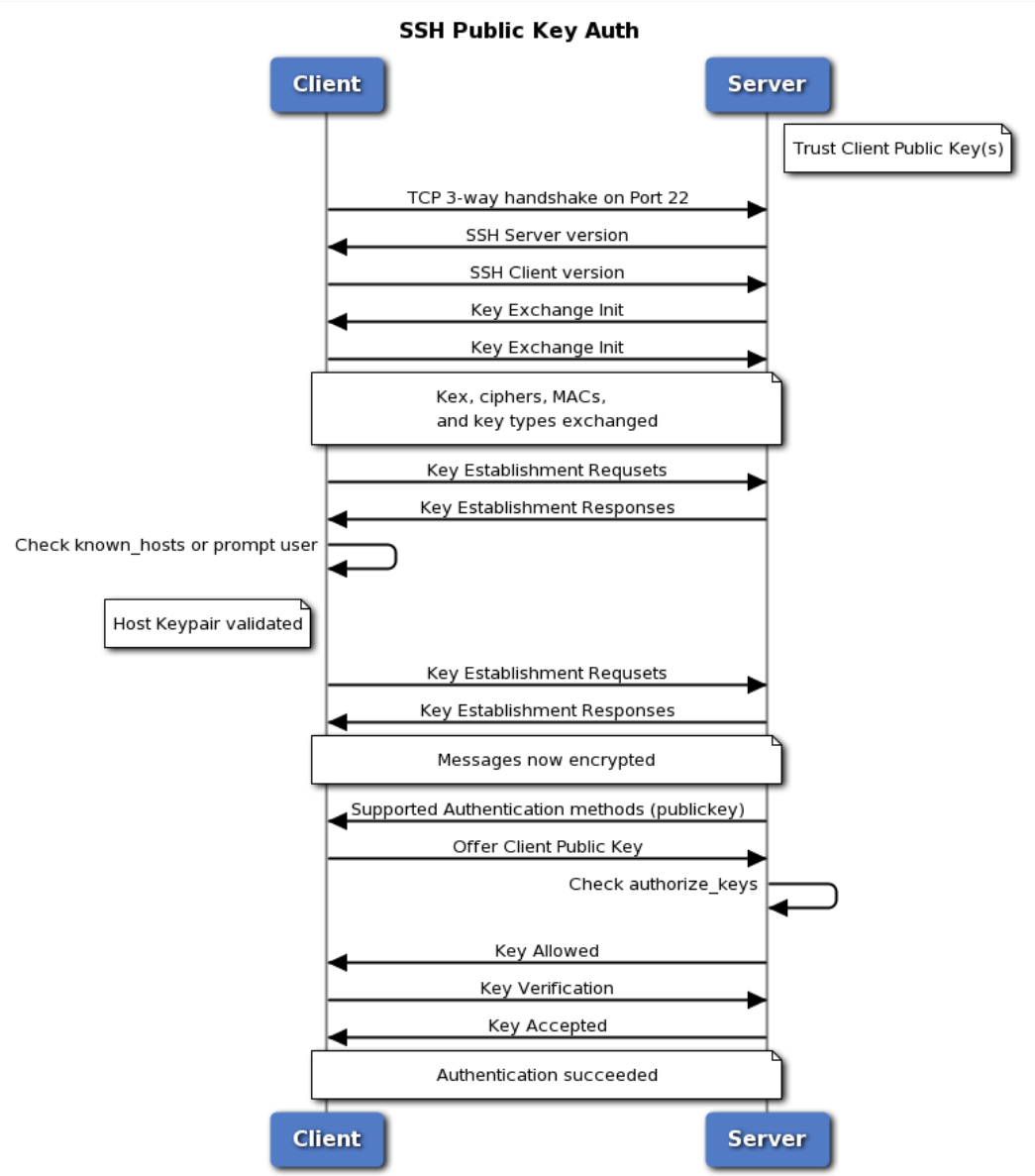

What about single use SSH keys?

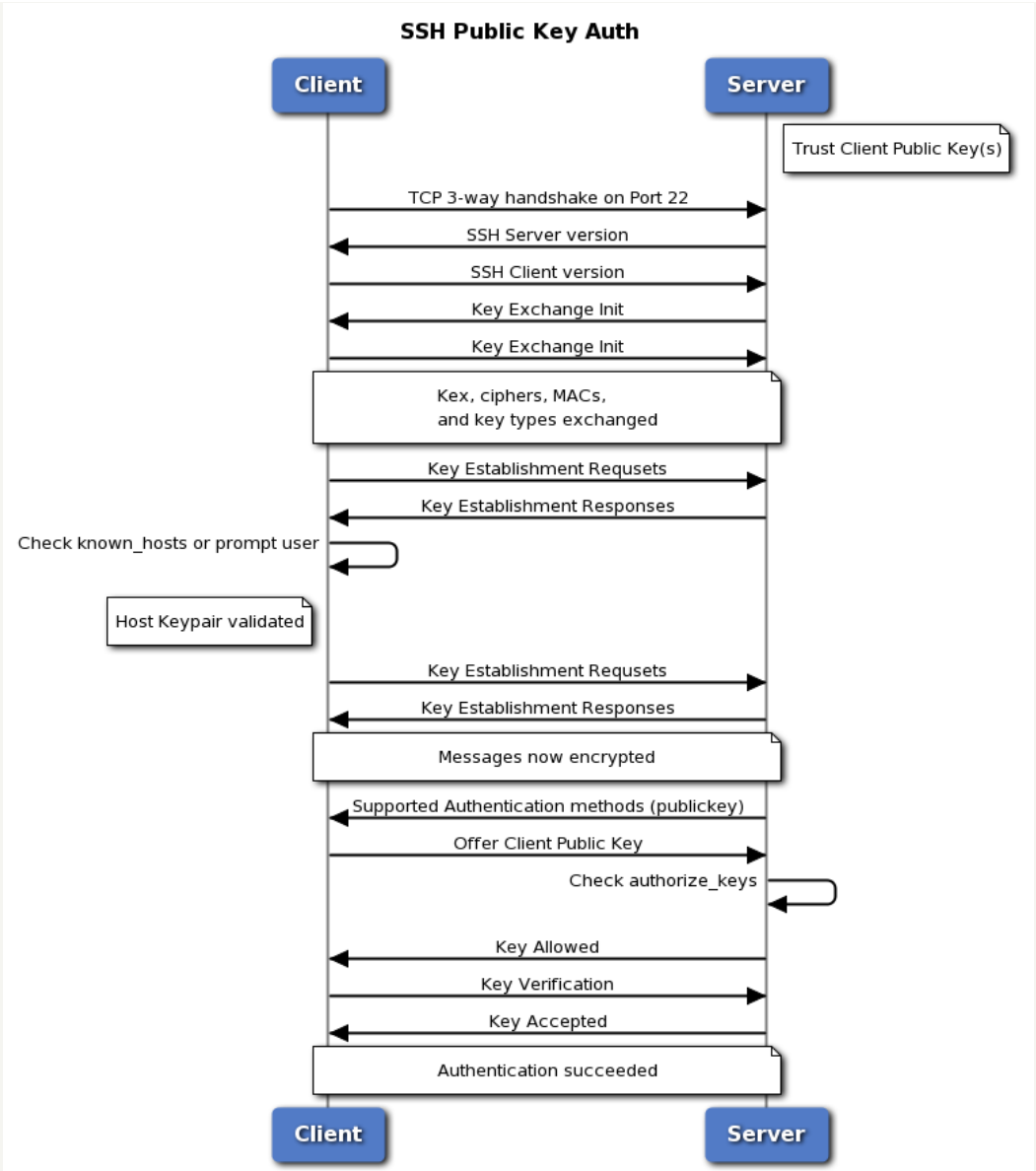

What if they left
great clues behind?

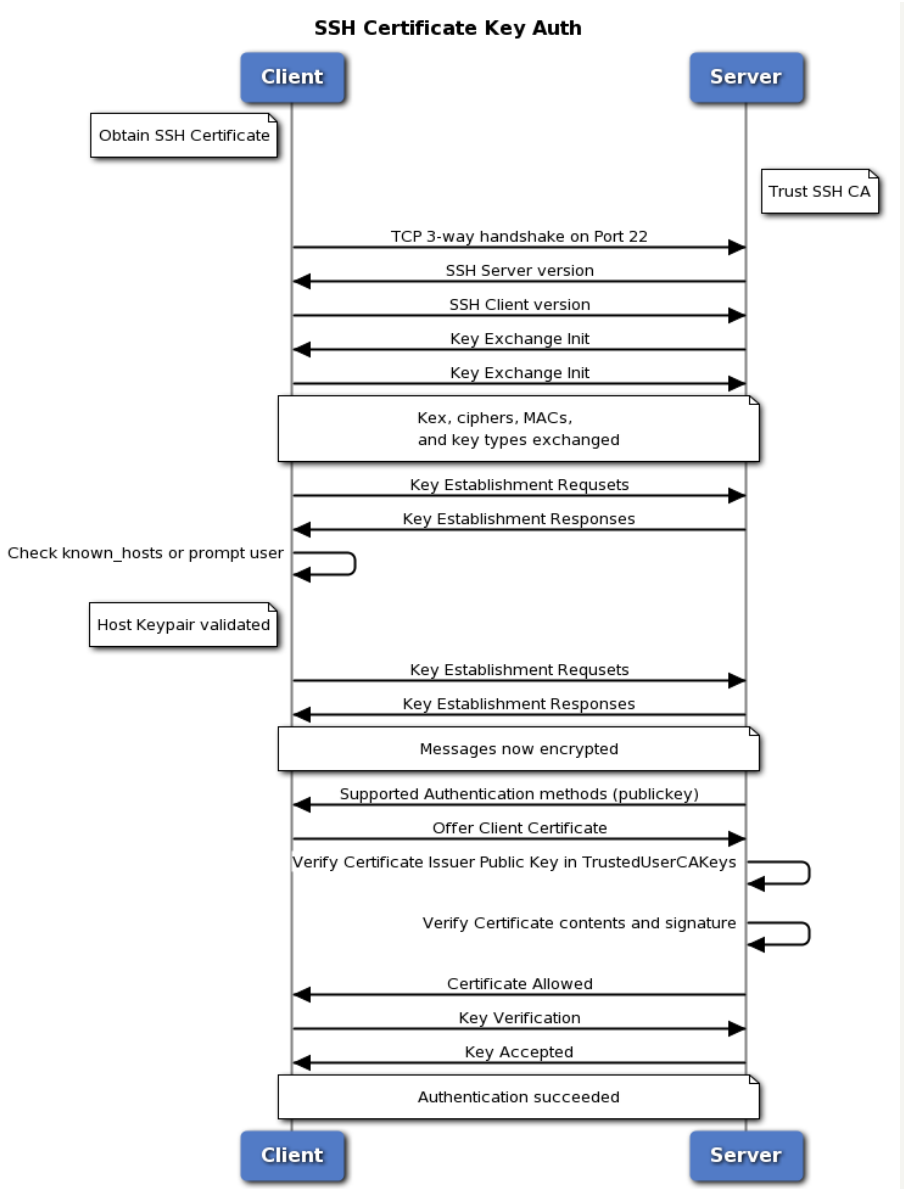

And offered
strong protections?

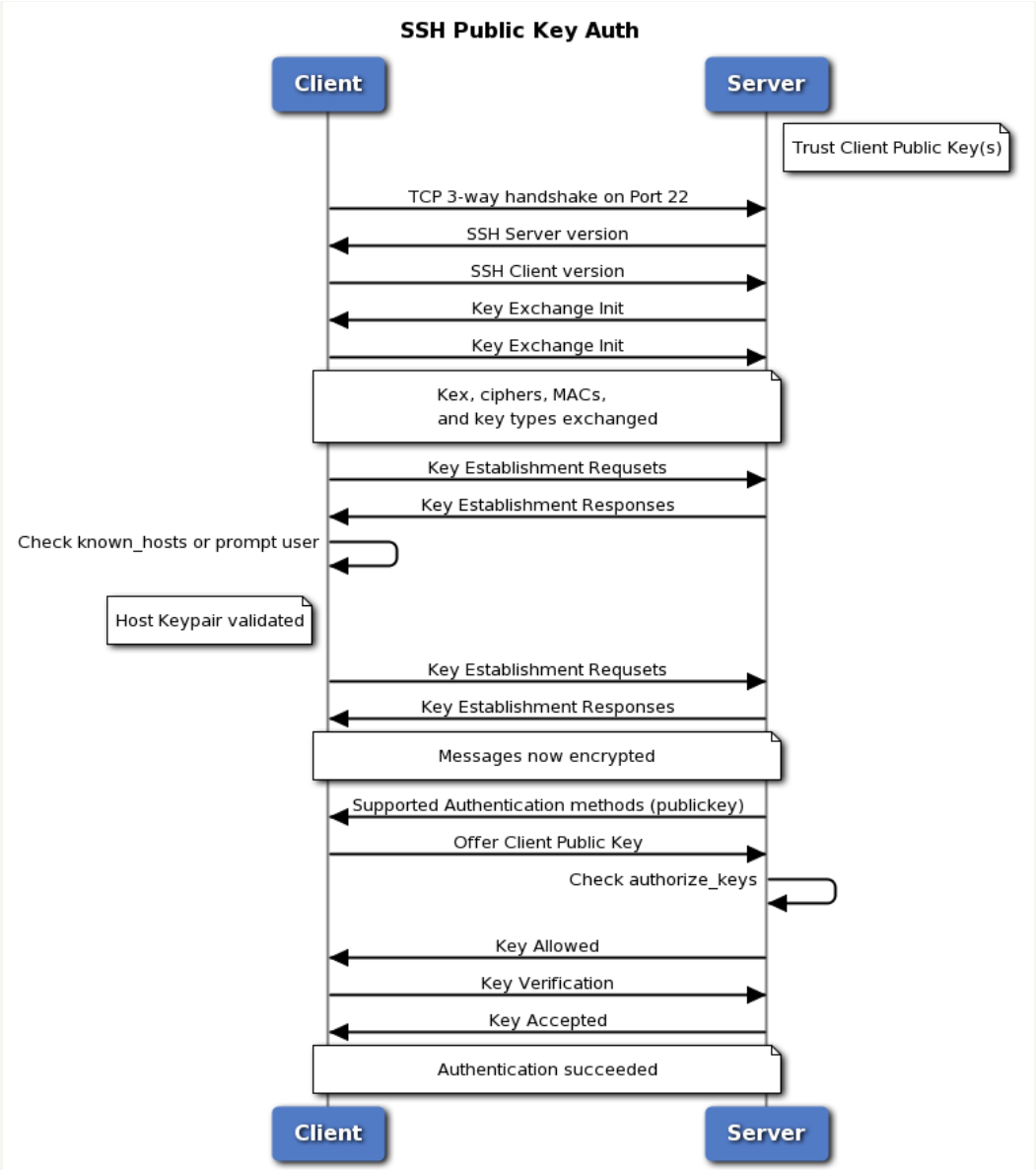
Netflix's Solution

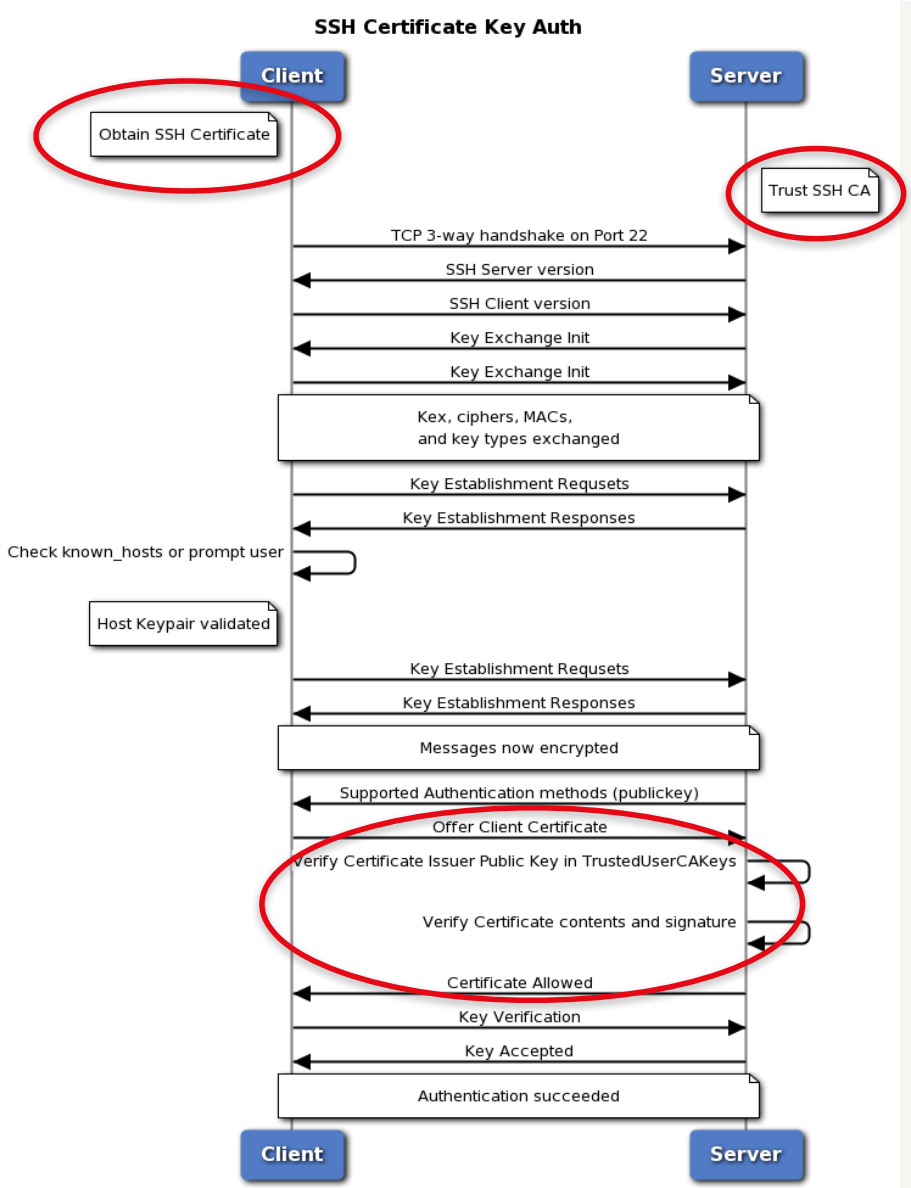
SSH Authentication

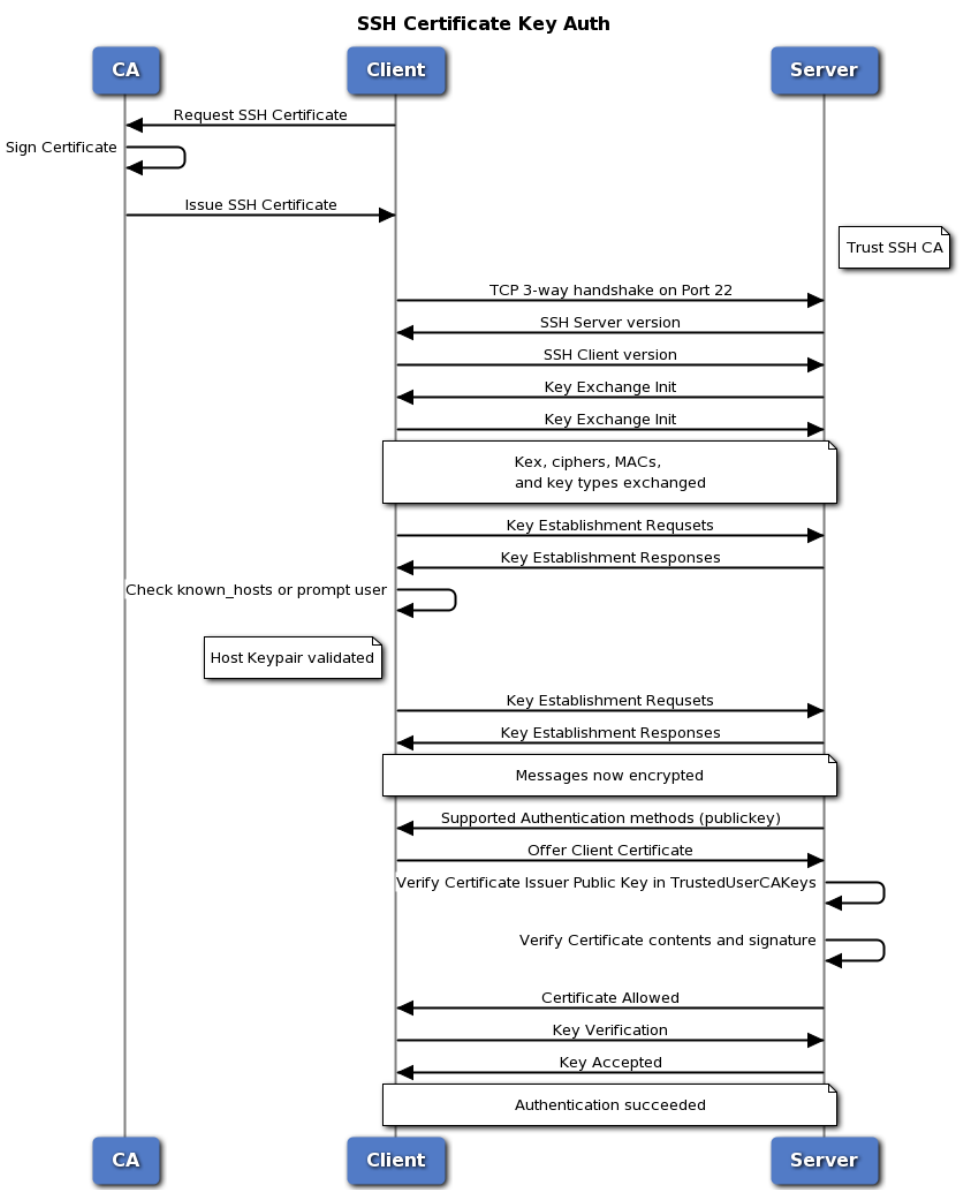

SSH Public Key Auth
Client
Server
Trust Client Public Key(s)
TCP 3-way handshake on Port 22
SSH Server version
SSH Client version
Key Exchange Init
Key Exchange Init
Kex, ciphers, MACs,
and key types exchanged
Key Establishment Requsets
Key Establishment Responses
Check known_hosts or prompt user
Host Keypair validated
Key Establishment Requsets
Key Establishment Responses
Messages now encrypted
Supported Authentication methods (publickey)
Offer Client Public Key
Check authorize_keys
Key Allowed
Key Verification
Key Accepted
Authentication succeeded
Client
Server


SSH Public Key Auth
Client
Server
Trust Client Public Key(s)
TCP 3-way handshake on Port 22
SSH Server version
SSH Client version
Key Exchange Init
Key Exchange Init
Kex, ciphers, MACs,
and key types exchanged
Key Establishment Requsets
Key Establishment Responses
Check known_hosts or prompt user
Host Keypair validated
Key Establishment Requsets
Key Establishment Responses
Messages now encrypted
Supported Authentication methods (publickey)
Offer Client Public Key
Check authorize_keys
Key Allowed
Key Verification
Key Accepted
Authentication succeeded
Client
Server


SSH Certificate Key Auth
Client
Server
Obtain SSH Certificate
Trust SSH CA
TCP 3-way handshake on Port 22
SSH Server version
SSH Client version
Key Exchange Init
Key Exchange Init
Kex, ciphers, MACs,
and key types exchanged
Key Establishment Requsets
Key Establishment Responses
Check known_hosts or prompt user
Host Keypair validated
Key Establishment Requsets
Key Establishment Responses
Messages now encrypted
Supported Authentication methods (publickey)
Offer Client Certificate
Verify Certificate Issuer Public Key in TrustedUserCAKeys
Verify Certificate contents and signature
Certificate Allowed
Key Verification
Key Accepted
Authentication succeeded
Client
Server


SSH Public Key Auth
Client
Server
Trust Client Public Key(s)
TCP 3-way handshake on Port 22
SSH Server version
SSH Client version
Key Exchange Init
Key Exchange Init
Kex, ciphers, MACs,
and key types exchanged
Key Establishment Requsets
Key Establishment Responses
Check known_hosts or prompt user
Host Keypair validated
Key Establishment Requsets
Key Establishment Responses
Messages now encrypted
Supported Authentication methods (publickey)
Offer Client Public Key
Check authorize_keys
Key Allowed
Key Verification
Key Accepted
Authentication succeeded
Client
Server


SSH Certificate Key Auth
Client
Server
Obtain SSH Certificate
Trust SSH CA
TCP 3-way handshake on Port 22
SSH Server version
SSH Client version
Key Exchange Init
Key Exchange Init
Kex, ciphers, MACs,
and key types exchanged
Key Establishment Requsets
Key Establishment Responses
Check known_hosts or prompt user
Host Keypair validated
Key Establishment Requsets
Key Establishment Responses
Messages now encrypted
Supported Authentication methods (publickey)
Offer Client Certificate
Verify Certificate Issuer Public Key in TrustedUserCAKeys
Verify Certificate contents and signature
Certificate Allowed
Key Verification
Key Accepted
Authentication succeeded
Client
Server

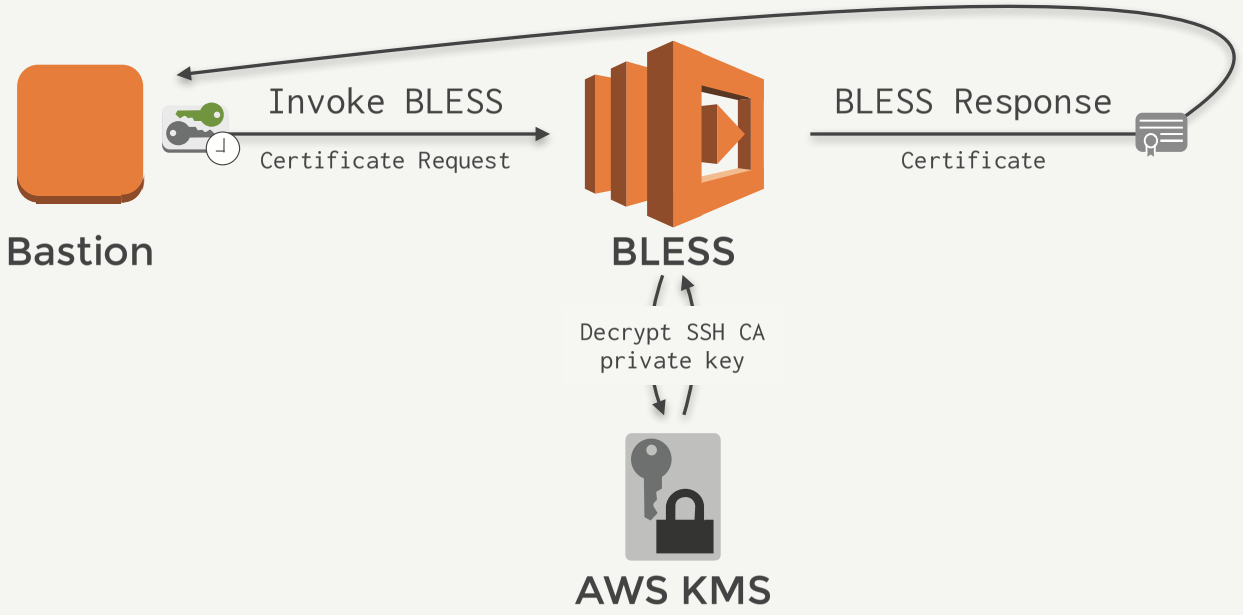

SSH Certificate Key Auth
CA
Client
Server
Request SSH Certificate
Sign Certificate
Issue SSH Certificate
Trust SSH CA
TCP 3-way handshake on Port 22
SSH Server version
SSH Client version
Key Exchange Init
Key Exchange Init
Kex, ciphers, MACs,
and key types exchanged
Key Establishment Requsets
Key Establishment Responses
Check known_hosts or prompt user
Host Keypair validated
Key Establishment Requsets
Key Establishment Responses
Messages now encrypted
Supported Authentication methods (publickey)
Offer Client Certificate
Verify Certificate Issuer Public Key in TrustedUserCAKeys
Verify Certificate contents and signature
Certificate Allowed
Key Verification
Key Accepted
Authentication succeeded
CA
Client
Server

Bastion's
Lambda
Ephemeral
Ssh
Service

BLESS

```
def my_handler(event, context):
    message = 'Hello {} {}!'.format(event['first_name'],
                                    event['last_name'])
    return {
        'message' : message
     }
```


Invoke BLESS
Certificate Request
BLESS Response
Certificate
Bastion
BLESS


Invoke BLESS
Certificate Request
BLESS Response
Certificate
Bastion
BLESS
Decrypt SSH CA
private key
AWS KMS


Instances
SSH with certificate
Invoke BLESS
Certificate Request
BLESS Response
Certificate
Bastion
BLESS
Decrypt SSH CA private key
AWS KMS

SSH Certificates

```
Type: ssh-rsa-cert-v01@openssh.com user certificate
Public key: RSA-CERT SHA256:BLAH
Signing CA: RSA SHA256:BLAH
Key ID: "Any ID information you want"
Serial: 0
Valid: from 2016-05-19T14:30:00 to 2016-05-19T14:34:00
Principals:
        host_username
Critical Options:
        source-address 192.168.1.1
        force-command /bin/date
Extensions:
        permit-X11-forwarding
        permit-agent-forwarding
        permit-port-forwarding
        permit-pty
        permit-user-rc
```

```
Type: ssh-rsa-cert-v01@openssh.com user certificate
Public key: RSA-CERT SHA256:BLAH
Signing CA: RSA SHA256:BLAH
Key ID: "Any ID information you want"
Serial: 0
Valid: from 2016-05-19T14:30:00 to 2016-05-19T14:34:00
Principals:
        host_username
Critical Options:
        source-address 192.168.1.1
        force-command /bin/date
Extensions:
        permit-X11-forwarding
        permit-agent-forwarding
        permit-port-forwarding
        permit-pty
        permit-user-rc
```

User or Host Certificates

Type: ssh-rsa-cert-v01@openssh.com user certificate
Public key: RSA-CERT SHA256:BLAH
Signing CA: RSA SHA256:BLAH
Key ID: "Any ID information you want"
Serial: 0
Valid: from 2016-05-19T14:30:00 to 2016-05-19T14:34:00
Principals:
    host_username
Critical Options:
    source-address 192.168.1.1
    force-command /bin/date
Extensions:
    permit-X11-forwarding
    permit-agent-forwarding
    permit-port-forwarding
    permit-pty
    permit-user-rc

Control over what is logged by SSHd

Type: ssh-rsa-cert-v01@openssh.com user certificate
Public key: RSA-CERT SHA256:BLAH
Signing CA: RSA SHA256:BLAH
Key ID: "Any ID information you want"
Serial: 0
Valid: from 2016-05-19T14:30:00 to 2016-05-19T14:34:00
Principals:
    host_username
Critical Options:
    source-address 192.168.1.1
    force-command /bin/date
Extensions:
    permit-X11-forwarding
    permit-agent-forwarding
    permit-port-forwarding
    permit-pty
    permit-user-rc

Short-lived certs reduce risk

```
Type: ssh-rsa-cert-v01@openssh.com user certificate
Public key: RSA-CERT SHA256:BLAH
Signing CA: RSA SHA256:BLAH
Key ID: "Any ID information you want"
Serial: 0
Valid: from 2016-05-19T14:30:00 to 2016-05-19T14:34:00
Principals:
        host_username
Critical Options:
        source-address 192.168.1.1
        force-command /bin/date
Extensions:
        permit-X11-forwarding
        permit-agent-forwarding
        permit-port-forwarding
        permit-pty
        permit-user-rc
```

Valid for a single target (account, app, username, etc)

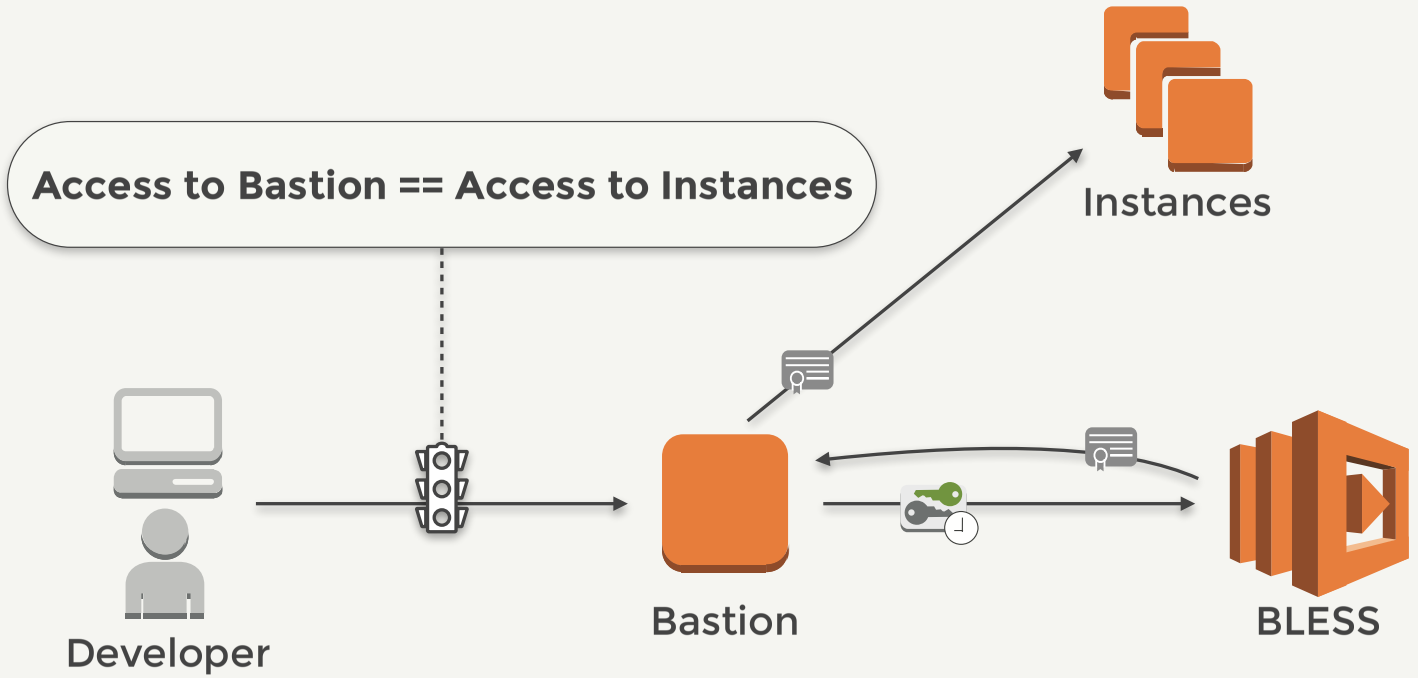
Type: ssh-rsa-cert-v01@openssh.com user certificate
Public key: RSA-CERT SHA256:BLAH
Signing CA: RSA SHA256:BLAH
Key ID: "Any ID information you want"
Serial: 0
Valid: from 2016-05-19T14:30:00 to 2016-05-19T14:34:00
Principals:
    host_username
Critical Options:
    source-address 192.168.1.1
    force-command /bin/date
Extensions:
    permit-X11-forwarding
    permit-agent-forwarding
    permit-port-forwarding
    permit-pty
    permit-user-rc

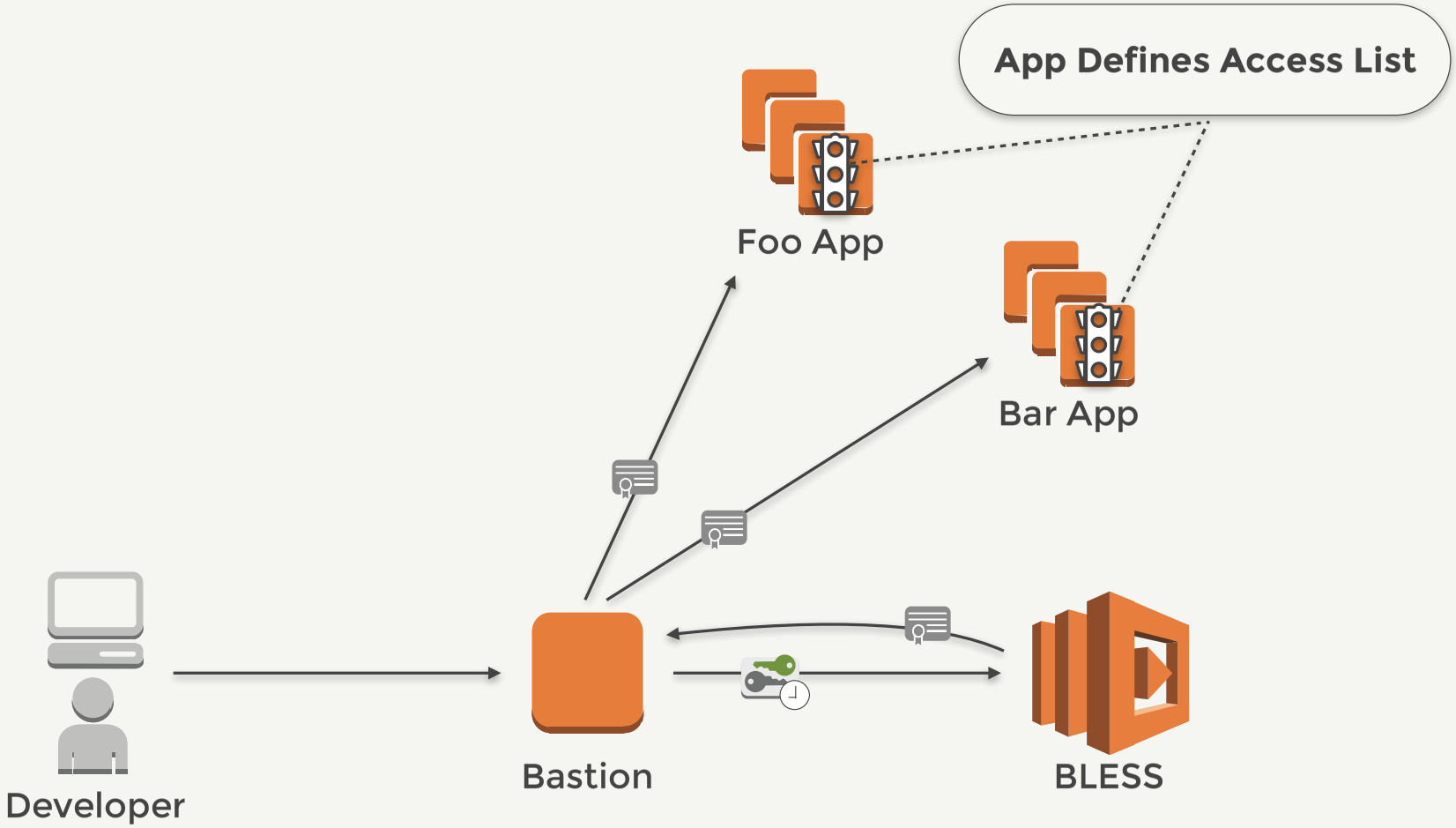
Valid from a single host

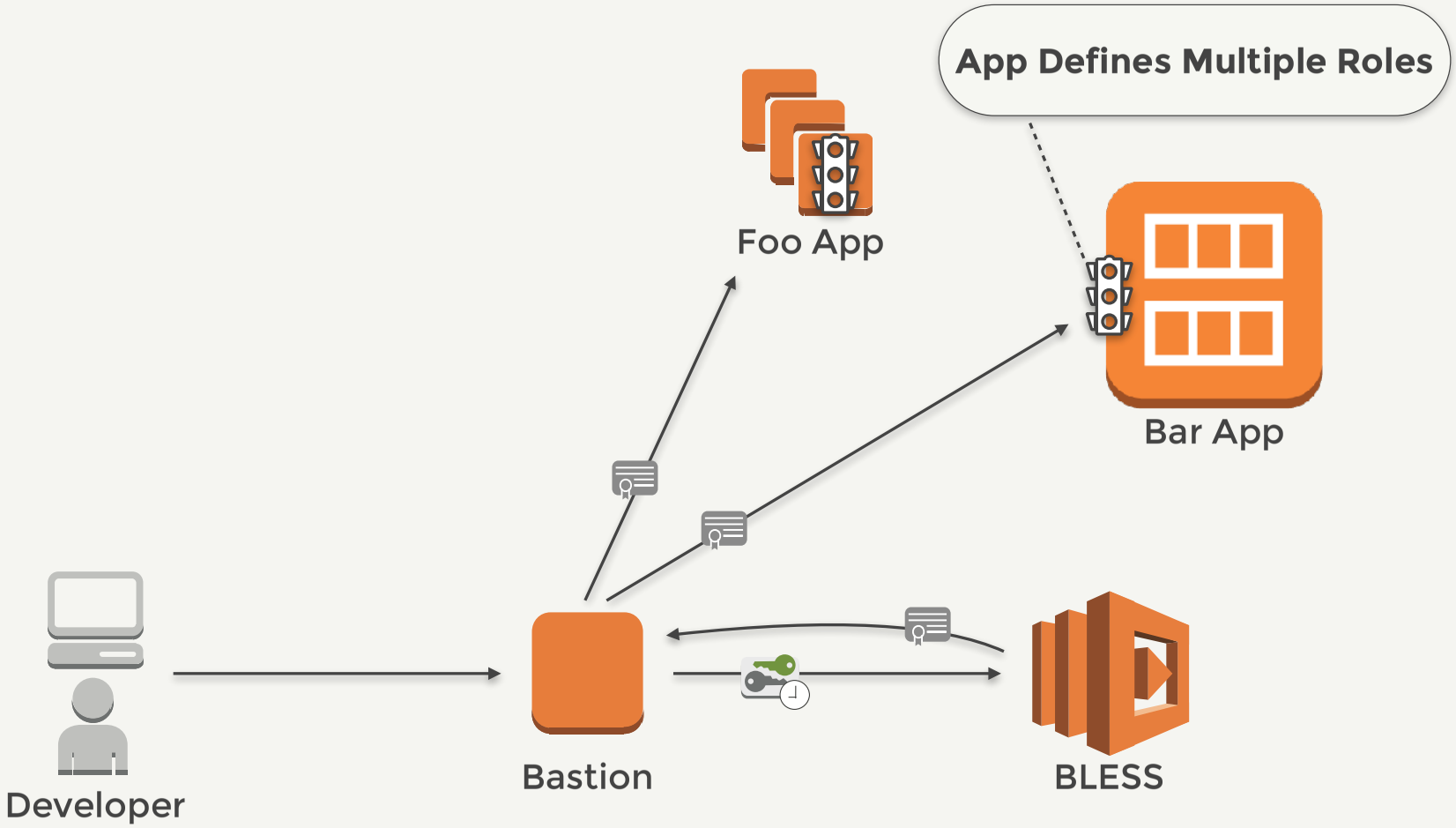
Type: ssh-rsa-cert-v01@openssh.com user certificate
Public key: RSA-CERT SHA256:BLAH
Signing CA: RSA SHA256:BLAH
Key ID: "Any ID information you want"
Serial: 0
Valid: from 2016-05-19T14:30:00 to 2016-05-19T14:34:00
Principals:
    host_username
Critical Options:
    source-address 192.168.1.1
    force-command /bin/date
Extensions:
    permit-X11-forwarding
    permit-agent-forwarding
    permit-port-forwarding
    permit-pty
    permit-user-rc

Control what the SSH session can be used for

Scoping Credentials


Access to Bastion == Access to Instances
Instances
Developer
Bastion
BLESS


App Defines Access List
Foo App
Bar App
Developer
Bastion
BLESS


App Defines Multiple Roles
Foo App
Bar App
Developer
Bastion
BLESS

Type: ssh-rsa-cert-v01@openssh.com user certificate
Public key: RSA-CERT SHA256:BLAH
Signing CA: RSA SHA256:BLAH
Key ID: "Any ID information you want"
Serial: 0
Valid: from 2016-05-19T14:30:00 to 2016-05-19T14:34:00
Principals:
    host_username
Critical Options:
    source-address 192.168.1.1
    force-command /bin/date
Extensions:
    permit-X11-forwarding
    permit-agent-forwarding
    permit-port-forwarding
    permit-pty
    permit-user-rc

instance_user:aws_account:app_name

Config File

/etc/ssh/sshd_config

```
# Entries to enable BLESS
TrustedUserCAKeys /etc/ssh/bless_user_ssh_cas.pub
AuthorizedPrincipalsFile /etc/ssh/authorized_principals/%u
```

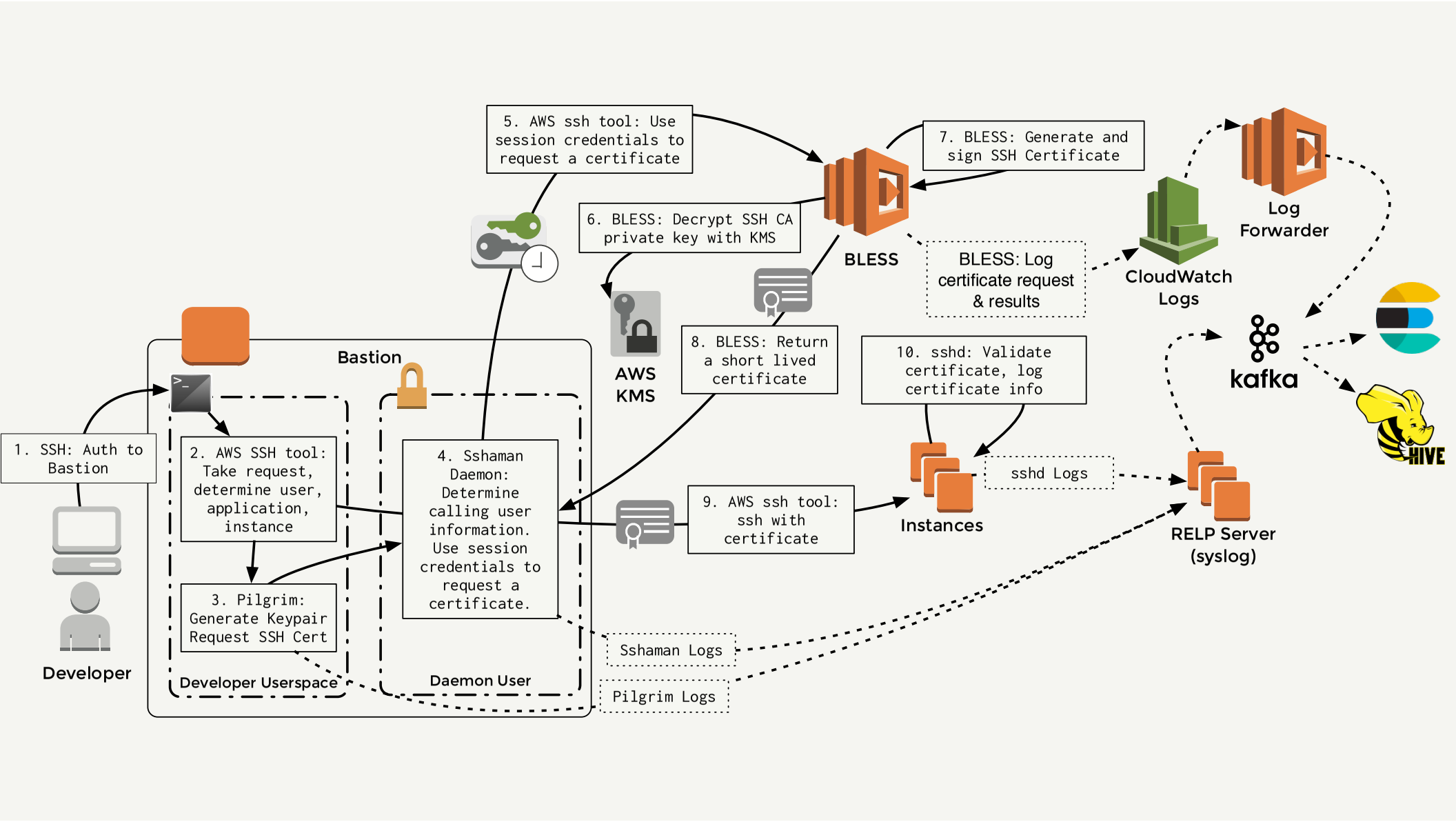
Config File

/etc/ssh/authorized_principals/blessdemo

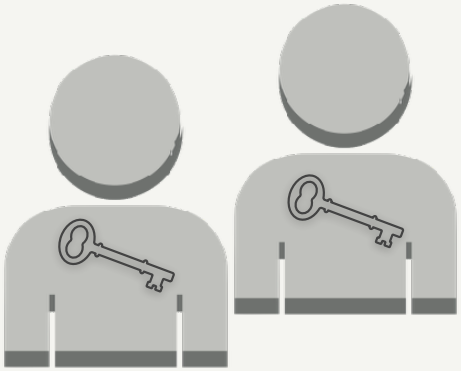
```
bless_demo_instances:bless_demo_instances:123456789012:i-18badf00ddeadbeef
```

Operational Wins

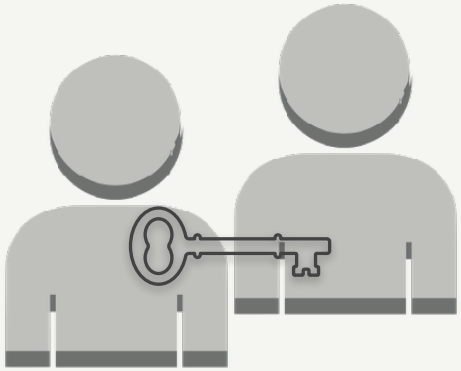

Instances
SSH with certificate
Invoke BLESS
Certificate Request
BLESS Response
Certificate
Bastion
BLESS
Decrypt SSH CA private key
AWS KMS


5. AWS ssh tool: Use session credentials to request a certificate
7. BLESS: Generate and sign SSH Certificate
6. BLESS: Decrypt SSH CA private key with KMS
BLESS
BLESS: Log certificate request & results
CloudWatch Logs
Log Forwarder
AWS KMS
8. BLESS: Return a short lived certificate
10. sshd: Validate certificate, log certificate info
kafka
Bastion
1. SSH: Auth to Bastion
2. AWS SSH tool: Take request, determine user, application, instance
4. Sshaman Daemon: Determine calling user information. Use session credentials to request a certificate.
9. AWS ssh tool: ssh with certificate
Instances
sshd Logs
RELP Server (syslog)
3. Pilgrim: Generate Keypair Request SSH Cert
Developer
Developer Userspace
Daemon User
Sshaman Logs
Pilgrim Logs

Key Secrecy



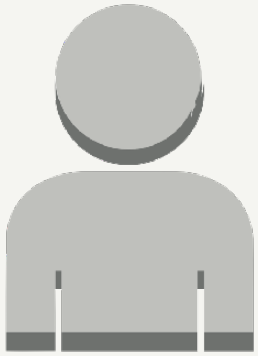
Personal Keys



Shared Keys

Expiration

Key Rotation



Human

VS

Machine

Logging Context

```
Jun 22 00:20:34 bless-demo-
instances-i-0123456789abcde
sshd[####]: Accepted publickey
for bless_demo_instances from
192.168.1.1 port ##### ssh2: RSA
SHA256:de:ad:be:ef:
00:00:00:00:00:de:ad:be
```

Traditional

```
Jun 22 00:20:55 bless-demo-
instances-i-0123456789abcde
sshd[####]: Accepted publickey
for bless_demo_instances from
192.168.1.1 port ##### ssh2:
RSA-CERT ID
request[##################]
for[user_name] from[10.0.1.1]
command[test:us-
east-1:bless_demo_instances:bles
s_demo_instances-v001:oq-ssh]
ssh_key[RSA de:ad:be:ef:
00:00:00:00:00:de:ad:be]
ca[arn:aws:lambda:region:account
:function:name]
valid_to[2017/06/22 00:25:53]
(serial 0) CA RSA
SHA256:8badf00d000000008bad
```

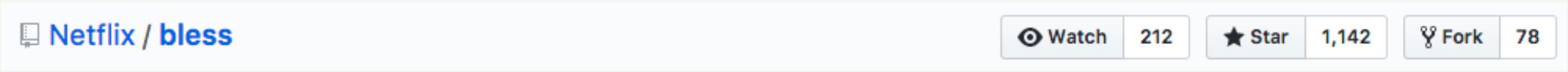
SSH certificates with BLESS

Availability Wins

Yes, It's Open Source!

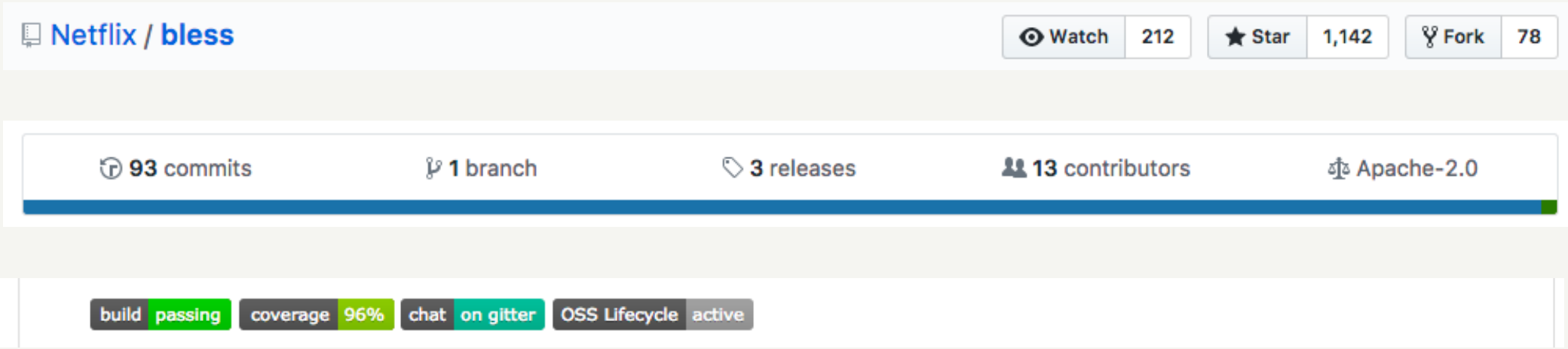
https://github.com/Netflix/bless

https://github.com/Netflix/bless



https://github.com/Netflix/bless

https://github.com/Netflix/bless



https://github.com/Netflix/bless

BLESS - Bastion's Lambda Ephemeral SSH Service

BLESS is an SSH Certificate Authority that runs as an AWS Lambda function and is used to sign SSH public keys.

SSH Certificates are an excellent way to authorize users to access a particular SSH host, as they can be restricted for a single use case, and can be short lived. Instead of managing the authorized_keys of a host, or controlling who has access to SSH Private Keys, hosts just need to be configured to trust an SSH CA.

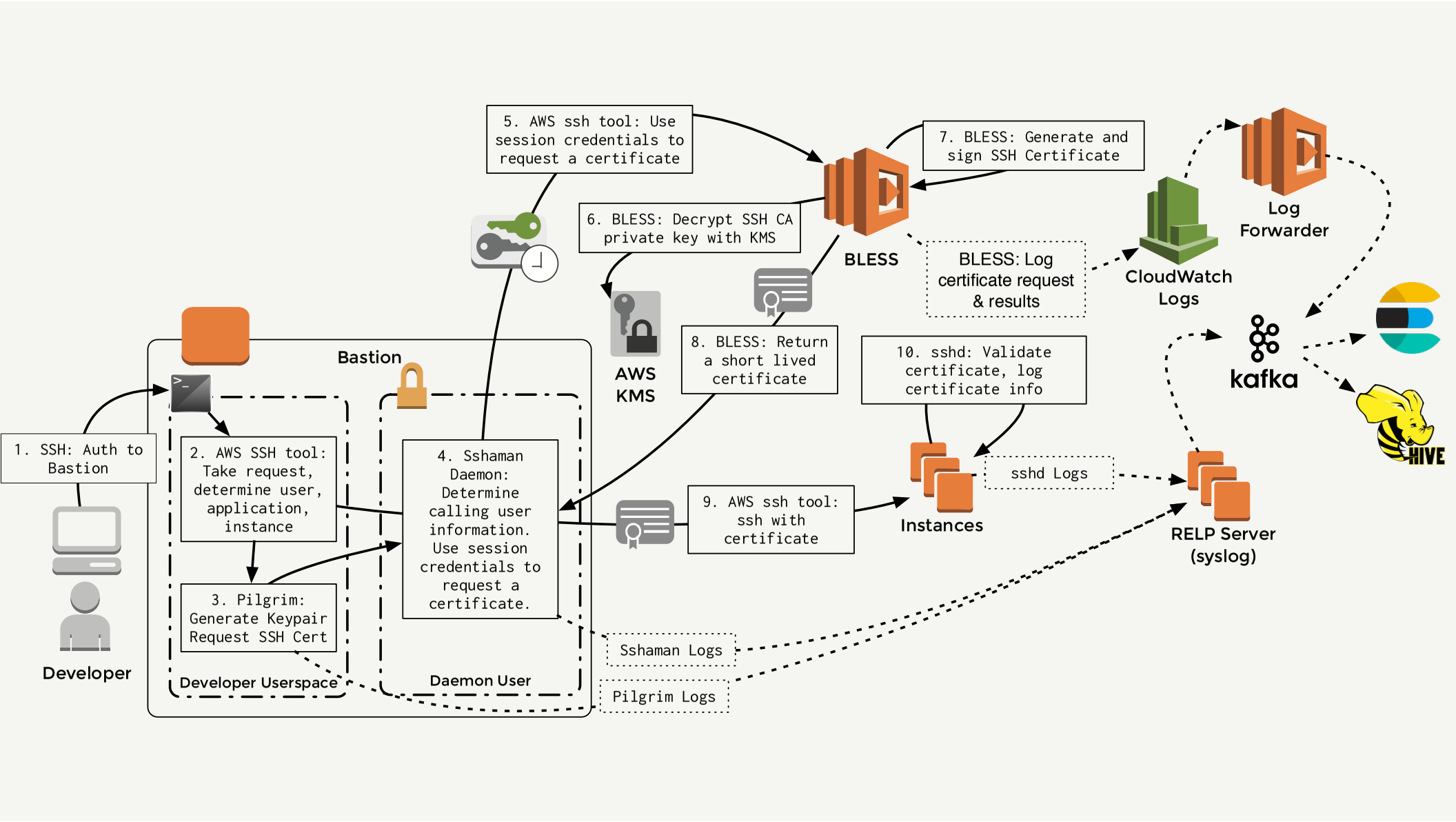
BLESS should be run as an AWS Lambda in an isolated AWS account. Because BLESS needs access to a private key which is trusted by your hosts, an isolated AWS account helps restrict who can access that private key, or modify the BLESS code you are running.

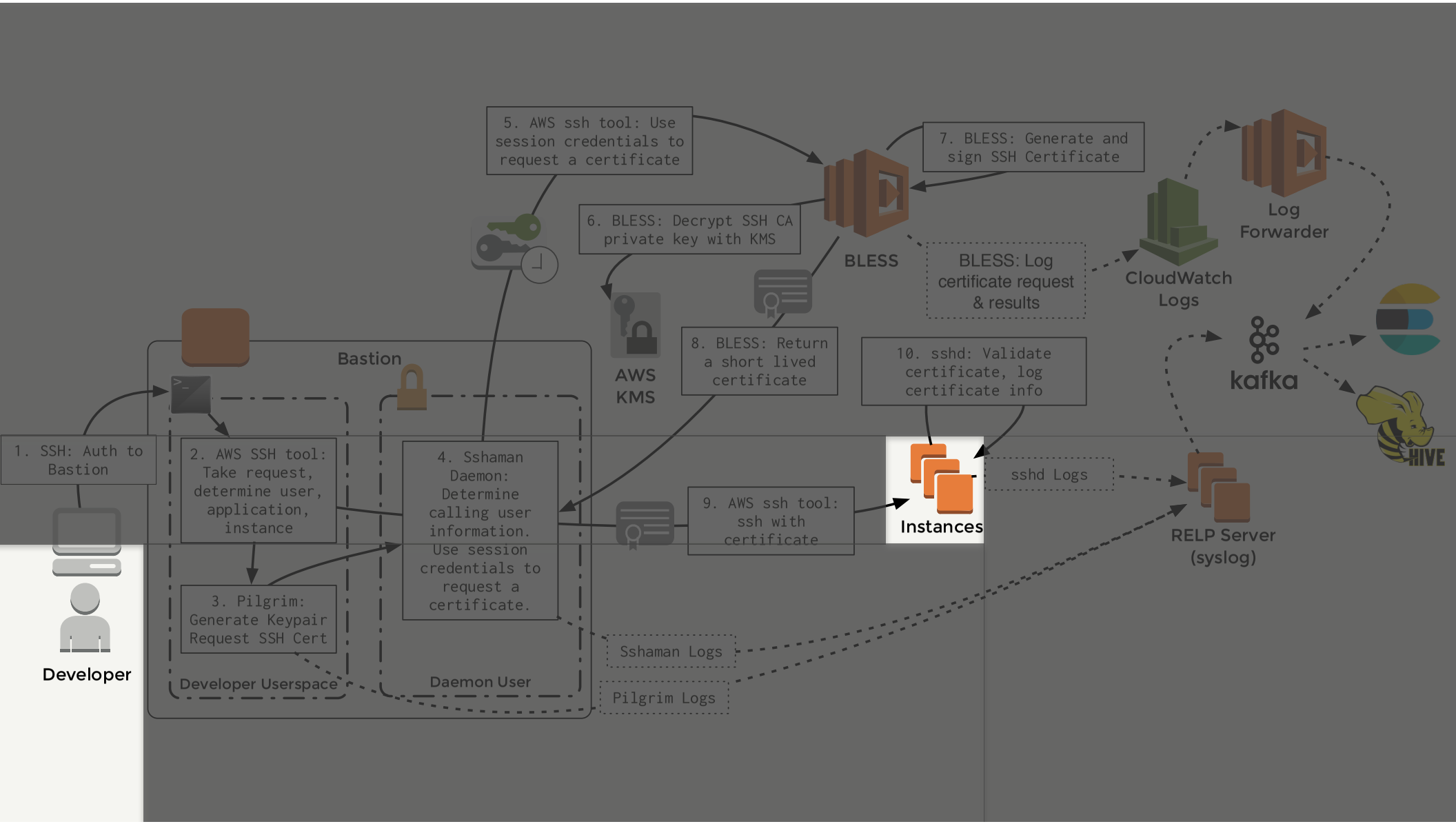
AWS Lambda functions can use an AWS IAM Policy to limit which IAM Roles can invoke the Lambda Function. If properly configured, you can restrict which IAM Roles can request SSH Certificates. For example, your SSH Bastion (aka SSH Jump Host) can run with the only IAM Role with access to invoke a BLESS Lambda Function configured with the SSH CA key trusted by the instances accessible to that SSH Bastion.

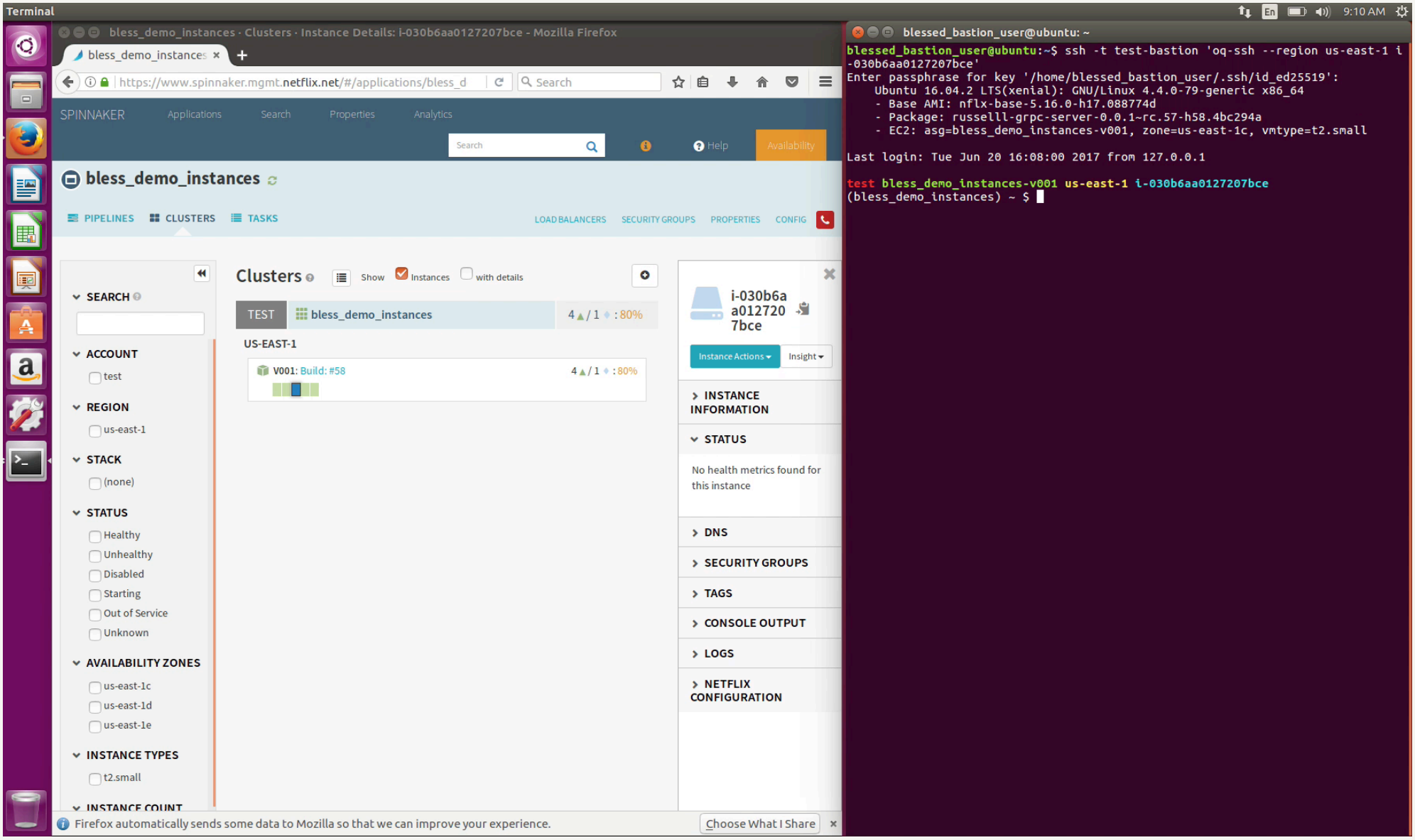
Getting Started

Demo Time

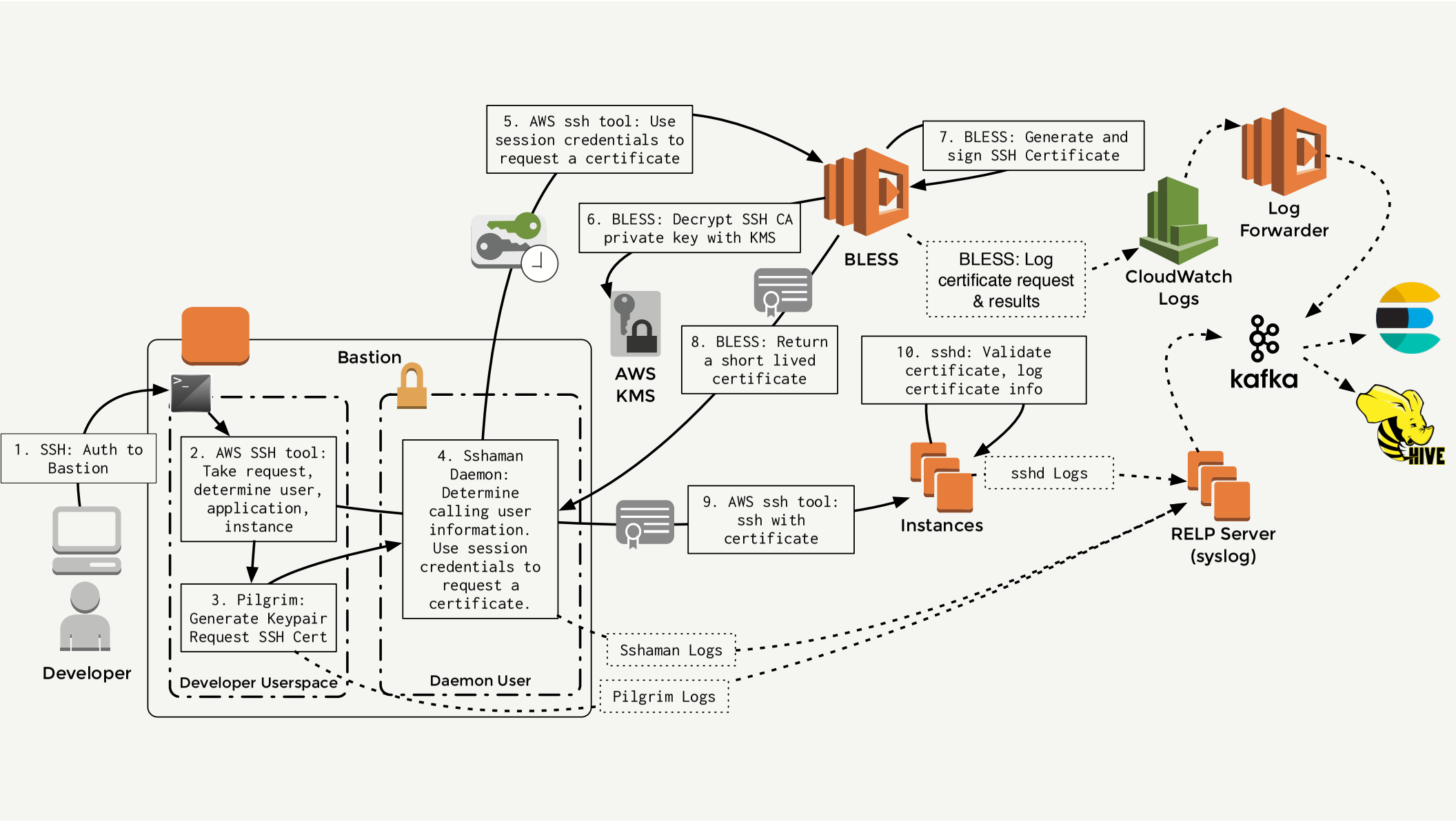
User Experience

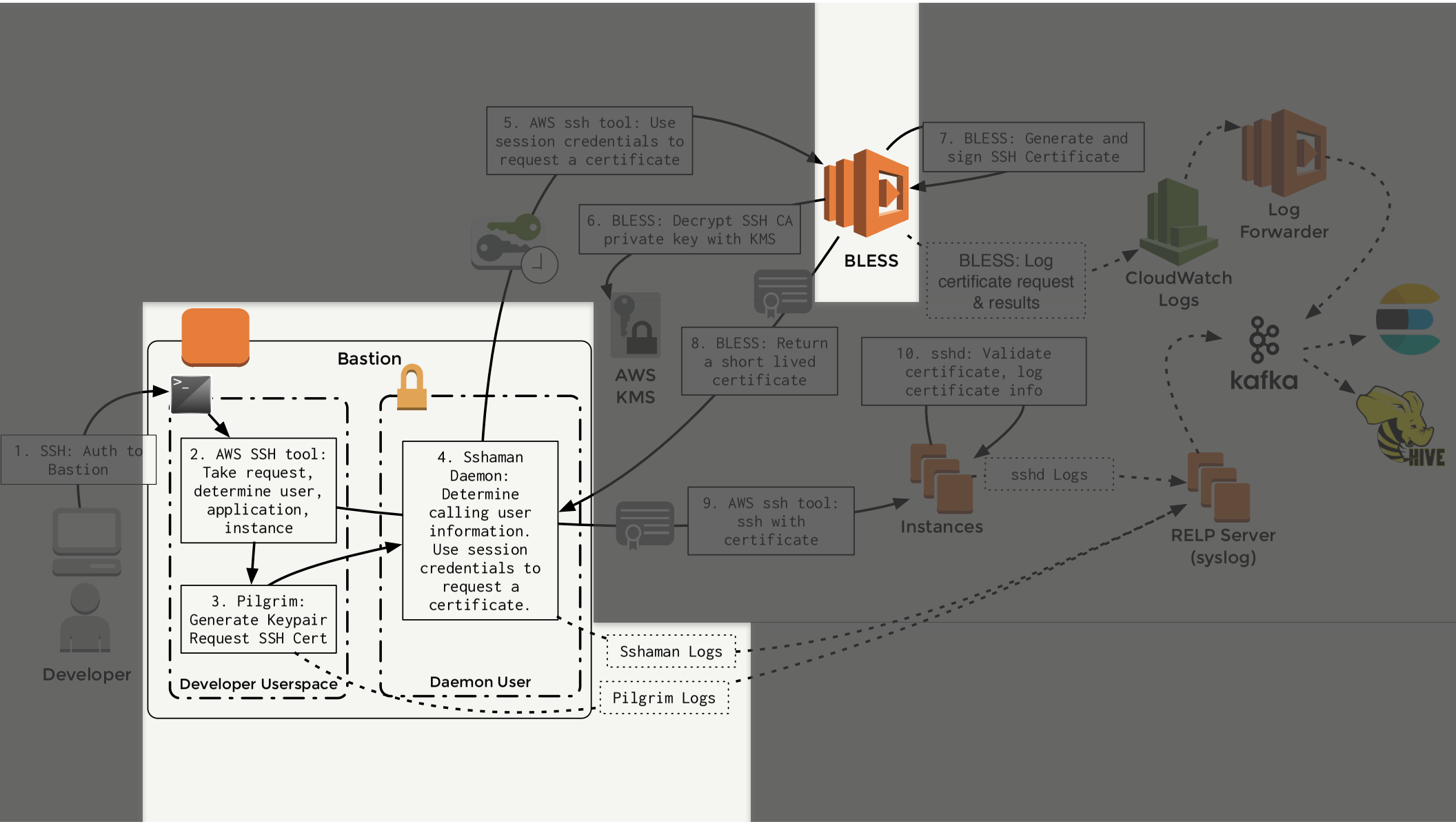

5. AWS ssh tool: Use session credentials to request a certificate
7. BLESS: Generate and sign SSH Certificate
6. BLESS: Decrypt SSH CA private key with KMS
BLESS
BLESS: Log certificate request & results
CloudWatch Logs
Log Forwarder
AWS KMS
8. BLESS: Return a short lived certificate
10. sshd: Validate certificate, log certificate info
kafka
Bastion
1. SSH: Auth to Bastion
2. AWS SSH tool: Take request, determine user, application, instance
4. Sshaman Daemon: Determine calling user information. Use session credentials to request a certificate.
9. AWS ssh tool: ssh with certificate
Instances
sshd Logs
RELP Server (syslog)
3. Pilgrim: Generate Keypair Request SSH Cert
Developer
Developer Userspace
Daemon User
Sshaman Logs
Pilgrim Logs

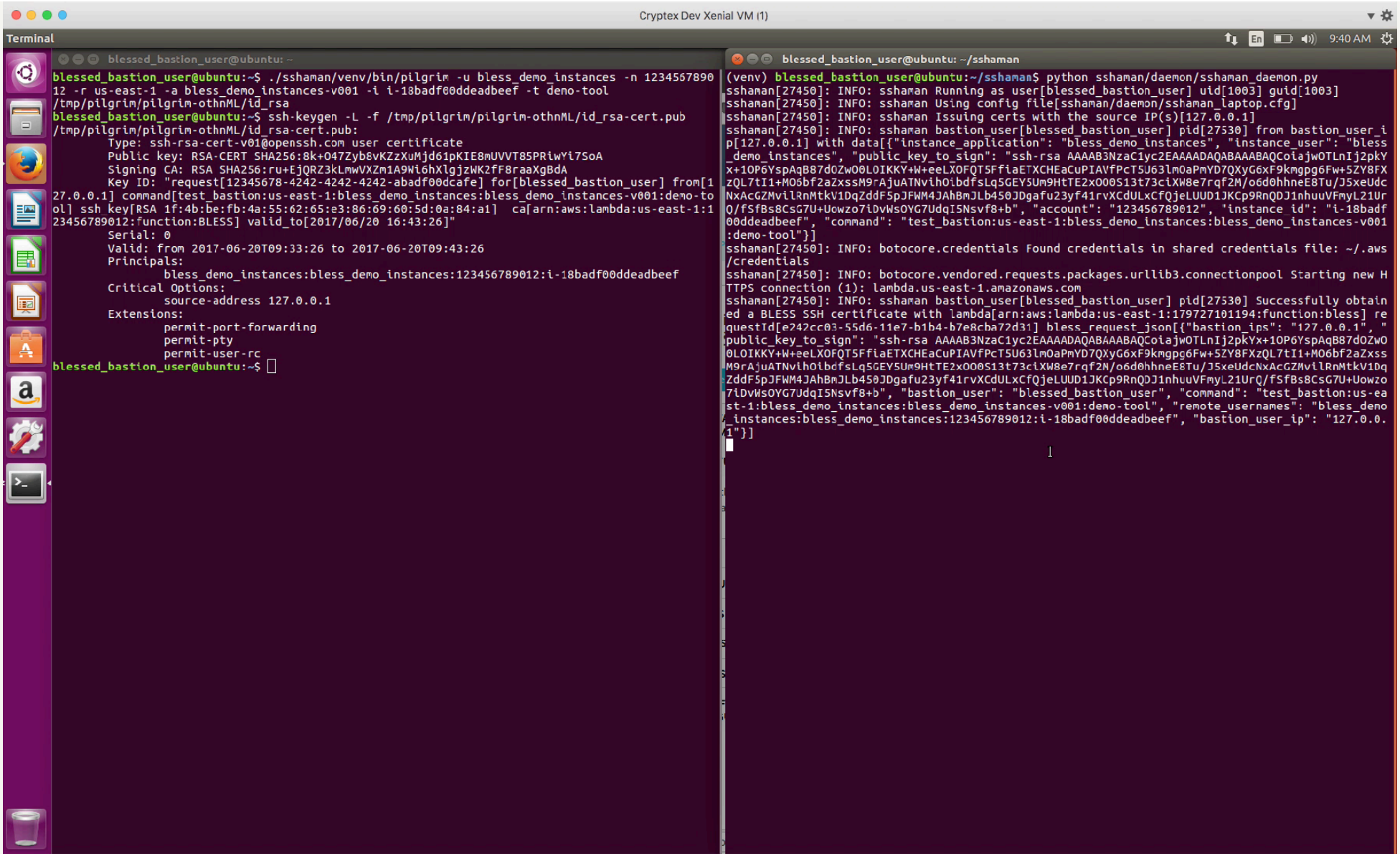

5. AWS ssh tool: Use session credentials to request a certificate
7. BLESS: Generate and sign SSH Certificate
6. BLESS: Decrypt SSH CA private key with KMS
BLESS
BLESS: Log certificate request & results
CloudWatch Logs
Log Forwarder
AWS KMS
8. BLESS: Return a short lived certificate
10. sshd: Validate certificate, log certificate info
kafka
Bastion
1. SSH: Auth to Bastion
2. AWS SSH tool: Take request, determine user, application, instance
4. Sshaman Daemon: Determine calling user information. Use session credentials to request a certificate.
9. AWS ssh tool: ssh with certificate
Instances
sshd Logs
RELP Server (syslog)
3. Pilgrim: Generate Keypair Request SSH Cert
Developer
Developer Userspace
Daemon User
Sshaman Logs
Pilgrim Logs


Terminal
9:10 AM
bless_demo_instances · Clusters · Instance Details: i-030b6aa0127207bce - Mozilla Firefox
https://www.spinnaker.mgmt.netflix.net/#/applications/bless_d
SPINNAKER
Applications
Search
Properties
Analytics
Help
Availability
bless_demo_instances
PIPELINES
CLUSTERS
TASKS
LOAD BALANCERS
SECURITY GROUPS
PROPERTIES
CONFIG
SEARCH
ACCOUNT
test
REGION
us-east-1
STACK
(none)
STATUS
Healthy
Unhealthy
Disabled
Starting
Out of Service
Unknown
AVAILABILITY ZONES
us-east-1c
us-east-1d
us-east-1e
INSTANCE TYPES
t2.small
Clusters
Show
Instances
with details
TEST
bless_demo_instances
4 / 1 : 80%
US-EAST-1
V001: Build: #58
4 / 1 : 80%
i-030b6a
a012720
7bce
Instance Actions
Insight
INSTANCE INFORMATION
STATUS
No health metrics found for this instance
DNS
SECURITY GROUPS
TAGS
CONSOLE OUTPUT
LOGS
NETFLIX CONFIGURATION
Firefox automatically sends some data to Mozilla so that we can improve your experience.
Choose What I Share
blessed_bastion_user@ubuntu: ~
blessed_bastion_user@ubuntu:~$ ssh -t test-bastion 'oq-ssh --region us-east-1 i
-030b6aa0127207bce'
Enter passphrase for key '/home/blessed_bastion_user/.ssh/id_ed25519':
Ubuntu 16.04.2 LTS(xenial): GNU/Linux 4.4.0-79-generic x86_64
- Base AMI: nflx-base-5.16.0-h17.088774d
- Package: russelll-grpc-server-0.0.1~rc.57-h58.4bc294a
- EC2: asg=bless_demo_instances-v001, zone=us-east-1c, vmtype=t2.small
Last login: Tue Jun 20 16:08:00 2017 from 127.0.0.1
test bless_demo_instances-v001 us-east-1 i-030b6aa0127207bce
(bless_demo_instances) ~ $

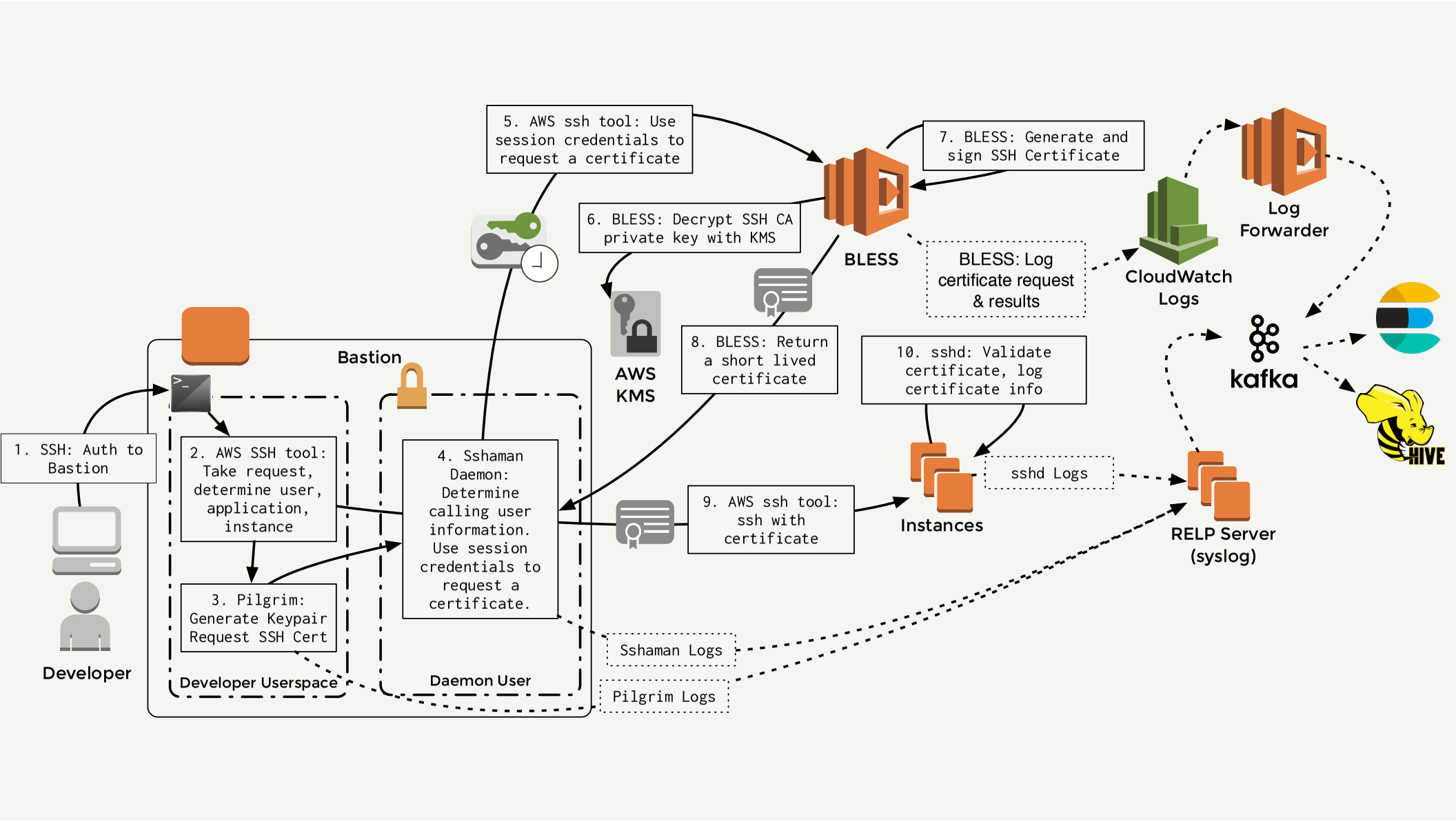
Bastion Using BLESS

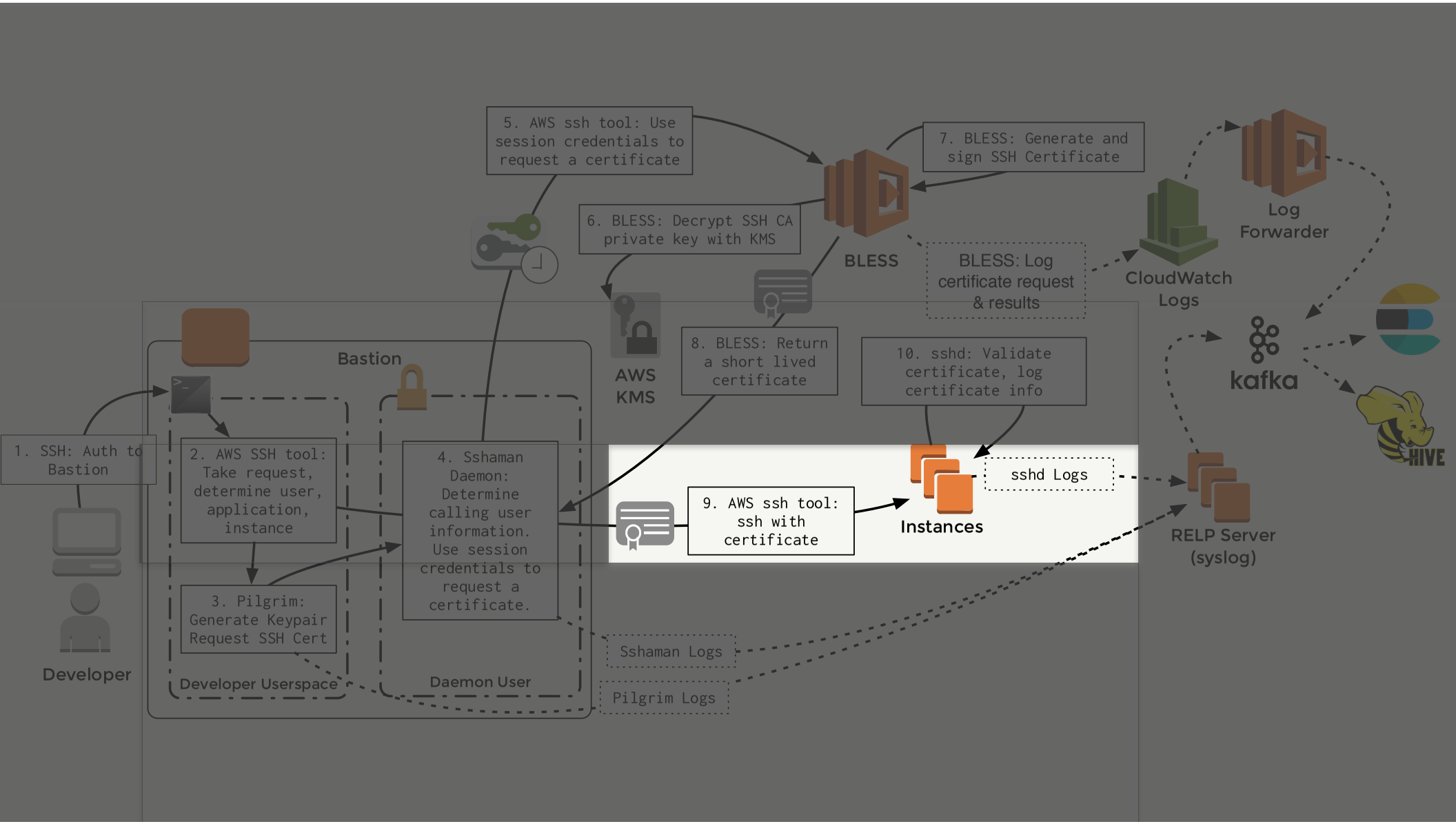

5. AWS ssh tool: Use session credentials to request a certificate
7. BLESS: Generate and sign SSH Certificate
6. BLESS: Decrypt SSH CA private key with KMS
BLESS
BLESS: Log certificate request & results
CloudWatch Logs
Log Forwarder
AWS KMS
8. BLESS: Return a short lived certificate
10. sshd: Validate certificate, log certificate info
kafka
Bastion
1. SSH: Auth to Bastion
2. AWS SSH tool: Take request, determine user, application, instance
4. Sshaman Daemon: Determine calling user information. Use session credentials to request a certificate.
9. AWS ssh tool: ssh with certificate
Instances
sshd Logs
RELP Server (syslog)
3. Pilgrim: Generate Keypair Request SSH Cert
Developer
Developer Userspace
Daemon User
Sshaman Logs
Pilgrim Logs

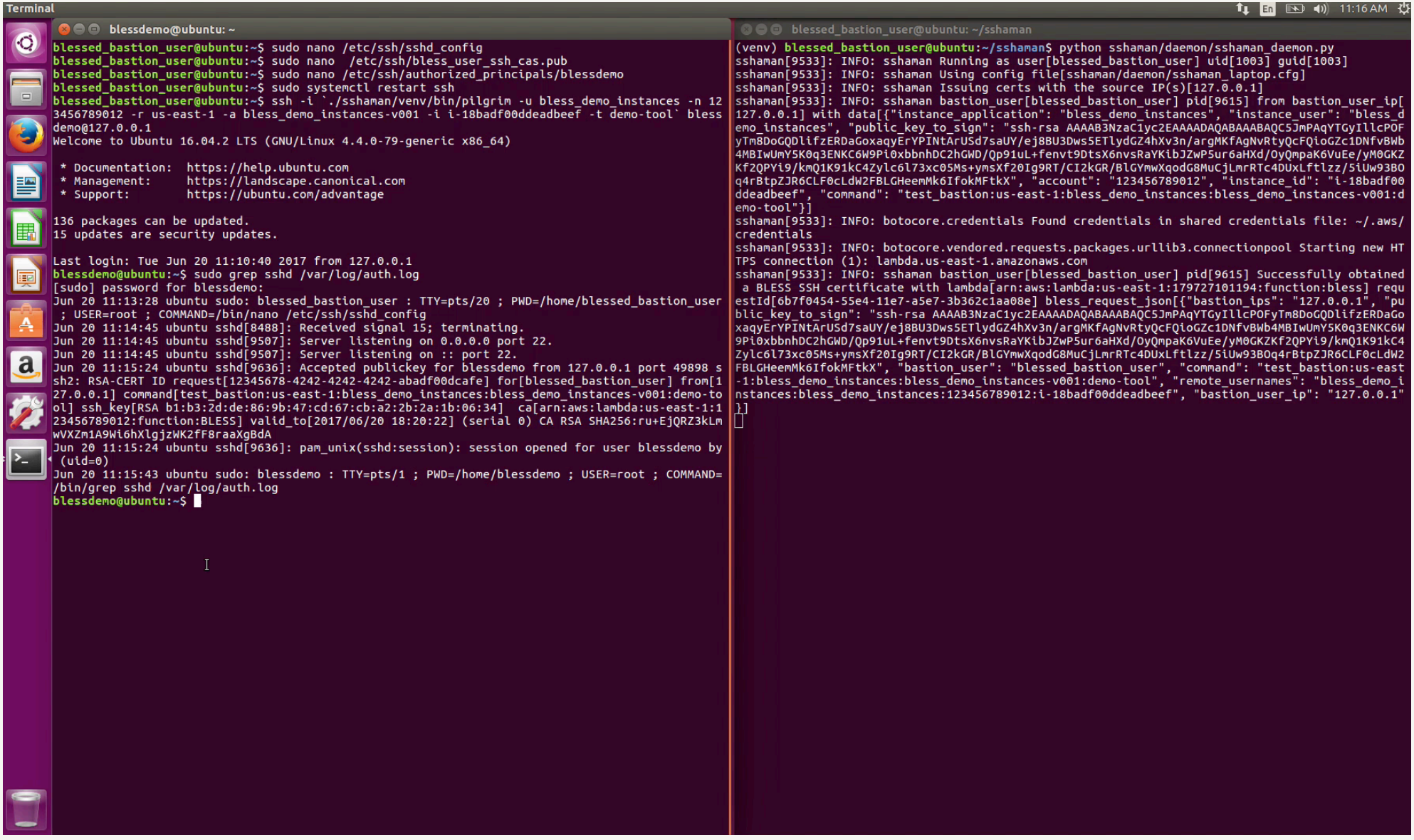

5. AWS ssh tool: Use session credentials to request a certificate
7. BLESS: Generate and sign SSH Certificate
6. BLESS: Decrypt SSH CA private key with KMS
BLESS
BLESS: Log certificate request & results
CloudWatch Logs
Log Forwarder
AWS KMS
8. BLESS: Return a short lived certificate
10. sshd: Validate certificate, log certificate info
kafka
Bastion
1. SSH: Auth to Bastion
2. AWS SSH tool: Take request, determine user, application, instance
4. Sshaman Daemon: Determine calling user information. Use session credentials to request a certificate.
9. AWS ssh tool: ssh with certificate
Instances
sshd Logs
RELP Server (syslog)
3. Pilgrim: Generate Keypair Request SSH Cert
Developer
Developer Userspace
Daemon User
Sshaman Logs
Pilgrim Logs


Cryptex Dev Xenial VM (1)
Terminal
9:40 AM
blessed_bastion_user@ubuntu: ~
blessed_bastion_user@ubuntu:~$ ./sshaman/venv/bin/pilgrim -u bless_demo_instances -n 123456789012 -r us-east-1 -a bless_demo_instances-v001 -i i-18badf00ddeadbeef -t demo-tool
/tmp/pilgrim/pilgrim-othnML/id_rsa
blessed_bastion_user@ubuntu:~$ ssh-keygen -L -f /tmp/pilgrim/pilgrim-othnML/id_rsa-cert.pub
/tmp/pilgrim/pilgrim-othnML/id_rsa-cert.pub:
Type: ssh-rsa-cert-v01@openssh.com user certificate
Public key: RSA-CERT SHA256:8k+O47Zyb8vKZzXuMjd61pKIE8mUVVT85PRiwYi7SoA
Signing CA: RSA SHA256:ru+EjQRZ3kLmwVXZm1A9Wi6hXlgjzWK2fF8raaXgBdA
Key ID: "request[12345678-4242-4242-4242-abadf00dcafe] for[blessed_bastion_user] from[127.0.0.1] command[test_bastion:us-east-1:bless_demo_instances:bless_demo_instances-v001:demo-tool] ssh key[RSA 1f:4b:be:fb:4a:55:62:65:e3:86:69:60:5d:0a:84:a1] ca[arn:aws:lambda:us-east-1:123456789012:function:BLESS] valid_to[2017/06/20 16:43:26]"
Serial: 0
Valid: from 2017-06-20T09:33:26 to 2017-06-20T09:43:26
Principals:
bless_demo_instances:bless_demo_instances:123456789012:i-18badf00ddeadbeef
Critical Options:
source-address 127.0.0.1
Extensions:
permit-port-forwarding
permit-pty
permit-user-rc
blessed_bastion_user@ubuntu:~$
blessed_bastion_user@ubuntu: ~/sshaman
(venv) blessed_bastion_user@ubuntu:~/sshaman$ python sshaman/daemon/sshaman_daemon.py
sshaman[27450]: INFO: sshaman Running as user[blessed_bastion_user] uid[1003] guid[1003]
sshaman[27450]: INFO: sshaman Using config file[sshaman/daemon/sshaman_laptop.cfg]
sshaman[27450]: INFO: sshaman Issuing certs with the source IP(s)[127.0.0.1]
sshaman[27450]: INFO: sshaman bastion_user[blessed_bastion_user] pid[27530] from bastion_user_ip[127.0.0.1] with data[{"instance_application": "bless_demo_instances", "instance_user": "bless_demo_instances", "public_key_to_sign": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCoiajwOTLnIj2pkYx+1OP6YspAqB87dOZwO0LOIKKY+W+eeLXOFQT5FfiaETXCHEaCuPIAVfPcT5U63lmOaPmYD7QXyG6xF9kmgpg6Fw+5ZY8FXzQL7tI1+MO6bf2aZxssM9rAjuATNvihOibdfsLq5GEY5Um9HtTE2xOO0S13t73ciXW8e7rqf2M/o6d0hhneE8Tu/J5xeUdcNxAcGZMvilRnMtkV1DqZddF5pJFWM4JAhBmJLb450JDgafu23yf41rvXCdULxCfQjeLUUD1JKCp9RnQDJ1nhuuVFmyL21UrQ/fSfBs8CsG7U+Uowzo7iDvWsOYG7UdqI5Nsvf8+b", "account": "123456789012", "instance_id": "i-18badf00ddeadbeef", "command": "test_bastion:us-east-1:bless_demo_instances:bless_demo_instances-v001:demo-tool"}]
sshaman[27450]: INFO: botocore.credentials Found credentials in shared credentials file: ~/.aws/credentials
sshaman[27450]: INFO: botocore.vendored.requests.packages.urllib3.connectionpool Starting new HTTPS connection (1): lambda.us-east-1.amazonaws.com
sshaman[27450]: INFO: sshaman bastion_user[blessed_bastion_user] pid[27530] Successfully obtained a BLESS SSH certificate with lambda[arn:aws:lambda:us-east-1:179727101194:function:bless] requestId[e242cc03-55d6-11e7-b1b4-b7e8cba72d31] bless_request_json[{"bastion_ips": "127.0.0.1", "public_key_to_sign": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCoiajwOTLnIj2pkYx+1OP6YspAqB87dOZwO0LOIKKY+W+eeLXOFQT5FfiaETXCHEaCuPIAVfPcT5U63lmOaPmYD7QXyG6xF9kmgpg6Fw+5ZY8FXzQL7tI1+MO6bf2aZxssM9rAjuATNvihOibdfsLq5GEY5Um9HtTE2xOO0S13t73ciXW8e7rqf2M/o6d0hhneE8Tu/J5xeUdcNxAcGZMvilRnMtkV1DqZddF5pJFWM4JAhBmJLb450JDgafu23yf41rvXCdULxCfQjeLUUD1JKCp9RnQDJ1nhuuVFmyL21UrQ/fSfBs8CsG7U+Uowzo7iDvWsOYG7UdqI5Nsvf8+b", "bastion_user": "blessed_bastion_user", "command": "test_bastion:us-east-1:bless_demo_instances:bless_demo_instances-v001:demo-tool", "remote_usernames": "bless_demo_instances:bless_demo_instances:123456789012:i-18badf00ddeadbeef", "bastion_user_ip": "127.0.0.1"}]

Instance SSHd Setup


5. AWS ssh tool: Use session credentials to request a certificate
7. BLESS: Generate and sign SSH Certificate
6. BLESS: Decrypt SSH CA private key with KMS
BLESS
BLESS: Log certificate request & results
CloudWatch Logs
Log Forwarder
AWS KMS
8. BLESS: Return a short lived certificate
10. sshd: Validate certificate, log certificate info
kafka
Bastion
1. SSH: Auth to Bastion
2. AWS SSH tool: Take request, determine user, application, instance
4. Sshaman Daemon: Determine calling user information. Use session credentials to request a certificate.
9. AWS ssh tool: ssh with certificate
Instances
sshd Logs
RELP Server (syslog)
3. Pilgrim: Generate Keypair Request SSH Cert
Developer
Developer Userspace
Daemon User
Sshaman Logs
Pilgrim Logs


5. AWS ssh tool: Use session credentials to request a certificate
7. BLESS: Generate and sign SSH Certificate
6. BLESS: Decrypt SSH CA private key with KMS
BLESS
BLESS: Log certificate request & results
CloudWatch Logs
Log Forwarder
AWS KMS
8. BLESS: Return a short lived certificate
10. sshd: Validate certificate, log certificate info
kafka
Bastion
1. SSH: Auth to Bastion
2. AWS SSH tool: Take request, determine user, application, instance
4. Sshaman Daemon: Determine calling user information. Use session credentials to request a certificate.
9. AWS ssh tool: ssh with certificate
Instances
sshd Logs
RELP Server (syslog)
3. Pilgrim: Generate Keypair Request SSH Cert
Developer
Developer Userspace
Daemon User
Sshaman Logs
Pilgrim Logs
HIVE


Terminal
11:16 AM
blessdemo@ubuntu: ~
blessdemo@ubuntu: ~
blessed_bastion_user@ubuntu:~$ sudo nano /etc/ssh/sshd_config
blessed_bastion_user@ubuntu:~$ sudo nano /etc/ssh/bless_user_ssh_cas.pub
blessed_bastion_user@ubuntu:~$ sudo nano /etc/ssh/authorized_principals/blessdemo
blessed_bastion_user@ubuntu:~$ sudo systemctl restart ssh
blessed_bastion_user@ubuntu:~$ ssh -i `./sshaman/venv/bin/pilgrim -u bless_demo_instances -n 123456789012 -r us-east-1 -a bless_demo_instances-v001 -i i-18badf00ddeadbeef -t demo-tool` blessdemo@127.0.0.1
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-79-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
136 packages can be updated.
15 updates are security updates.
Last login: Tue Jun 20 11:10:40 2017 from 127.0.0.1
blessdemo@ubuntu:~$ sudo grep sshd /var/log/auth.log
[sudo] password for blessdemo:
Jun 20 11:13:28 ubuntu sudo: blessed_bastion_user : TTY=pts/20 ; PWD=/home/blessed_bastion_user ; USER=root ; COMMAND=/bin/nano /etc/ssh/sshd_config
Jun 20 11:14:45 ubuntu sshd[8488]: Received signal 15; terminating.
Jun 20 11:14:45 ubuntu sshd[9507]: Server listening on 0.0.0.0 port 22.
Jun 20 11:14:45 ubuntu sshd[9507]: Server listening on :: port 22.
Jun 20 11:15:24 ubuntu sshd[9636]: Accepted publickey for blessdemo from 127.0.0.1 port 49898 ssh2: RSA-CERT ID request[12345678-4242-4242-4242-abadf00dcafe] for[blessed_bastion_user] from[127.0.0.1] command[test_bastion:us-east-1:bless_demo_instances:bless_demo_instances-v001:demo-tool] ssh_key[RSA b1:b3:2d:de:86:9b:47:cd:67:cb:a2:2b:2a:1b:06:34] ca[arn:aws:lambda:us-east-1:123456789012:function:BLESS] valid_to[2017/06/20 18:20:22] (serial 0) CA RSA SHA256:ru+EjQRZ3kLmwVXZm1A9Wi6hXlgjzWK2fF8raaXgBdA
Jun 20 11:15:24 ubuntu sshd[9636]: pam_unix(sshd:session): session opened for user blessdemo by (uid=0)
Jun 20 11:15:43 ubuntu sudo: blessdemo : TTY=pts/1 ; PWD=/home/blessdemo ; USER=root ; COMMAND=/bin/grep sshd /var/log/auth.log
blessdemo@ubuntu:~$
blessed_bastion_user@ubuntu: ~/sshaman
(venv) blessed_bastion_user@ubuntu:~/sshaman$ python sshaman/daemon/sshaman_daemon.py
sshaman[9533]: INFO: sshaman Running as user[blessed_bastion_user] uid[1003] guid[1003]
sshaman[9533]: INFO: sshaman Using config file[sshaman/daemon/sshaman_laptop.cfg]
sshaman[9533]: INFO: sshaman Issuing certs with the source IP(s)[127.0.0.1]
sshaman[9533]: INFO: sshaman bastion_user[blessed_bastion_user] pid[9615] from bastion_user_ip[127.0.0.1] with data[{"instance_application": "bless_demo_instances", "instance_user": "bless_demo_instances", "public_key_to_sign": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC5JmPAqYTGyIllcPOFyTm8DoGQDlifzERDaGoxaqyErYPINtArUSd7saUY/ej8BU3Dws5ETlydGZ4hXv3n/argMKfAgNvRtyQcFQioGZc1DNfvBWb4MBIwUmY5K0q3ENKC6W9Pi0xbbnhDC2hGWD/Qp91uL+fenvt9DtsX6nvsRaYKibJZwP5ur6aHXd/OyQmpaK6VuEe/yM0GKZKf2QPYi9/kmQ1K91kC4Zylc6l73xc05Ms+ymsXf20Ig9RT/CI2kGR/BlGYmwXqodG8MuCjLmrRTc4DUxLftlzz/5iUw93BOq4rBtpZJR6CLF0cLdW2FBLGHeemMk6IfokMFtkX", "account": "123456789012", "instance_id": "i-18badf00ddeadbeef", "command": "test_bastion:us-east-1:bless_demo_instances:bless_demo_instances-v001:demo-tool"}]
sshaman[9533]: INFO: botocore.credentials Found credentials in shared credentials file: ~/.aws/credentials
sshaman[9533]: INFO: botocore.vendored.requests.packages.urllib3.connectionpool Starting new HTTPS connection (1): lambda.us-east-1.amazonaws.com
sshaman[9533]: INFO: sshaman bastion_user[blessed_bastion_user] pid[9615] Successfully obtained a BLESS SSH certificate with lambda[arn:aws:lambda:us-east-1:179727101194:function:bless] requestId[6b7f0454-55e4-11e7-a5e7-3b362c1aa08e] bless_request_json[{"bastion_ips": "127.0.0.1", "public_key_to_sign": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC5JmPAqYTGyIllcPOFyTm8DoGQDlifzERDaGoxaqyErYPINtArUSd7saUY/ej8BU3Dws5ETlydGZ4hXv3n/argMKfAgNvRtyQcFQioGZc1DNfvBWb4MBIwUmY5K0q3ENKC6W9Pi0xbbnhDC2hGWD/Qp91uL+fenvt9DtsX6nvsRaYKibJZwP5ur6aHXd/OyQmpaK6VuEe/yM0GKZKf2QPYi9/kmQ1K91kC4Zylc6l73xc05Ms+ymsXf20Ig9RT/CI2kGR/BlGYmwXqodG8MuCjLmrRTc4DUxLftlzz/5iUw93BOq4rBtpZJR6CLF0cLdW2FBLGHeemMk6IfokMFtkX", "bastion_user": "blessed_bastion_user", "command": "test_bastion:us-east-1:bless_demo_instances:bless_demo_instances-v001:demo-tool", "remote_usernames": "bless_demo_instances:bless_demo_instances:123456789012:i-18badf00ddeadbeef", "bastion_user_ip": "127.0.0.1"}]

Related Work

- Lyft
  - Uses BLESS with client that runs on laptops
  - https://eng.lyft.com/blessing-your-ssh-at-lyft-a1b38f81629d
- Facebook
  - Leverages signed certificates with principals
  - https://code.facebook.com/posts/365787980419535/scalable-and-secure-access-with-ssh/
- Wikimedia
  - SSH-agent proxy to protect private key on bastion
  - https://blog.wikimedia.org/2017/03/22/keyholder/

Questions?

bryanp@netflix.com
https://bryanpayne.org

[PS... I'm hiring!]