PKI at Scale Using Short-Lived Certificates

Bryan D. Payne Engineering Manager, Platform Security
Elastic Load Balancers

Web Service

Web Service

Web Service

Web Service

...
Securely Deploy Certificate / Key
Communicate Securely

API & UI for Certificate Creation
Lemur

Get Certificate & Key
Private CA
CloudCA
Public CA

Seal Secrets
Metatron

Deployment Management
Spinnaker

Version Control
Git

AMI

Server with TLS
Karyon
Tomcat
Apache

Metatron

Client with TLS
Ribbon
Revocation Is Hard
CRL (rfc2459)
OCSP (rfc2560)
OCSP stapling (rfc6066)
OCSP must staple (draft-hallambaker-muststaple-00)
CRL: Certificate Revocation List

1: TLS Handshake

2: Check CRL

Browser

Web Server (Content)

Web Server (CRL)

Update

Certificate Authority

Internet

CRL Cache
CRL (rfc2459)
OCSP (rfc2560)
OCSP stapling (rfc6066)
OCSP must staple (draft-hallambaker-muststaple-00)
OCSP: Online Certificate Status Protocol

1: TLS Handshake
2: Get Certificate Status

Browser

Web Server (Content)

OCSP Responder

Update

Certificate Authority

Internet
CRL (rfc2459)
OCSP (rfc2560)
OCSP stapling (rfc6066)
OCSP must staple (draft-hallambaker-muststaple-00)
OCSP Stapling

1: TLS Handshake
2: Return Certificate Status

Web Server (Content)

OCSP Responder

Certificate Authority

Update

Internet

Browser
CRL \((\text{rfc2459})\)

OCSP \((\text{rfc2560})\)

OCSP stapling \((\text{rfc6066})\)

OCSP must staple \((\text{draft-hallambaker-muststaple-00})\)
<table>
<thead>
<tr>
<th></th>
<th>OCSP must-staple</th>
<th>OCSP staple</th>
<th>OCSP</th>
<th>CRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JavaScript</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Short-Lived Certificates

AWS HMAC Generation

Lifecycle of AccessKeyID and SecretKey is of utmost interest here.

Not real secret keys, sorry.
Circa 2012: AWS SDKs Introduce the Provider Paradigm

```java
// provider paradigm dynamically asks for keys every time
AWSCredentialsProvider prov = new AWSCredentialsProvider(){
    public AWSCredentials getCredentials(){
        RESTfulObj AWSKey = RESTService.get("server/getAWSKey");
        return new BasicAWSCredentials(
            AWSKey.getAccessID(), AWSKey.getSecretKey());
    }
};

AmazonSimpleDBClient client = new AmazonSimpleDBClient(prov);
client.listDomains();
```

The `client` object in the above code example no longer caches keys.
On Instance Credentials

```
{
  "Code" : "Success",
  "LastUpdated" : "2015-09-17T01:29:49Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAI6IJJCXLEXAMPLE",
  "SecretAccessKey" : "iXKQe8qXbhnN0jUe7JGVqFNXmTnP5pI6example",
  "Token" : "...",
  "Expiration" : "2015-09-17T07:47:45Z"
}
```
Alice sends Bob an ID Proof + Credential Request.

Bob validates Alice's ID and generates a New Short-Lived Credential.

Alice receives the New Short-Lived Credential.
Don’t use short-lived cred to get updated cred!
Linux Kernel (with AppArmor or SELinux)

Credential Management Process

Credential Renewal Protocol

System Identity Credentials (TPM or SGX)

(write)

Service with TLS

TLS Session

Short-Lived Certificate and Key Files

(read)

Linux Kernel (with AppArmor or SELinux)
Loading new certificates into service…

- Send signal to service
- Restart service
- Design service to reload certificates periodically

Service with TLS

TLS Session

Short-Lived Certificate and Key Files

(write)  (read)

Level (with AppArmor or SELinux)
<table>
<thead>
<tr>
<th>Service</th>
<th>How to load a new certificate and private key?</th>
<th>Zero downtime?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>graceful restart</td>
<td>Maybe</td>
</tr>
<tr>
<td>Nginx</td>
<td>reload</td>
<td>Yes</td>
</tr>
<tr>
<td>Tomcat</td>
<td>restart</td>
<td>No</td>
</tr>
<tr>
<td>HAPProxy</td>
<td>reload</td>
<td>No</td>
</tr>
<tr>
<td>Stunnel</td>
<td>HUP</td>
<td>No</td>
</tr>
<tr>
<td>Ghostunnel</td>
<td>SIGUSR1</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Develop & Deploy Code

Version Control
Git

Deployment Management
Spinnaker

AMI

Initialize Secrets
Metatron

API & UI for Certificate Creation
Lemur

Private CA
CloudCA

Public CA

Provision Credentials at Startup

Deployment Management

Spinnaker

AMI

Initialize Secrets
Metatron

API & UI for Certificate Creation
Lemur

Private CA
CloudCA

Public CA

Provision Credentials at Startup
Develop & Deploy Code

Communicate Securely

Provision Credentials at Startup

Version Control
- **Git**

Deployment Management
- **Spinnaker**

AMI

Initialize Secrets
- **Metatron**

API & UI for Certificate Creation
- **Lemur**

Private CA
- **CloudCA**
- **Public CA**

Server with TLS
- **Metatron**
- **Karyon**
- **Tomcat**
- **Nginx**

Client with TLS
- **Metatron**
- **Ribbon**

Deployment Management

Spinnaker

Client with TLS

Ribbon

Server with TLS

Metatron

API & UI

for Certificate Creation

Lemur

Public CA

Private CA

CloudCA

Version Control

Git

Lemur

Metatron

Private CA

Public CA

CloudCA

Spinnaker

AMI

Server with TLS

Metatron

Karyon

Tomcat

Nginx

Client with TLS

Metatron

Ribbon

Deployment Management

Spinnaker

Client with TLS

Ribbon

Server with TLS

Metatron

Karyon

Tomcat

Nginx

Communicate Securely
<table>
<thead>
<tr>
<th>Long-Lived Certificates</th>
<th>Short-Lived Certificates</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Improve attack detection, in practice</td>
<td>• Refresh certificates</td>
</tr>
<tr>
<td>• Retrofit your applications to support revocation</td>
<td>• Update server / client to support graceful reloading of certificates</td>
</tr>
</tbody>
</table>
From Vision to Reality...
Questions?

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

[PS... I’m hiring!]