.conf2015

Best Practices for Splunk SSL

Duane Waddle

Defense Point Security

splunk>

About me and DPS

- Duane
  - Security Engineer at Defense Point Security
  - Splunk admin since 2010, Splunk Certified Architect
  - Occasionally masquerade as a fez-wearing duck on the Interwebs
- Defense Point Security
  - Provider of cyber security services to commercial and gov't clients
  - 2014 Splunk Partner of the year, ProServ Public Sector
  - We're hiring! (https://www.defpoint.com)

DEFENSE POINT SECURITY

SSL Refresher

- Authentication of the server (the server is who they say they are)
- Optional authentication of the client
- Bulk encryption of data in transit
- Several moving parts, “CAs”, “keys”, “CSRs”, “certs”
- We often say "SSL" when we mean "TLS". True SSL is effectively dead.

Splunk Architecture and SSL

- Splunkweb (SSL to browsers)
- Splunk-to-splunk data transfer (forwarders to indexers)
- Splunkd REST port (Inter-Splunk)
  - Deployment Client / Deployment Server
  - REST API / SDKs
  - Distributed Search
- LDAP connections
- Clustering

Splunk's default SSL posture

- The out-of-the-box configuration:
  - All certificates are generated on a default-shipped CA configuration
  - Splunkweb does not use SSL
  - Splunkd uses SSL for the REST port - with certificate verification disabled
  - No SSL data inputs/outputs are defined
  - Splunkd LDAP can use SSL - again with no certificate verification

http://docs.splunk.com/Documentation/Splunk/latest/Security/AboutsecuringyourSplunkconfigurationwithSSL

Type of exchange	Client function	Server function	Encryption	Certificate Authentication	Common Name checking	Type of data exchanged
Browser to Splunk Web	Browser	Splunk Web	**NOT enabled by default**	dictated by client (browser)	dictated by client (browser)	search term results
Inter-Splunk communication	Splunk Web	splunkd	enabled by default	**NOT enabled by default**	**NOT enabled by default**	search term results
Forwarding	splunkd as a forwarder	splunkd as an indexer	**NOT enabled by default**	**NOT enabled by default**	**NOT enabled by default**	data to be indexed
Inter-Splunk communication	splunkd as a deployment client	splunkd as deployment server	enabled by default	**NOT enabled by default**	**NOT enabled by default**	configuration data
Inter-Splunk communication	splunkd as a search head	splunkd as search peer	Enabled by default	**NOT enabled by default**	**NOT enabled by default**	search data

Why this stuff matters

- A DPS penetration tester found himself on a random Linux box as an unprivileged user. This box was:
  - Running Splunk forwarder as root
    - ... with the default admin/changeme password
    - ... and default SSL configs, trusting any certificate

Why this stuff matters

- He was able to:
  - Use the REST API to change the deployment server IP (to his box)
  - Restart the forwarder
  - Download an app to the forwarder that started a reverse root shell
  - Pivot from root on that box to downloading the site's Chef repo
  - Lift a copy of all of their recipes, including AWS API keys

Moral of the story:

Whoever controls your DS controls the users running your forwarders.

(some) Best Practices Checklist

- ☑ Run Splunk forwarders as an unprivileged user
- ☑ Change forwarder admin passwords
- ☑ Enable strong SSL authentication between DS client and DS server
- ☑ Use host based firewall to limit outbound connections to trusted IPs
- ☑ Pick an appropriate cipherSuite
- ☑ Use wildcard SSL certs with caution
- ☑ Decide on FIPS mode early on and talk to Splunk first

Commercial CA or Private CA?

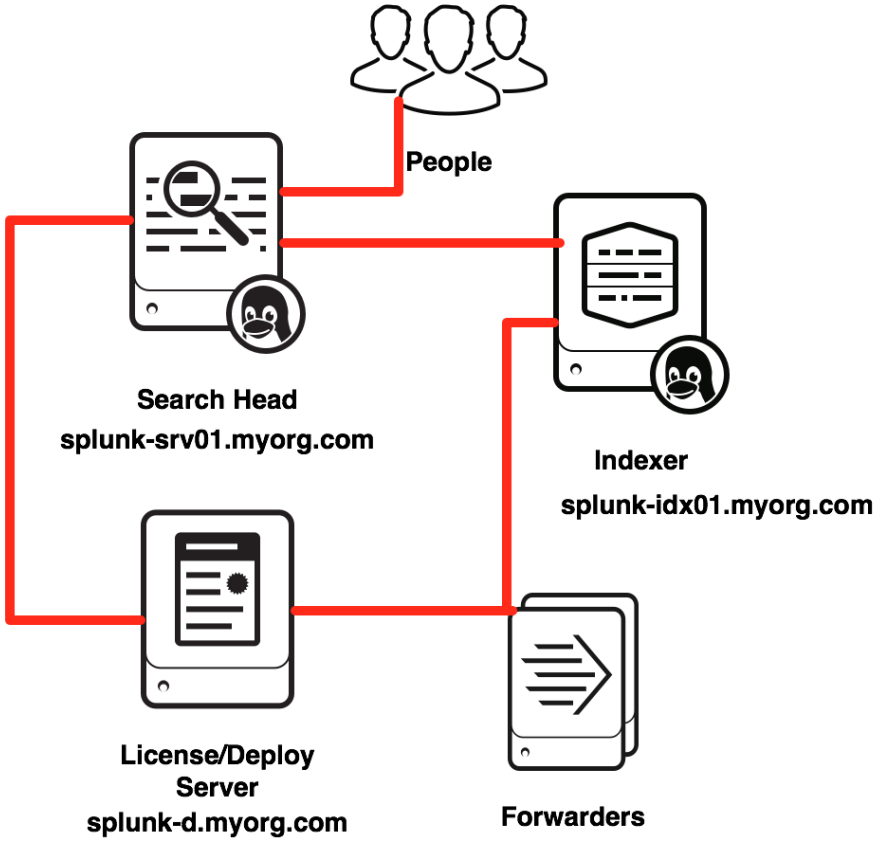
- Commercial
  - Root certs are in everyone's browser already
  - Costs real money (potentially a LOT if you use ECC)
  - Potential renewal nightmare
- Private
  - You have to run a CA (likely already are...)
  - Free (ish)
  - Root certs must be distributed
  - You can do very long expirations (in theory)

How many certs do I need?

- Splunkweb - Search Head:
  - A **3rd party** CA cert and its root / intermediates
- Splunkd:
  - A root cert and its intermediates - (either 3rd party or private)
  - One per Splunk Server non Search Head

    (or one per role in large envs)
- One throwaway certificate for all of the Splunk UFs to share

Our Example Architecture



Create Splunk Server Key & CSR

```
$ mkdir $SPLUNK_HOME/etc/auth/myOrg
$ cd $SPLUNK_HOME/etc/auth/myOrg
$ openssl req -nodes -newkey rsa:2048 -keyout
   splunk-srv1.web.key -out splunk-srv1.csr
$ openssl rsa -in splunk-srv1.web.key -des3 -out
   splunk-srv1.key
```

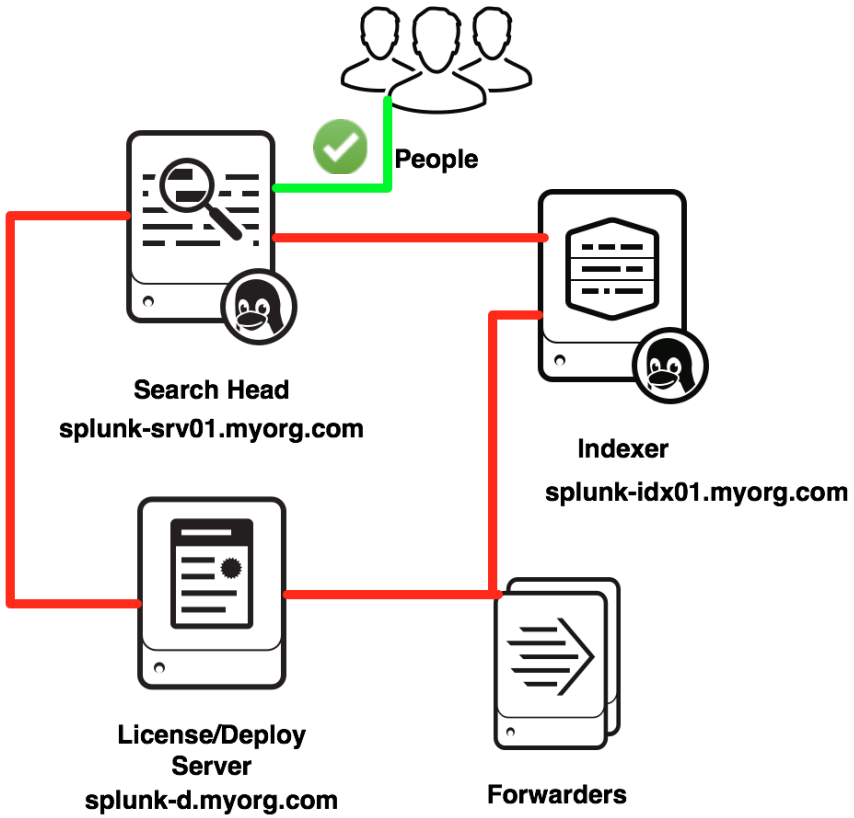
Get the CA root certificate chain and put in `auth/myOrg` as `cacert.crt`.

Copy the CA-returned crt file to `auth/myOrg/splunk-srv1.crt`

The Search Head - Splunk Web

```
$ cd $SPLUNK_HOME/etc/auth/myOrg
$ cat splunk-srv1.crt cacert.crt > splunk-srv1.web.pem
$ vi $SPLUNK_HOME/etc/system/local/web.conf
[settings]
enableSplunkWebSSL = 1
httpport = 8443
privKeyPath = etc/auth/myOrg/splunk-srv1.web.key
caCertPath = etc/auth/myOrg/splunk-srv1.web.pem
sslVersions = tls, -tls1.0
cipherSuite = ?
```

Architecture - Status



Indexers

First make certificates as you would for SplunkWeb.

```
$ cd $SPLUNK_HOME/etc/auth/myOrg
$ openssl req -nodes -newkey rsa:2048 -keyout splunk-idx01.key -out splunk-idx01.csr
$ openssl rsa -in splunk-idx01.web.key -des3 -out
   splunk-idx01.key
```

To make the Indexer formatted .PEM:

```
$ cat splunk-idx01.crt splunk-idx01.key cacert.crt > splunk-idx01.pem
```

The Indexer - Inputs.conf

```
vi $SPLUNK_HOME/etc/system/local/inputs.conf


[splunktcp-ssl://9998]
disabled = 0


[SSL]
password = <REDACTED>
rootCA = $SPLUNK_HOME/etc/auth/myOrg/cacert.crt
serverCert = $SPLUNK_HOME/etc/auth/myOrg/splunk-idx01.pem
sslVersions = tls, -tls1.0
cipherSuite = ?
requireClientCert = true|false
```

The Forwarder

First make certificates as you would for SplunkWeb.
This can be done on your deployment server.

```
$ cd $SPLUNK_HOME/etc/auth/myOrg/forwarder
$ openssl req -nodes -newkey rsa:2048 -keyout splunk-forwarder.web.key -out splunk-forwarder.csr
```

On forwarders only, make the key password "password" for reasons ...

```
$ openssl rsa -in splunk-forwarder.web.key -des3 -out splunk-forwarder.key
```

Throw away splunk-forwarder.web.key

```
$ cat splunk-forwarder.crt splunk-forwarder.key cacert.crt > splunk-forwarder.pem
```

Copy the splunk-forwarder.pem and cacert.crt to your Forwarder(s): Yes you could use an APP for this.

Forwarder to Indexer - Outputs.conf

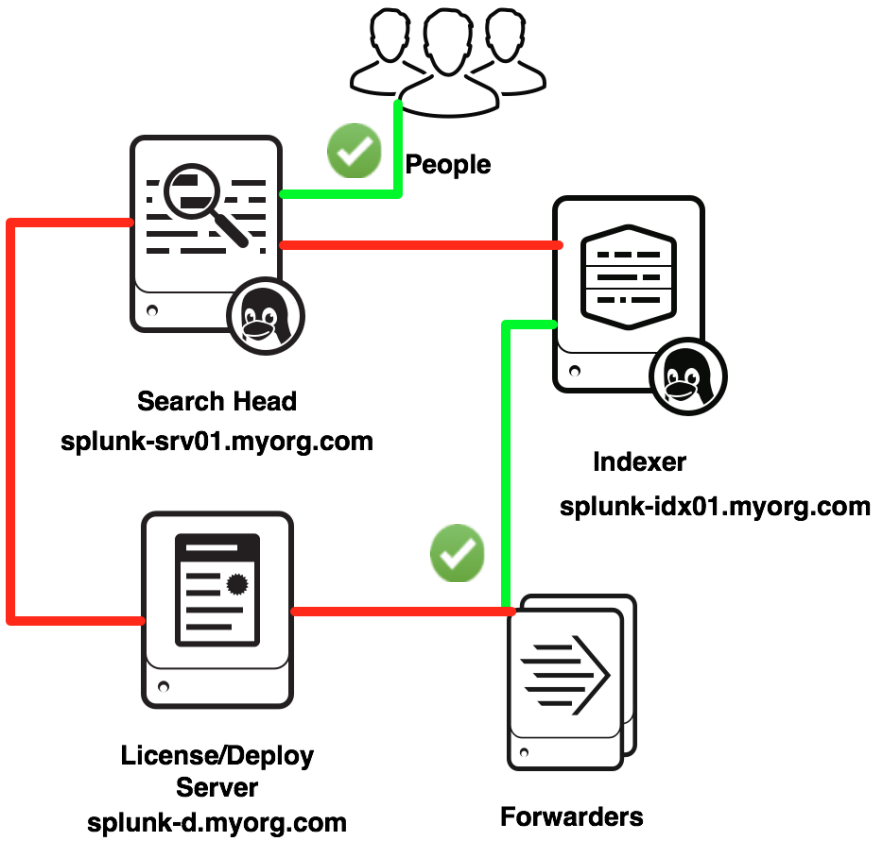
```
vi $SPLUNK_HOME/etc/system/local/outputs.conf

[tcpout]
defaultGroup = myIndexers

[tcpout:myIndexers]
server = splunk-idx01.myorg.com:9998
sslCertPath = $SPLUNK_HOME/etc/auth/myOrg/splunk-forwarder.pem
sslPassword = password  # For Reasons
sslRootCAPath = $SPLUNK_HOME/etc/auth/myOrg/cacert.crt
sslVerifyServerCert = true
sslCommonNameToCheck = splunk-idx01.myorg.com
```

Gotcha - Forwarder to Indexer

- If you mistype the `sslRootCAPath` argument in outputs.conf, the forwarder will default to not-SSL when trying to talk to indexer. The error on the indexer will look like the following:

```
6-23-2014 20:46:48.918 +0000 ERROR TcpInputProc - Error
encountered for connection from src=10.0.1.57:41778. error:
140760FC:SSL routines:SSL23_GET_CLIENT_HELLO:unknown
protocol
```

Architecture - Status



The Deployment Server

First make certificates as you would for SplunkWeb.

```
$ cd $SPLUNK_HOME/etc/auth/myOrg
$ openssl req -nodes -newkey rsa:2048 -keyout splunk-d.web.key -out splunk-d.csr
$ openssl rsa -in splunk-d.web.key -des3 -out splunk-d.key
```

To make the Deployment Server formatted pem:

```
$ cat splunk-d.crt splunk-d.key cacert.crt > splunk-d.pem
```

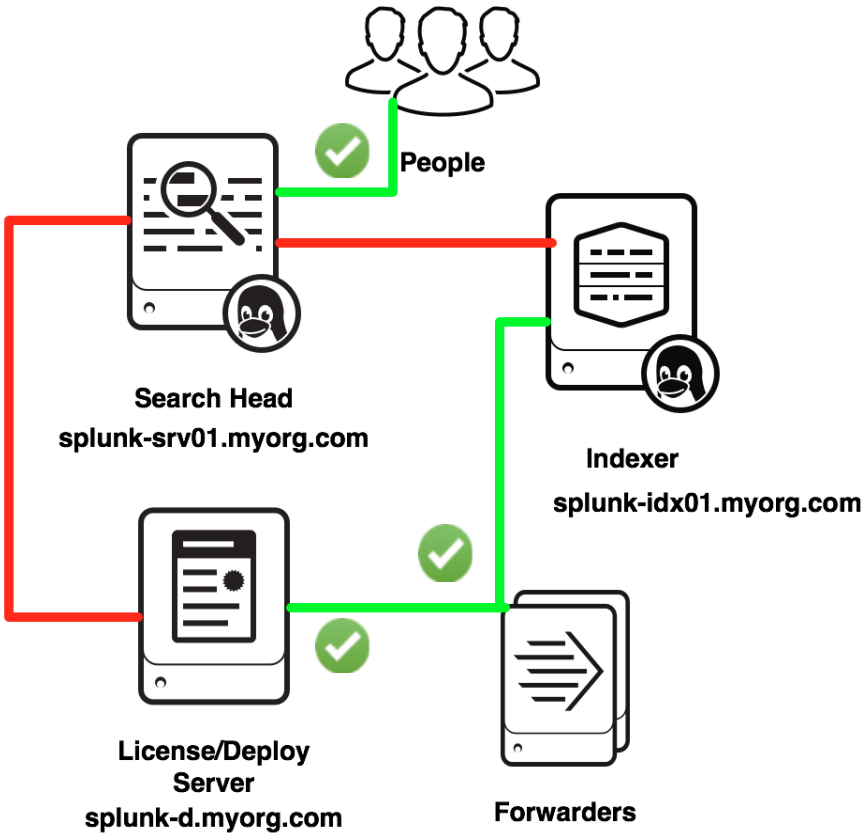
The Deployment Server - server.conf

```
vi $SPLUNK_HOME/etc/system/local/server.conf

[sslConfig]
caCertFile = cacert.crt
caPath = $SPLUNK_HOME/etc/auth/myOrg
sslKeysfile = splunk-d.pem
sslKeysfilePassword = <REDACTED>
sslVersions = tls, -tls1.0
cipherSuite = ?
requireClientCert = false
```

Splunk Forwarder - DS Clients

```
vi $SPLUNK_HOME/etc/system/local/server.conf

[sslConfig]
caCertFile = cacert.crt
caPath = $SPLUNK_HOME/etc/auth/myOrg
sslKeysfile = splunk-forwarder.pem
sslKeysfilePassword = password  # Reasons
sslVersions = tls, -tls1.0
sslVerifyServerCert = true
sslCommonNameToCheck = splunk-d.myorg.com
```

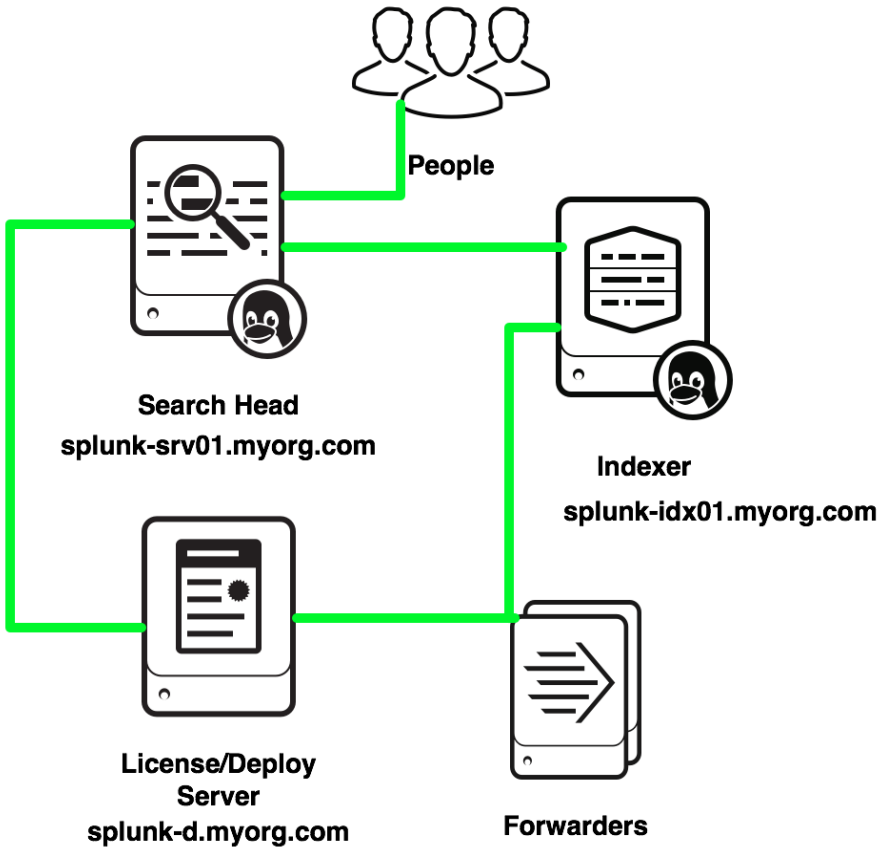
Architecture - Status



Splunk - Server to Server

```
vi $SPLUNK_HOME/etc/system/local/server.conf

[sslConfig]
caCertFile = cacert.crt
caPath = $SPLUNK_HOME/etc/auth/myOrg
sslKeysfile = splunk-srvXX.pem
sslKeysfilePassword = <REDACTED>
sslVersions = tls, -tls1.0
cipherSuite = ?
requireClientCert = false
sslVerifyServerCert = true
sslCommonNameList = splunk-srv01.myorg.com, splunk-d.myorg.com, splunk-idx01.myorg.com, splunk-idx02.myorg.com, ...
```

Architecture - Status



Splunk LDAPS 

Each LDAP strategy has an SSL toggle on/off

- In GUI, it's a checkbox
- In `authentication.conf`, each LDAP stanza needs `SSLEnabled=1`

Minimum Certificate settings in `$SPLUNK_HOME/etc/openldap/ldap.conf`

```
TLS_REQCERT demand
TLS_CACERT /opt/splunk/etc/auth/LDAProotcert.crt
TLS_CIPHER_SUITE
```

(equivalent to cipherSuite)

Indexer Clustering

- Indexer clustering uses both REST API and a dedicated cluster data transfer port
  - Certs & config for REST API are all covered above
  - SSL signature and common name checking occur BEFORE pass4SymmKey checking
- Protip: If building a cluster from scratch, use the same `splunk.secret` on all cluster nodes
- Converting a cluster from default certs to production certs can be brittle
  - Enable sslVerifyServerCert and sslCommonNameList LAST
- sslCommonNameList needs to list all possible REST communications partners
  - All indexers, cluster master, license server, and search heads ...

Indexer Clustering - SSL Data Transfer

- Minimal documentation - only one reference to it in the docs
  - http://docs.splunk.com/Documentation/Splunk/latest/Admin/Serverconf
- In server.conf comment out `replication-port` stanza and add:

```
[replication_port-ssl://8002]
password = <REDACTED>
rootCA = $SPLUNK_HOME/etc/auth/myOrg/cacert.crt
serverCert = $SPLUNK_HOME/etc/auth/myOrg/splunk-idx01.pem
```

- Try this out in a test cluster first!
  - This is NOT a common setting in the wild

SHC and KVStore

- SHC - same REST port rules apply as with indexer clustering
- KVStore has its own SSL config stanza in `server.conf`:

```
[KVstore]
caCertPath = …
sslKeysPath = …
sslKeysPassword = …
```

Docs mention these ONLY work in FIPS mode - needs more testing

Thank You!

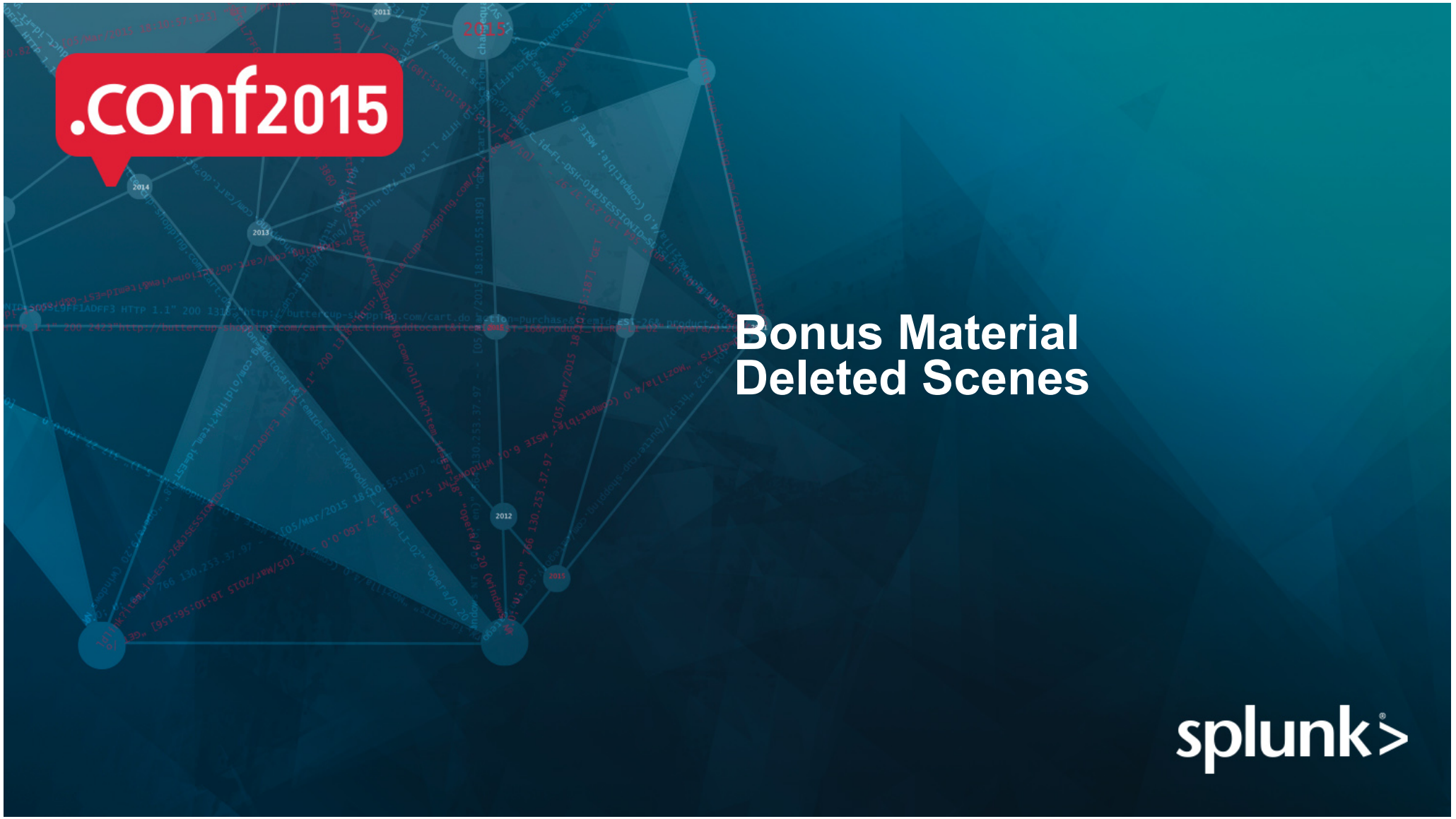
Other resources

Splunk IRC (EFNet #splunk)
Splunk Answers (http://answers.splunk.com)
Splunk community wiki (http://wiki.splunk.com)
Splunk User Group Slack (http://splunk-usergroups.slack.com)

http://www.georgestarcher.com/
http://www.duanewaddle.com/

And here's a plug for some other excellent .conf sessions...

Beyond the Lookup Glass
The 'State' of Splunk - Using the KVStore to Maintain App State
Creating and Using Custom Alert Actions
Optimizing Splunk Knowledge Objects - A Tale of Unintended Consequences
Hold Me Closer Tiny Data



Bonus Material
Deleted Scenes

splunk>

Be your own Certificate Authority

- We will use ECC crypto for higher performance
- Start out by making a CA Root key and certificate.
- Very helpful Splunk Blogs post by Jose Hernandez
  http://blogs.splunk.com/2014/06/03/generate-elliptical-curve-certkeys-for-splunk/

- You will be prompted for passphrases for multiple keys
  - Keep them secret
  - Keep them safe
  - Use a different passphrase for every key

Create the CA Root Key & Cert - ECC

```
$ cd $SPLUNK_HOME/etc/auth/myOrg
$ splunk cmd openssl ecparam  -name "prime256v1" -genkey |
    splunk cmd openssl ec -des3 -out CAroot.key
Enter PEM pass phrase: <abc123>
Verifying - Enter PEM pass phrase: <abc123>

$ splunk cmd openssl req -key CAroot.key -sha1 -subj
    "/CN=Splunk Root CA/O=myOrg" -new -x509 -days 3650
    -set_serial 1 -out cacert.crt
Enter pass phrase for CAroot.key: <abc123>
```

Create Splunk Server Key & CSR - ECC

```
$ splunk cmd openssl ecparam  -name "prime256v1" -genkey
     -out splunk-d.web.key
 $ splunk cmd openssl ec -des3 -in splunk-d.web.key
      -out splunk-d.key
Enter PEM pass phrase: <def234>
Verifying - Enter PEM pass phrase: <def234>
 $ splunk cmd openssl req -key splunk-d.key -subj
   "/CN=splunk-d.myorg.com/O=myOrg" -new -out
splunk-d.csr
```

Sign the Splunk Cert using Root Cert - ECC

```
$ splunk cmd openssl x509 -req -days 1095 -in splunk-d.csr
   -CA cacert.crt -CAkey CAroot.key -set_serial 02 -out
   splunk-d.crt
Signature ok
subject=/CN=splunk-d.myorg.com/O=myOrg
Getting CA Private Key
Enter pass phrase for CAroot.key: <abc123>
```

Now we have a keyfile (both encrypted and not) and a cert issued by our CA

Some other ways to be your own CA

- Active Directory Certificate Services
- Fedora Certificate Server
  - http://pki.fedoraproject.org/wiki/PKI_Main_Page
  - Also a part of the FreeIPA suite
    - (Commercially as Red Hat Directory Server)

Bonus Material

- Splunk Blog:
- http://blogs.splunk.com/2014/06/03/generate-elliptical-curve-certkeys-for-splunk/

- Troubleshooting:
- http://mikeberggren.com/post/28429473721/chain-check

Test connectivity with openssl s_client

- OpenSSL has a built-in SSL client that you can use to do basic connectivity testing.
- Works ‘just like TELNET’ but over SSL
- No certificate verification by default, but you can get it to dump the presented certs so you can check them by hand.
- It will also dump TLS protocol version and negotiated cipher specification
-
- `$ openssl s_client -connect 10.10.10.10:8089 -showcerts`
-
- The returned certs can be checked in plaintext by copypasting into a file and running
- `$ openssl x509 -text -noout -in xxxx.crt`

Forwarder to LB Indexers - Outputs.conf -1

```
vi $SPLUNK_HOME/etc/system/local/outputs.conf
(or use an app)

[tcpout]
defaultGroup = myIndexers

[tcpout:myIndexers]
maxQueueSize = 128MB
useACK = true
autoLB = true
server = splunk-idx01.myorg.com:9998, splunk-idx02.myorg.com:9998
sslCertPath = $SPLUNK_HOME/etc/auth/myOrg/splunk-forwarder.pem
sslPassword = <REDACTED>
sslRootCAPath = $SPLUNK_HOME/etc/auth/myOrg/cacert.crt
```

Forwarder to LB Indexers - Outputs.conf -2

```
vi $SPLUNK_HOME/etc/system/local/outputs.conf
(or use an app)

[splunk-idx01.myorg.com]
sslVerifyServerCert = true
sslCommonNameToCheck = splunk-idx01.myorg.com
[splunk-idx02.myorg.com]
sslVerifyServerCert = true
sslCommonNameToCheck = splunk-idx02.myorg.com
```

File formats can and will trip you up

- Different areas of Splunk use SSL key files / cert files formatted slightly differently

- Splunk always expects PEM encoded certs & keys
  - Some CAs will send DER and you'll have to convert
  - Some will send PKCS7, PKCS12, or even stranger files
- Splunkweb v6.1.x and older has CherryPy dependencies
  - SSL key file must be unencrypted
  - SSL key and SSL cert must be in separate files
- Splunkd expects key / cert / root-cert all in one file

Handling PKCS7 packaged certs

- Sometimes happens from a SSL admin grabbing certs from Comodo and often has the whole certificate chain. Yeah, this happened to George helping someone rebuild their Splunk.

```
Starts like:
-----BEGIN PKCS7-----
MIIOewYJKoZIhvc
```

- To change the format:

```
openssl pkcs7 -inform PEM -in $PKCS7_FILE -outform PEM -print_certs > splunk-srv1.pem
```

Copy the file splunk-srv1.pem to cacert.pem

vi cacert.pem and delete the first certificate and save the file

Copy the file splunk-srv1.pem to splunk-srv1.crt and delete the last two certificates and save the file

Handling PKCS12 formatted certs

- Sometimes you'll even get PKCS12 (.pfx) files back from the certificate authority / SSL admin
- PKCS12 files may contain both certs and keys
- To change the format:
  - `openssl pkcs12 -in $PKCS7_FILE -out splunk-srv1.pem`
  - Take the resulting .pem file, and break it up into different files for each part
    - CA Root / Intermediate certs
    - Your issued certs
    - Keys (if any)
  - You "should" be able to tell which is which by the common name and issuer
    - If not, run each through '`openssl x509 -text -noout -in <file>`'

Certificate verification vs common-name matching

- Unique, but complementary, parts of the SSL authentication scheme
- Splunk can do same CA verification without common-name matching
- Splunk CN matching does require CA verification be true
- Certificate verification is a cryptographic operation.
  - Does a cert's signature by its issuer cryptographically verify when checked using the issuer's public key?
- Common-Name matching comes next
  - Does the CN= in the certificate match the CN you are expecting?
  - Browsers do this comparison against the DNS host name in the URL
  - Splunk does this by hard coded configuration entry

errors :)

- This is from enabling sslVerifyServerCert = true and screwing up a cluster peer's cert on purpose

- 09-07-2014 00:51:55.619 -0400 ERROR SSLCommon - Certificate doesn't verify, err=19
- 09-07-2014 00:51:55.619 -0400 INFO NetUtils - SSL Connection could not be made - server authentication error
- 09-07-2014 00:51:55.619 -0400 WARN HTTPClient - SSL_ServerAuthError connecting to=104.131.13.214:8089
- 09-07-2014 00:51:55.619 -0400 WARN HTTPClient - Connect to=104.131.13.214:8089 timed out; exceeded 30sec

more errors 

- This is from (again on purpose) putting in a false CommonNameToCheck

- 09-07-2014 15:53:33.771 -0400 ERROR SSLCommon - Common name doesn't match server cert common name=splunk-d.myorg.com. Tried to match aaa.bbb.cc.
- 09-07-2014 15:53:33.771 -0400 WARN HTTPClient - SSL Connection could not be made - server authentication failed
- 09-07-2014 15:53:33.771 -0400 WARN HTTPClient - SSL_ServerAuthError connecting to=splunk-d.myorg.com:8089
- 09-07-2014 15:53:33.771 -0400 WARN HTTPClient - Connect to=splunk-d.myorg.com:8089 timed out; exceeded 30sec

CipherSuite Errors ⚠

- Ran into an error setting up SSL on indexer cluster.
- After enabling new cert on the CM, error doing a 'splunk apply cluster-bundle'
- Splunkd.log on CM shows

```
error:1408A0C1:SSL routines:SSL3_GET_CLIENT_HELLO:no shared cipher
```

- Some wiresharking later ...

```
cipherSuite = HIGH fixed it
```

FIN

splunk>