How to Deploy CFEngine in the Open Internet

Dimitrios Apostolou
jimis@cfengine.com
CFEngine AS

Bootstrapping

Trust

TLS Protocol

Selective Asset Distribution

Diagnosing problems

Bootstrapping

Trust

TLS Protocol

Selective Asset Distribution

Diagnosing problems

How do I start with CFEngine?

You “bootstrap”...

`cf-agent --bootstrap 1.2.3.4`

- Removes all `inputs/`
- Generates `inputs/failsafe.cf`
- Checks if address provided is local; if so:
  - Sets `am_policy_hub`
  - Touches `state/am_policy_hub`
  - Checks existence of `masterfiles/promises.cf`
- Writes `policy_server.dat`
- Evaluates `failsafe.cf`

failsafe.cf

- Runs `cf-key`, i.e. generates `ppkeys/localhost.{priv,pub}`
- `inputs => copy_from hub:/var/cfengine/masterfiles`
- `cf-execd`
- `cf-agent -f update.cf`

In broad terms, bootstrapping ensures that:

- Policy is copied over from the Hub
- Executor is started
- Agent **trusts** the Hub

Bootstrapping

Trust

TLS Protocol

Selective Asset Distribution

Diagnosing problems

Peer's identity: `MD5-0123456789abcdef.pub`

```
# cf-key  -p /var/cfengine/ppkeys/localhost.pub
MD5=839bfd0b494358fe67aaf9a607246c41
```

- A peer **trusts** all identities that are in `ppkeys/`
- A peer **does not trust** anybody else.

- SSH-like: No certificates, no CAs, no CRLs
- SSH-unlike: Two-way trust

Trust Establishment

```
body copy_from:
   ➔trustkey => “true”;      #failsafe.cf

body server control:
   ➔trustkeysfrom => {“0.0.0.0/0”};
```

All connecting peers are always accepted and key is stored as trusted.

Trust Established

```
body copy_from:
   ➔trustkey => “false”;        #default

body server control:
   ➔trustkeysfrom => {};
```

Any connecting peer with an unknown key is immediately rejected.

Trust revocation

- Keep your `trustkeysfrom` list empty and your ACLs closed down
- Delete respective key file

Standard bootstrapping (easiest)

(1) Ensure protected internal network

(2) Automatically bootstrap each client (tip: visually verify MD5 IDs)

(3) Empty `trustkeysfrom` and close-down the rest of the ACLs (`allowconnects` mostly)

Note: Client can then be deployed anywhere and even change address, keys are not attached to IPs.

Standard bootstrapping (easiest)

(1) Ensure protected internal network

(2) Automatically bootstrap each client (tip: visually verify MD5 IDs)

(3) Empty `trustkeysfrom` and close-down the rest of the ACLs (`allowconnects` mostly)

Note: Client can then be deployed anywhere and even change address, keys are not attached to IPs.

–- Not recommended in the Open Internet ---

Alternative ways to bootstrap

Run from the hub:

```
cf-runagent -i -H $CLIENTIP
```

Manually exchange keys:

DEMO

DEMO

- Hub up and running, properly secured (`failsafe.cf`, `update.cf`, `promises.cf`, `cf_serverd.cf`)
  - `failsafe.cf` changed and served in masterfiles
- Client: `cf-key  # generate key pair`
- Client: `echo $HUB_IP > policy_server.dat`
- Copy `failsafe.cf` to client's inputs
- Client: running `failsafe.cf` fails, must copy keys
  - `cf-key -p` gives the MD5 ID of the key
  - `scp localhost.pub $CLIENT_IP:root-MD5=aaaa.pub`
  - `scp $CLIENT_IP:localhost.pub root-MD5=bbbb.pub`
- Client: `cf-agent -f failsafe.cf  # Success!`

Bootstrapping

Trust

TLS Protocol

Selective Asset Distribution

Diagnosing problems

Why prefer TLS?

- Fully encrypted and integrity checked channel
- Frequently attacked
- Frequently updated
- Industry standard
- Slightly faster (more optimised code), higher grade ciphers
- User experience remains the same – same old trust model, no certificates, no CAs, no CRLs

How to enforce TLS in CFEngine

`body common control` in `promises.cf`, `update.cf`, `failsafe.cf`:

- `protocol_version => “latest”`
- `trustkey => ”false”`

`body server control`:

- `allowlegacyconnects => {}`

Plan is to extend TLS support

- Support new versions of the TLS spec
- More configurability (ciphers, versions etc)
- Change to TLS as default

Bootstrapping

Trust

TLS Protocol

Selective Asset Distribution

Diagnosing problems

Generic Guideline

- Do not distribute secrets in the policy
  - Can't avoid secrets' distribution? Encrypt using client's public key, decrypt using clients's private key

cf-keycrypt (community effort)
https://github.com/cfengineers-net/cf-keycrypt

There are cases that even not-so-sensitive data needs to be protected, e.g. password hashes

- `bundle server access_rules:`
  - `admit_keys`
  - `shortcut`

DEMO

- bundle server access_rules:

```
"/var/cfengine/priv/$(connection.key)/shadow"
      shortcut   => "myshadow",
      admit_keys => { "$(connection.key)" };
```

- Agent policy:

```
“/etc/shadow” => remote_cp(
  “myshadow”,
  “$(sys.policy_hub)”
);
```

- Equivalent to

```
“/etc/shadow” => remote_cp(
  “/var/cfengine/priv/MD5=.../shadow”,
  “$(sys.policy_hub)”
);
```

Bootstrapping

Trust

TLS Protocol

Selective Asset Distribution

Diagnosing problems

Diagnosing Problems

TODO: Launch verbose server listening on different port, connect from problematic server to that port! … ;-)

Questions?

Ideas?

Flames?