Monitoring Certificate Validity

Moshe Zadka – https://cobordism.com

2020

Acknowledgement of Country

San Francisco Bay Area Peninsula
Ancestral home of the Ramaytush Ohlone

Why Internal TLS?

- Defense in depth
- "Internal"

Local Certificate Authority

- "Real" CA too cumbersome
- Internal domains/IPs

Frequent rotation

Short validity (7-90 days)

Trust Management

Rotating the CA?

Every Rotation is a Chance to Fail

Frequently...

Failed Creation

- Crypto problems
- OS problems
- ...and more

Failed Uptake

- Caches
- Long running processes

Unobservable by Default

Walking of a cliff

Correlated

Similar machines, similar code

Coordinated Expiry

Catastrophic failure

Goal: Validity Time

NOT:

- Security tool
- Check correct signature
- Check valid CA

Comfort with Low-level TLS

Uncommon goals: specialized tools

TLS Auth Failures

Post-date handshake

Relevant Cert

Last one

Tool: Python PyOpenSSL

- Built-in SSL too high-level
- cryptography too low-level
- Sans-IO
- 1:1 OpenSSL API

TLS Handshake: Step

```
def handshake_loop(sock, sock_ssl):
    for i in range(100):
        try:
            sock_ssl.do_handshake()
        except WantReadError:
            with suppress(WantReadError):
                to_send = sock_ssl.bio_read(4096)
                sock.sendall(to_send)
            read_bytes = sock.recv(4096)
            if len(read_bytes) == 0: break
            sock_ssl.bio_write(read_bytes)
        except SSL.Error as err: break
        else: break
```

TLS: Context

```python
def permissive_ctx(certs):
    ret = SSL.Context(SSL.SSLv23_METHOD)
    ret.check_hostname = False
    def callback(conn, cert, *args):
        certs.append(cert)
        return True
    ret.set_verify(SSL.VERIFY_NONE, callback)
    return ret
```

TLS Handshake: Loop

```
def get_cert_from_sock(sock):
    certs = []
    ctx = permissive_ctx(certs)
    sock_ssl = Connection(ctx, None)
    handshake_loop(sock, sock_ssl)
    # Ignore errors for now
    return certs[-1].to_cryptography()
```

TLS Handshake: Networking

```
def make_socket(host, port):
    sock = socket.socket()
    sock.settimeout(1)
    sock.connect((host, port))
    return sock
```

TLS Handshake: Combining

```
def get_cert(host, port):
    sock = make_socket(host, port)
    return get_cert_from_socket(sock)
```

Calculate Validity

```
def days_left(cert):
    delta = cert.not_valid_after - now()
    return delta / timedelta(days=1)
```

Prometheus Exporter

```
def metrics(request):
    cert = get_cert(HOST, PORT)
    reg = CollectorRegistry()
    metric = Gauge(”days_left”, registry=reg)
    metric.set(days_left(cert))
    content = generate_latest(reg)
    return Response(content, CONTENT_TYPE_LATEST)
```

StatsD

```
while True:
    cert = get_cert(HOST, PORT)
    statsd.gauge(”days_left”, days_left(cert))
    time.sleep(60)
```

Alerting

Nobody likes watching dashboards

Alerting

Nobody likes watching dashboards
and these ones are watching paint dry.

Parameters

- Input: Validity: 7-90

Parameters

- Input: Validity: 7-90
- Input: Rotation period: 1-45

Parameters

- Input: Validity: 7-90
- Input: Rotation period: 1-45
- Output: Expected range

Low-urgency

Longest "office shutdown"

Conclusion

- Internal TLS is here to stay
- Short validity is currently best practice
- Fifty ways to fail your rotation
- Monitor, alert, fix

Conclusion

- Internal TLS is here to stay
- Short validity is currently best practice
- Fifty ways to fail your rotation
- Monitor, alert, fix
- ...or end up in a post-mortem meeting