Protecting Keys/Secrets in Network Automation Solutions

- Dhananjay Pavgi, Tech Mahindra Ltd
- Srinivasa Addepalli, Intel
Agenda

• Introduction
• Private Key Security
• Secret Management
• Tamper Detection
• Summary
Protecting Credentials: Private Keys, Security Certificates aren’t enough
(Trojans via Vulnerabilities and Phishing, Malicious VMM administrators)

Heartbleed attack – Vulnerability in OpenSSL, that expose private keys. How widespread is this? The most notable software using OpenSSL are the open source web servers like Apache and nginx. The combined market share of just those two out of the active sites on the Internet was over 66%.


https://wiki.xenproject.org/wiki/Virtual_Machine_Introspection - Libvmi that enable VMM Software to look into the VM memory – Can be used to extract VM secrets.

Implications: Financial loss and brand loss from web site spoofing, account misuse, money withdrawals and huge fines.


• Backdoor.Beasty
• Infostealer.Snifula
• Downloader.Parshell
• Trojan.Spyeye
• W32.Cridex
• W32.Qakbot
• Infostealer.Shiz
• Trojan.Carberp
• Trojan.Zbot
Attack Surfaces & Threats – In case of Network Automation projects

Major threats
- Credential (Private keys & Passwords) Stealing
  - Files & DB
  - Memory
  - Images with default passwords
  - Captured traffic
- Tampering
  - OS/VMM
  - Container/VM image
- Data Stealing
  - From files, Databases, Disks

Edge deployments increase attack surface significantly

Micro Service architectures – Key at every container.

Edge Clouds – Keys at every NFVI, VNF, VIM
### Adversaries

<table>
<thead>
<tr>
<th>Different type of adversaries</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unprivileged Software Adversary</td>
<td>Adversary who has user space privileges</td>
</tr>
<tr>
<td>System Software Adversary</td>
<td>Adversary who gets root privileges</td>
</tr>
<tr>
<td>Network Adversary</td>
<td>Adversary who exploits vulnerabilities in software</td>
</tr>
<tr>
<td>Software Side-Channel Adversary</td>
<td>Adversary who can get hold of information due to shared resources.</td>
</tr>
<tr>
<td>Hardware Adversary</td>
<td>Adversary who is hardware expert and gets access to hardware</td>
</tr>
</tbody>
</table>
Many services still use password based authentication – Various database servers, Publish/Subscribe brokers, Log Service, VIM and many more...

Passwords are stored in files in many services.

Passwords and Certificate private keys are sent to services via environment variables or files (mostly in clear)

There are no checks on whether the server is tampered, before sending sensitive information

Security project intention is to make it hard for adversaries to get hold of credentials (keys & Passwords), secrets and blacklist compute nodes if tampered.
Digging Deeper: Private keys Security - Need

Concern: Snapshot cloud & memory swap features puts the keys in storage in clear → Key exposure to cloud provider administrators, attackers

Need: Keys shall not be in clear in storage memory.

Concern: Keys are in clear in application memory. Exploits to application can result into Keys being stolen by attackers. 65% of vulnerabilities can be used to gather information, including keys.

Examples: Heartbleed attack → 17% of all servers in 2014 are vulnerable → It provides that majority of servers store the keys in clear.

Need: Keys shall not be in clear in application memory.
Private Key Security – Challenges in Micro Services world

Micro Service Architecture
- External Services
- Internal Services (Due to decomposition)

Many Micro Services and Scale-out
- Each Service requires Certificate & private key (for Mutual TLS)
- Usage of Internal CA (Less expensive, On-demand)

Polyglot - Micro Services implemented in different languages in a project
- Every language has their own Crypto library

Service Mesh technologies (eg. ISTIO) does not reduce attack surface
- Side car
- As many side cars as number of services
- Service meshes have their own CA.

Cloud world use technologies PKCS11 based Network HSMs (e.g CloudHSM) to protect keys and perform security cryptography. Keys are never in clear in memory or storage.
Good news - Many Crypto libraries support PKCS11 for Private Key Security

- ONAP and other open source projects are polyglot.
- PKCS11 seems to be the right interface → We selected PKCS11 to protect keys.
ONAP Security Project – Architecture blocks for CA

CA Private key – Security is very important. Should be able to use External HSM. Current plan:
- Upgrade ONAP Java CA with PKCS11
- Integrate SoftHSM
- Upgrade SoftHSM to take advantage of HW security (start with TPM plugin and use SGX later)
- Future: Update ISTIO CA with PKCS11.

TPM, SGX and TZ all have following properties:
- Ability to upload private key securely or generate keys securely.
- Never exposes private keys out of HW boundary.
- Perform RSA/ECDSA/DSA private key operations (Sign) and Decrypt in case of RSA
Each Service requires secure storage & secure crypto execution on private keys

Some highlights:
- PKCS11 enablement - Ensures that each service uses PKCS11 key label instead of private key
- Usage of local HW facilities (such as TPM and SGX).
- Usage of SoftHSMv2 as HSM with hardware support
Secret Management Service - Need

**Need**

- Avoid passwords sprawl in files.
- Avoid passing passwords via environment variables to services.
- Avoid clear passwords in storage.
- Should support not only secret protection not only at Micro Services, but also at remote edges (VIM, VNFs, NFVI) – Not only containers, but VMs, bare-metal SW etc...
Secret Server – In ONAP

Creating Secret Domain
1. Admin user creates authentication session by passing username/password OR grant token given by Auth Service.
2. SMS validates the Admin user credentials using Auth Service.
3. Admin user instructs SMS to create secret domain (Policies, CA-Cert, Set of Subject name prefix to allow vs permissions)

Service instance bring up (Assumes that service already got the certificate provisioned)
4. Service makes Secret service request via TLS. (Create Secret/Read Secret)
5. SMS validates the certificate credentials, if valid and if policy allows it, perform the operation. Also, create and return SMS-token for future operations.
6. SMS returns the results

Service uses secrets for service specific processing

- Create secrets at central place.
- Secure secrets using HW Security
- Get Secrets on demand basis.
- Zeroization secret upon usage by Service.
ONAP R2 will have Secret Management functionality.
Being done in generic fashion (can be adopted by other projects)
Master key is protected using HW security (TPM to start with and SGX in future).
Future: Keep the secret secure at the service level (Caching the keys) for performance reasons)
Tamper Detection Need

**Need**

- **Software tampering can result in stealing data and monitoring sensitive data.**

- **Discovering of any tampering (Tampering proof itself should not tampered)**

- **Tampering detection – BIOS, OS/VMM, User space libraries/services**

- **Secure VNF images at edges (IP protection as well as for tampering)**

- **Bring up VNFs on trusted compute nodes**
Tamper Detection – Marinating DB of trusted/untrusted Servers

- Integrity Service:
  - Gets the measurements and verifies using white listed values to check for any tampering.
  - Maintains the DB of trusted and untrusted servers.
  - Provides query services for others to get information on trust of servers.

- Server agents:
  - Measures the SW component integrity at boot up.
  - Measures any new installed software
  - Usage of TPM: measured values are stored in TPM

Status: We feel that there is a need for this in ONAP and planning to propose it as a project.
Summary (What we have learned so far)

• Micro Service architecture increases attack surface.
• Edge-Clouds can be attack mounting points.
• New HW Servers have built-in HW security (TCG TPM, SGX etc..)
• Combining learnings from Cloud → addressing new attack surfaces and leveraging HW technologies
  - Private key security using HSMs with security rooted in HW.
  - Secret protection rooted in HW.
  - Tamper detection using TCG TPM.

• Please join us and contribute.