S-NFV: Securing NFV states by using SGX

Ming-Wei Shih  Mohan Kumar  Taesoo Kim  Ada Gavrilovska
Georgia Institute of Technology
Network Function Virtualization (NFV)

Virtualized Network Functions (VNFs)

- NAT
- IDS
- Web Caching
- VNF

- OS
- VM

Hypervisor

Hardware

NFV Infrastructure
Stateful network functions

Virtualized Network Functions (VNFs)

- NAT: IP address
- IDS: Policy
- Web Caching: Cached Web
- VNF: States

- Hypervisor

NFV Infrastructure
“Introspection Risk for NFV
Hypervisor introspection, including administrative and process introspection, presents a risk to **confidentiality**, **integrity**, and **availability** of the NFV. Introspection can enable the ability to **view**, **inject**, and/or **modify operational state** information associate with NFV…” — ETSI GS NFV-SEC 003
S-NFV: Design Goal

• Threat Model

  • Underlying software is untrusted

• How can remote parties gain trust on VNFs?

• How to ensure the security of NFV stats?
S-NFV: Design Goal

- New NFV framework
- Integrate with Intel SGX
- Ensure the security of NFV applications’ states
- Allow remote party to verify
- Requires only application-level changes
Intel Software Guard Extensions (Intel SGX)

- Intel CPU extensions
  - Code/Data can be kept in a secure container (*enclave*)
  - Dedicated physical memory (Enclave Page Cache, EPC)
  - Different memory access semantics are enforced
  - Support remote attestation over enclave
- Supported by Intel Skylake CPUs
  - SGX-enabled version is released on October 2015
**S-NFV Overview**

Virtualized Network Functions (VNFs)

- **NAT**: IP address
- **IDS**: Policy
- **Web Caching**: Cached Web
- **VNF States**

- **OS**
- **VM**

- **Hypervisor**

- **Hardware**

**S-NFV Framework**
S-NFV Overview

- Decouple original VNF
- S-NFV Enclave: contains states and related logics
- S-NFV Host: the rest code of VNF
S-NFV Overview

• S-NFV Enclave Design

• Clear Isolation
  
  • Separating out states and related operations from original VNF

• Safe APIs

  • Provide interfaces to support host and enclave interactions without revealing states
Remote Attestation

- Leverage SGX’s remote attestation feature to attest S-NFV enclave
- Secure bootstrap
- Establish secure channel

S-NFV Framework

- VNF
  - S-NFV Enclave
  - SGX loader
  - S-NFV Host
- Service Provider
- Deployment Request Attestation
- Report
- Quoting Enclave (EPID)
Case Study: Snort

- Snort
  - Lightweight network intrusion detection system
  - States: IDS policy (TagNode data structure)
  - Configured during the bootstrap
  - Dynamically create/update and used to check packet during the runtime
Implementation

• Implement prototype on OpenSGX

• Extract TagNode and Tag Operations from Snort

• Port on SGX-supported machine (no available SDK as the time of submission)
Case Study: Snort

• Result

• Modify 5 Tag operation APIs

• 489 LoC changes to original Snort

<table>
<thead>
<tr>
<th>API</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>void InitTag(void)</td>
<td>-</td>
</tr>
<tr>
<td>void CleanupTag(void)</td>
<td>-</td>
</tr>
<tr>
<td>int CheckTagList(char*, Event*,</td>
<td>Packet* → char*</td>
</tr>
<tr>
<td>void*)</td>
<td>void** → void*</td>
</tr>
<tr>
<td>void SetTags(char*, TagData*,</td>
<td>Packet* → char*</td>
</tr>
<tr>
<td>, uint16_t)</td>
<td>OptTreeNode* → TagData*</td>
</tr>
<tr>
<td></td>
<td>RuleTreeNode* → -</td>
</tr>
<tr>
<td>void TagCacheReset(void)</td>
<td>-</td>
</tr>
</tbody>
</table>
Evaluation

• Based on Packet Performance Monitor plugin in Snort

• ~20% overhead on packet processing

• ~10% overhead on rule checking
Conclusion

• We take a first step toward protecting network function’s states by proposing new NFV framework

• Use Snort as a case study

• decoupling an original NFV application to fit S-NFV model

• preliminary evaluation on real hardware