Dependable Virtual Appliances

Nuno Santos
In collaboration w/ Nuno Lopes
October 5, 2014
Virtual Appliance: Cloud’s *de facto* executable

- Virtual Appliance: pre-configured virtual machine image
  - Provides, e.g., LAMP web server, mail server, database server…

- Today, 1000s of VAs available online, e.g., through Amazon EC2
But beware of risks!

- Researchers looked at 100s of public VAs [Bugiel11,Huh13]

- Lack defenses. How can we mitigate these risks?

  - Data from deleted files
  - Passwords in files and logs
  - Private IP addresses
  - Key material
  - Malware
  - Misconfigured SW
  - Unpatched SW
  - Unlicensed SW

Risks for VA creators!

Risks for VA users!
Our Vision: Dependable Virtual Appliances

- We call them *depliances*
  - Creators / users can verify config properties before shipment / usage
Key Techniques for Building DEpliances

Model Checking
To verify properties of VA model

Trusted Computing
To build and attest VAs

Creator's endpoint

VA Template

Model Checker
Creator's Required Properties

Trusted Builder

Cloud

User's endpoint

VA Template

Remote Attestation

User's Required Properties

Model Checker

VA

VM
Example: Joomla Depliance

- What is Joomla?
  - Content management system
  - Written in PHP
  - For publishing web content

- How to set it up?
  - Configure the following SW
    - MySQL
    - PHP
    - Apache HTTP Server
    - Linux
    - Virtual Machine
Step 1. VA Template Specification

- Specify modules and inter-dependencies
  - E.g., Linux, MySQL...

- Specify config attributes
  - E.g., “Root Pass”

- Specify files
  - E.g., /usr/mysql/my.cnf
Step 2. Property Specification and Verification

Examples of properties to verify

- **Efficiency:**
  
  “check number of concurrent threads for Apache and PHP based on memory and CPU cores”

- **Confidentiality:**
  
  “check passwords of Apache, MySQL, and PHP admin not default nor private identities”

- **Integrity:**
  
  “check versions of SW and missed patches”

- Describe modules’ behavior as state machines
  - Config and file attributes as inputs

- Specify properties as logic conditions
  - Over config and file attributes

- Model checker to verify properties
  - E.g., PRISM, SPIN
Step 3. VA Image Generation

- Through dedicated programs: trusted builders
- Input: verified VA template
- For each module:
  - Resolve dependencies
  - Run trusted builder to install and configure files of module
- Output: VA image
Step 4. VA Instantiation and Attestation

- Remote attestation protocol yields original VA template

User

\[\text{sign}\{\text{nonce, PCRs, template}, AIK_{\text{priv}}\}\]

Host w/ TPM chip

Certificates

\[\text{sign}\{\text{PCRs, SW}_{HH}, CA_{\text{key}}\}\]

\[\text{sign}\{\text{AIK}_{\text{pub}}, \text{TPM}_{\text{provider}}, CA_{\text{key}}\}\]

Certificate Authority

VM Instance of Joomla Deppliance

Hypervisor
Usage Scenarios

- Condition: VA image must be built by trusted party

Case I
VA built by creator
Cloud only stores VA

- Fits today’s model
- Must trust the creator

Case II
VA built by cloud
Creator shares VA template

- No trust needed in creator
- Requires compute time in cloud

Case III
VA built by user
Creator shares VA template

- No trust needed in creator nor cloud
- Requires compute time locally
Open Challenges and Future Work

- Thousands of SW config attributes is overwhelming
  - Idea: decouple DSL abstractions from SW config attributes

- Different properties require different verification approaches
  - Idea: study best encoding logic and verification tools

- Enable (partial) offload of VA generation to untrusted party
  - Idea: generate trustworthy log of untrusted party operations
Conclusions

- Virtual appliances popular, but creators and users incur risks
  - Misconfigurations, security threats, privacy breaches, etc.

- We propose depliance model to build dependable VAs
  - Creators / users verify properties of VA before publication / usage

- Depliance model enabled by two techniques:
  - Model checking and trusted computing

- Many open challenges to be addressed in the future
  - Find right level of DSL abstractions, efficient way of verifying properties
Thanks!
Questions?

http://www.gsd.inesc-id.pt/~nsantos/