Leveraging Virtual Machine Introspection for Hot-Hardening of Arbitrary Cloud-User Applications

Sebastian Biedermann, Stefan Katzenbeisser, Jakub Szefer

6th USENIX Workshop on Hot Topics in Cloud Computing
Philadelphia, 17th – 18th June 2014

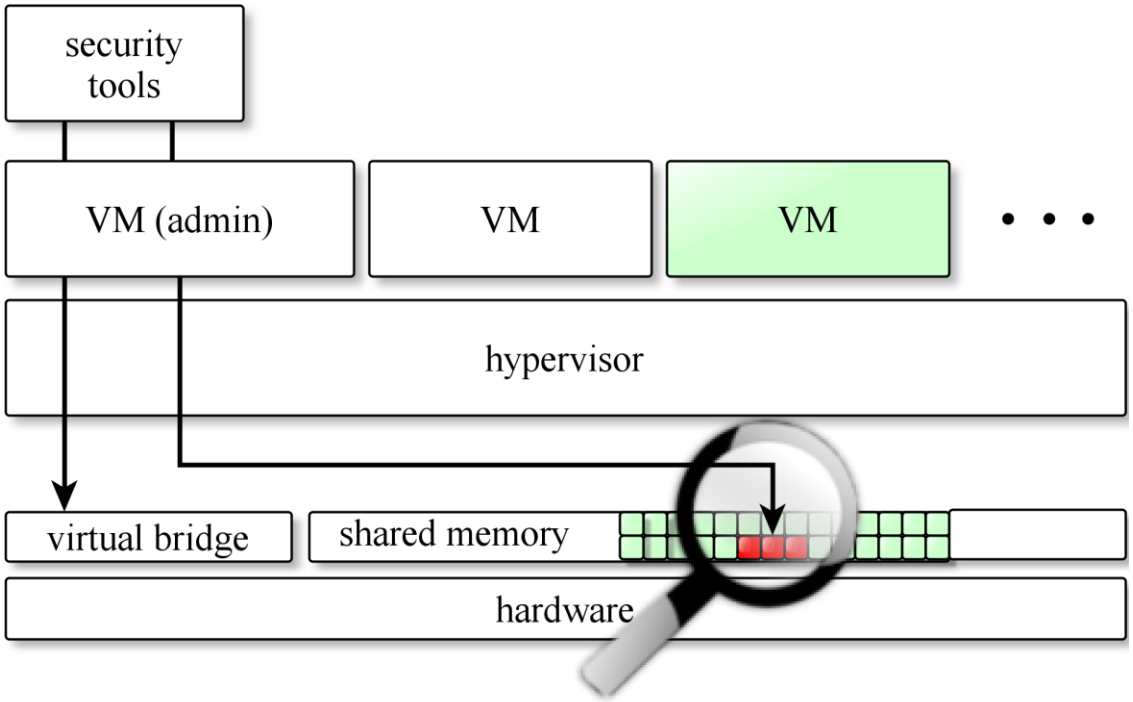
Security Engineering Group (SECENG), Technische Universität Darmstadt, Germany

Computer Architecture and Security Laboratory (CAS Lab),Yale University, USA



Introduction: Virtual Machine Introspection

Virtual Machine Introspection (VMI) is the technique of locating and accessing specific memory regions of a running VM (user) from another running VM (admin) which is co-located on the same hardware.



Yale
TECHNISCHE UNIVERSITÄT DARMSTADT

Related Work: Hot-Patching & Run-time Program Analysis

Hot-Patching:

- Upgrading a running program by modifying its code while it executes
- Hot-patching reduces downtime in systems
- Several papers in recent years (e.g. Payer et al. or Ramaswamy et al.)

Run-time Program Analysis:

- Semi-automated techniques which can diagnose configuration errors
- Help finding the point of failure or can infer configuration requirements
- Often, they analyze byte code (Java)

Hot-Hardening: An Overview

Definition of Hot-Hardening:

> *"Continuous and transparent improvement of security-related configuration settings of running applications."*

We propose an agent:

- Transparent and scalable cloud service ("Security-as-a-Service")
- No interruptions and no user interaction required, works autonomous
- Uses different techniques: VMI, live forensics on storage, live cloning, vBridge

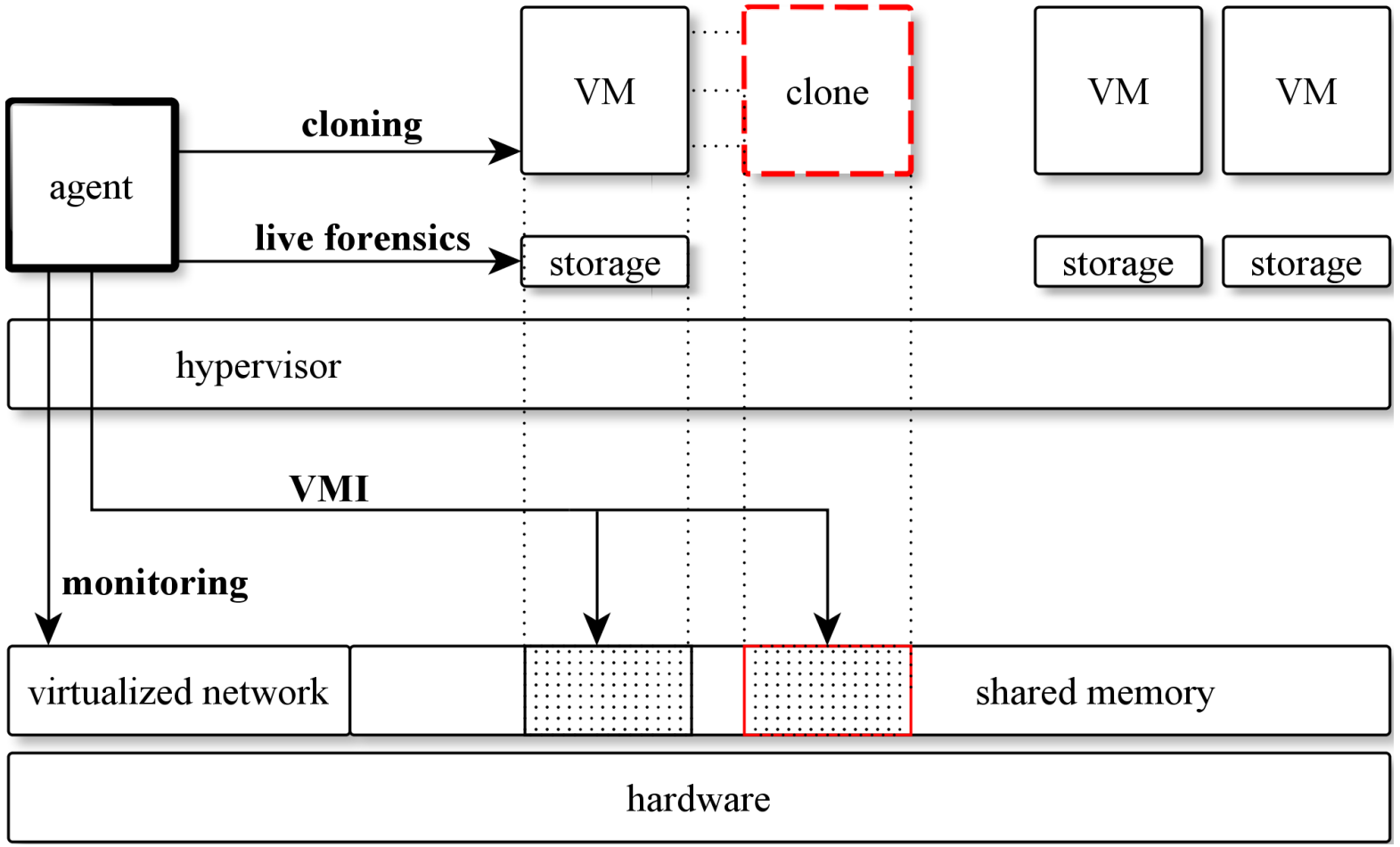
Hot-Hardening: An Overview

Configuration settings of applications are almost always stored in memory and they are frequently referenced (e.g. a struct in the stack or the heap).

Hot-Hardening procedure performed by the agent:

1. Identify currently deployed settings (live forensics on the raw storage)
2. Locate target data structure containing the settings in memory (VMI)
3. Create a clone of the VM (for testing of new settings)
4. Replace settings inside the clone by directly writing to the memory (VMI)
    - Full replacement: 1) In memory 2) Config file in page cache 3) Config file on storage
5. Test effect of the updated settings on the clone (trigger application)
6. Finally deploy better settings on original VM (VMI)

Hot-Hardening: An Overview



Yale
TECHNISCHE UNIVERSITÄT DARMSTADT

Example: Apache2 Hot-Hardening

→ Apache2's configuration file is located in /etc/apache2/apache2.conf.

→ Settings are stored in a struct data type called server_rec defined in include/httpd.h.

```
struct server_rec {
        ...
        /** Timeout, before we give up */
        apr_interval_time_t timeout;

        /** Interval we will wait for another request */
        apr_interval_time_t keep_alive_timeout;

        /** Maximum requests per connection */
        int keep_alive_max;
        ...
};
```


Yale
TECHNISCHE UNIVERSITÄT DARMSTADT

Example: Apache2 Hot-Hardening

- Struct members are stored in the order they are declared (C99 standard)
- Ensuring correct alignment with padding and considering Little-Endianness in memory

Example settings:

- Allowed time between new connections and first request: 300 s = 300x10^6 µs = **11e1a300**
- Allowed time for maintaining connection for next request: 30 s = 30 x 10^6 µs = **1c9c380**
- Allowed length of HTTP request lines in bytes: 8190 bytes = **1ffe**
- Allowed number of requests per connection: 100 = **64**

→ Search pattern **= 00a3e111**(0)*?**80c3c901** ... **fe1f**(0)*?**64**

→ Can be located and replaced in anonymous mappings of Apache2

Hot-Hardening of Arbitrary Applications

→ Techniques need to be generic because application and settings are unknown

Identification of currently deployed settings (find a potential configuration file)

- Investigate the application's memory for strings that appear to be file names
- Locate files on storage and search value unions

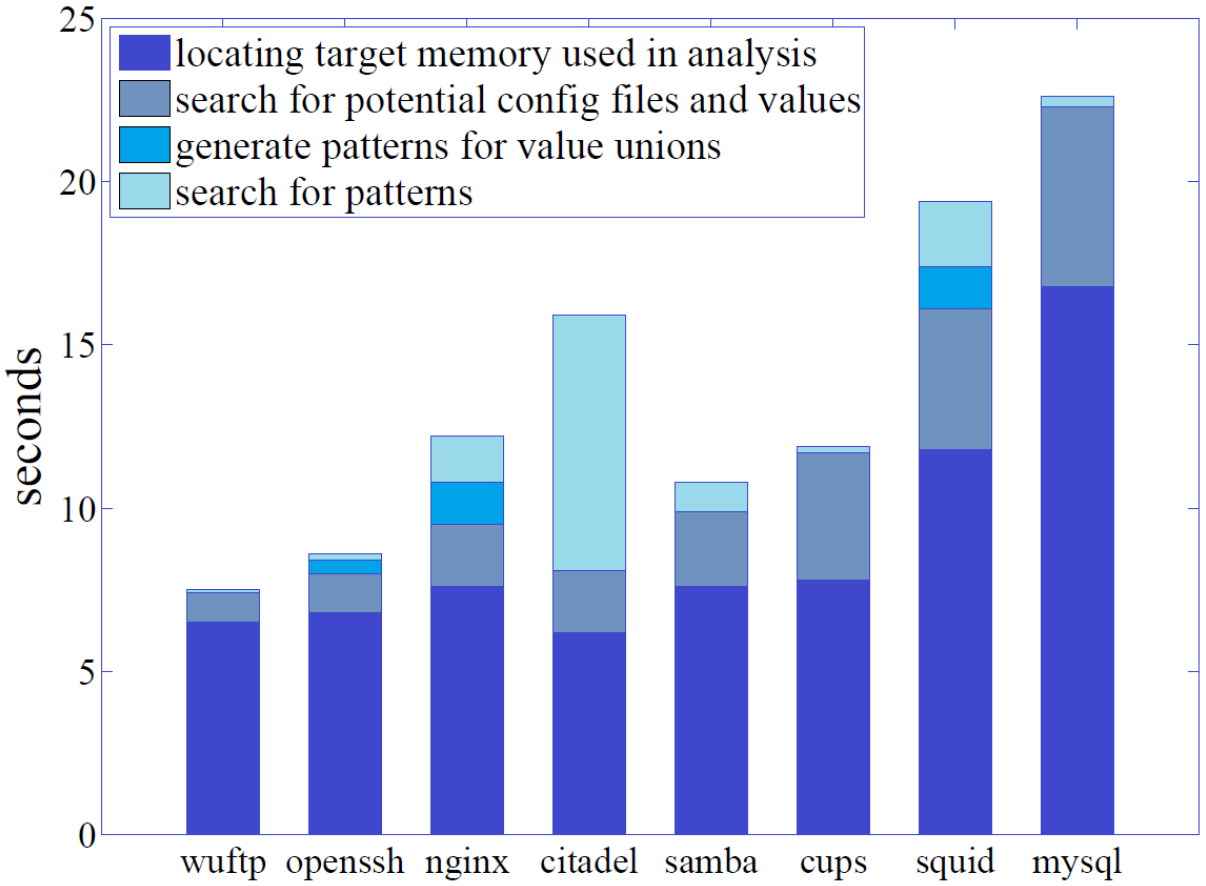
Pattern generation and localization in memory (unknown order of settings)

- Create different patterns for all orders: $\sum_{i=0}^{n}(n!/(n-i)!)$
- Scan the application's memory for the patterns

Yale
TECHNISCHE UNIVERSITÄT DARMSTADT

Evaluation

Timings required to analyze different example applications

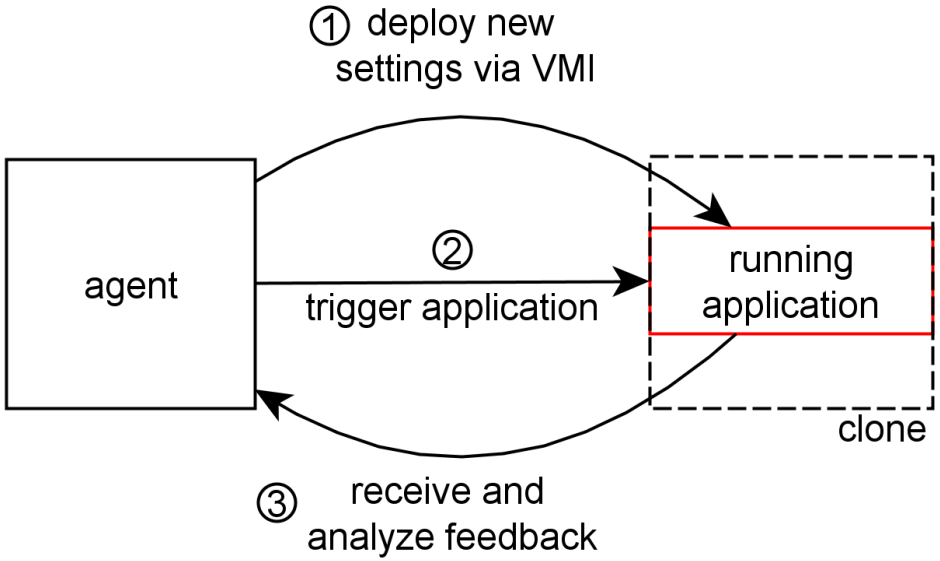


Yale
TECHNISCHE UNIVERSITÄT DARMSTADT

Hot-Hardening of Arbitrary Applications

Changing unknown settings:

- Stepwise change single settings and execute previously defined tests
- Assumptions about security characteristics need to be made
- Meaningful and universal tests need to be evaluated



Yale
TECHNISCHE UNIVERSITÄT DARMSTADT

Evaluation

Feedback: Simple test on the network layer:

- Perform multiple connections successively and in parallel as well as logins
- Count TCP RST and FIN flags sent by the application
- Assumption: The more resets, the more restrictive is the application

Results highlight:

- Could improve a setting of three of eight example applications
- E.g. could identify setting for limit of maximum allowed connections in parallel

Summary

- Introduced Hot-Hardening as transparent cloud security service
- Proposed an agent which can autonomously change settings
- Presented setting discovery strategy for unknown applications
- Designed setting improvement strategy for unknown applications

Ongoing Work:

- Use an optimization algorithm and more tests (e.g. for system calls)
- Hot-hardening when some information about application is known
- Computing custom-made settings for a target specific setup
- Adaptation of the settings of child processes independently

Thank you!

Questions?

Leveraging Virtual Machine Introspection
for Hot-Hardening of Arbitrary Cloud-User Applications

Sebastian Biedermann, Stefan Katzenbeisser, Jakub Szefer