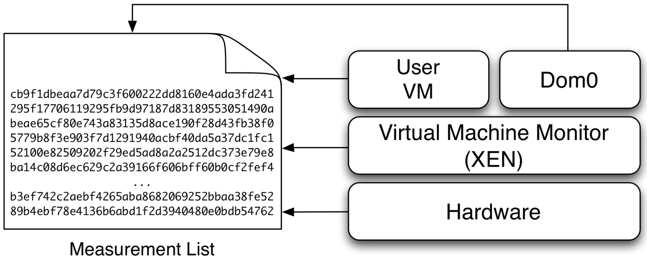
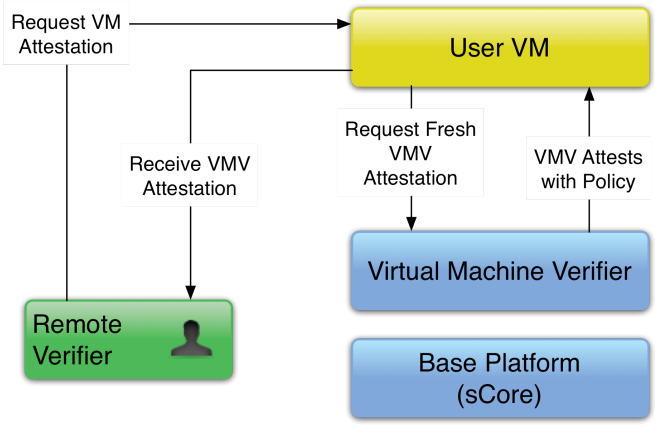
Verifying Virtual Machine Integrity by Proxy

Joshua Schiffman, Thomas Moyer, Hayawardh Vijayakumar,
Trent Jaeger and Patrick McDaniel

Integrity measurement enables remote parties to assess whether a system meets a set of security goals. However, all existing approaches place the burden of verifying often large and semantically diverse attestations on the verifier. Adding virtualization has complicated matters by increasing the amount of code that must be verified.


cb9f1dbeaa7d79c3f600222dd8160e4ada3fd241
295f17706119295fb9d97187d83189553051490a
beae65cf80e743a83135d8ace190f28d43fb38f0
5779b8f3e903f7d1291940acbf40da5a37dc1fc1
52100e82509202f29ed5ad8a2a2512dc373e79e8
ba14c08d6ec629c2a39166f606bff60b0cf2fef4
...
b3ef742c2aebf4265aba8682069252bbaa38fe52
89b4ebf78e4136b6abd1f2d3940480e0bdb54762
User VM
Dom0
Virtual Machine Monitor (XEN)
Hardware

Measurement List



Virtual Machine Verifier

We propose a small and easy to verify Virtual Machine Verifier (VMV), which functions as a verifier by proxy for remote parties. The VMV is a static VM that uses a distribution specific integrity policy to verify VM attestations locally.

The VMV runs as a privileged domain (DomP) to host the VM it monitors. When a remote party wishes to vet a VM, the VM returns an attestation of the VMV vouching for it. Thus, the remote party only needs to verify that a simple platform is enforcing a policy they trust.

Flexibility

VMVs are flexible in how they verify VM integrity. They can use any VM verification technique including:

- VM introspection
- Virtual hardware-based techniques
- Service commitments

Since the VMV runs in its own VM, malicious VMVs are isolated from the rest of the system.

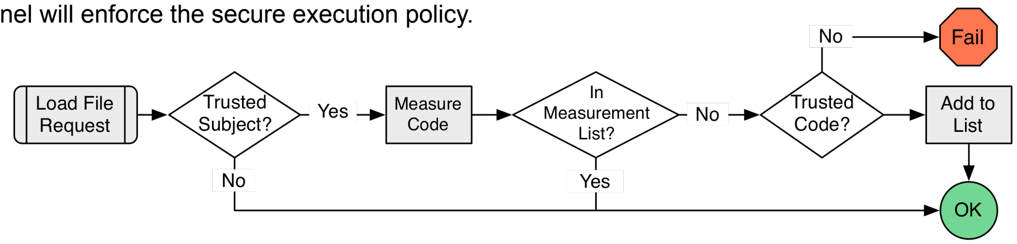
Defining Integrity Polices

Users often place their trust in entire software distributions. When a vendor adds or changes programs, users often expect others to use that same version. For our experiment, we created a custom Debian repository based off Ubuntu 8.04. Our distribution specifies its VMV and policy as a downloadable disk image and policy package.

Integrity Enforcement Mechanism

As an example method of verifying VM integrity, we modified SELinux and PRIMA to perform secure code execution. Given a policy database of trusted code hashes, the VM's kernel denies execution of any code running with a trusted subject label that is not found in the database. The VMV ensures the VM maintains load time integrity by checking that the VM's kernel will enforce the secure execution policy.

Using information flow analysis, we identified 23 types that form the trusted computing base of a standard Ubuntu distribution. The policy file for our custom repository contains 34,239 program hashes. This is about 668KB in uncompressed form.



Sponsored by NSF (CNS-0627551) and IBM Research