Poster title: Preserving the integrity of enterprise platforms via an Assured eXecution Environment (AxE)

Authors: Anurag Sharma (ansharma@us.ibm.com), Steve Welch (swelch@almaden.ibm.com)
IBM Almaden Research Center

Demo: A working prototype of the AxE runtime implemented for the Windows operating system

Problem statement and motivation:
Protecting enterprise systems from malware is a difficult task that requires constant vigilance, which creates significant overhead with mixed results. Once deployed, the continued integrity of an enterprise computing platform is hard to assure, since malware can enter the system via a multitude of ingress points. Heuristic-based technologies for malware defense require a speedy response and continuous attention by the systems administrator. Non-heuristic-based defense technologies do exist—such as Windows SAFER software restriction policies, code-signing, and Trusted Computing as specified by TCG and Microsoft's Palladium architecture. However, each have their respective shortcomings: SAFER does not control several execution entry points (such as device-drivers), a significant portion of the code that runs on enterprise platforms is unsigned, and Trusted Computing does not address the existing install base of PC's which do not have trusted hardware support. Therefore, even though many technologies are currently used in concert to address the malware problem, the challenge to maintain the integrity of enterprise computing platforms still exists.

Solution approach:
AxE aims to preserve the integrity of the enterprise computing platforms by:

1) Ensuring that only binaries "approved" by the systems administrator can execute on these machines.
2) Preventing any changes to the system configuration state from occurring.

It achieves this by providing OS kernel extensions (built using documented mechanisms) that monitor and prevent changes to the persistent system state, and also regulate the execution of all the binaries in the system (including device-drivers)—this approach ensures that even if the malware threat evades the existing defense mechanisms, it will still be prevented from running and harming the system. Our solution ensures protection regardless of how the "unapproved" software arrives on the system, e.g. via network or removable media. The architecture can control the execution of scripts in a manner similar to binaries, but in our current implementation scripts are prevented from running by blocking the binaries that comprise the various script runtimes.

Furthermore, our solution does not depend on a publicly available signature verification infrastructure—i.e. participation from 3rd party software vendors (OEMs) is not required for AxE to be effective. Our approach essentially gives each enterprise its own "approved code" deployment and verification infrastructure. All binaries approved to run within an enterprise platform are tagged with appropriate metadata (analogous to code-signing)—which can later be used to verify the integrity of the code when it is loaded for execution.

AxE also consists of deployment tools used by the systems administrator to "approve" the binaries within system images as well as the augmentations to them, such as applications and update packages. The AxE runtime itself is included within the "approved" system image. The integrity of the system image during updates is maintained by using the "AxE managed update" method—where a privileged process is allowed by the AxE runtime to make changes to the entire machine. These updates can include changes to the base OS and applications.

Early observations from the development and use of AxE:
AxE can increase the performance of signature-based defense technologies: "approved" binaries cannot change and therefore do not require repeated scanning for virus-signatures. Since technologies such as "no-execute" page-protection are being embedded in hardware, AxE does not have to implement its own in-memory buffer overflow defense techniques to ensure the runtime integrity of a process. As a side-effect, AxE also provides valuable intelligence on threats that have penetrated perimeter and desktop defenses (e.g. firewalls, virus-scanners), and are attempting to inflict damage upon the enterprise.

Ongoing and Future work:
We are currently investigating and prototyping in the following areas to increase security, while maintaining usability of the enterprise computing platform:

- Reducing the exposure of secrets on the AxE client runtime: Instead of running update programs and new application installers in the context of a privileged process on each desktop, we can capture the changes that an installation program will make to the desktop's file-system state, and to the structured state within files, and apply them as patches. This update method will allow the signing of the application-installer generated binaries on the deployment server, and eliminate the need to temporarily expose the encryption facilities and private keys (used in the signing operation) on each desktop system.
- Extension of AxE protection to handle Script runtimes at a per-script granularity (instead of simply preventing all script runtimes from executing).
- Extension of system configuration protection: Currently, as a part of the AxE security model, we make a large portion of the system state read-only (such as system configuration and binaries). This protection can be extended to make the entire system read-only, with virtualized writes, while ensuring that only the user data is persisted. This known-good state can then be reloaded either across logon sessions or reboots. Note: this needs to be done in a manner that is secure (i.e. leveraging user-mode/kernel-mode privilege separation), and uses documented methods (i.e. no binary patching and system-call interception).
- Lastly, audit information captured from systems within the enterprise can be used to construct threat profiles for intrusion detection and prevention systems, resulting in further fortification of perimeter malware defenses.

The above extensions to AxE pose some unique systems implementation challenges and research opportunities, and we hope to find innovative and sustainable methods to design and implement these technologies so they can eventually be used in production environments.