LKIM: The Linux Kernel Integrity Measurer

J. Aa P a a Ka N. McG

T[illegible] (LKIM) [illegible] LKIM [illegible] LKIM [illegible] APL [illegible] (NSA) [illegible] LKIM [illegible]

INTRODUCTION

cuting software is behaving consistentl with its static definition. Although d namic integrit measurement cannot guarantee that software is trustworth in the sense of not being exploitable, it is able to establish that an assurance gained b static anal sis is maintained b the executing software.

The Linux Kernel Integrit Measurer (LKIM) is an implementation of a d namic measurement technique targeting the Linux operating s stem kernel. Unlike most other s stems for malware detection, d namic integrit measurement s stems (IMSs) such as LKIM do not rel on a database of known malware signatures. This means that LKIM is able to detect previousl unknown ero-da malware. Although LKIM was originall developed to verif the integrit of the Linux kernel, researchers in the As mmetric Operations Department of APL have reconfigured LKIM to target the state of a piece of software (referred to as a *target*). This evidence is presented to the DM in a process called *attestation* that supports the trustworthy evaluation of the target's state. The DM is responsible for evaluating advantageous. The exact frequency with which to run LKIM is still an open research question. Because LKIM s results represent only a moment in time, a long time period between measurements may allow an adversary to break into a system, accomplish his mission, and restore the kernel s integrity without causing a failed measurement. In some sense, any window is too long because some adversary missions, such as stealing cryptographic keys, may be accomplished in microseconds. A recommended practice is to perform a fresh LKIM measurement as part of an access control decision such as network access control. This scheme allows the access control policy to make its decision on the basis of fresh evidence, without unduly burdening the target.

Integrity measurement data may be as complex as a legacy systems and does make it somewhat more difficult for rootkits to hide.

It is impossible to develop a measurement system with no impact on the target. Any measurement engine running on the same hardware as the target will have to compete for the finite computational resources available, such as processor time or memory. We have made efforts to minimi e the impact LKIM poses on the target both by optimi ing LKIM s code to reduce its use of these resources and by leveraging architectural features such as VM snapshotting to avoid activities such as pausing the target for long periods of time. Beyond these performance impacts, a measurement system may impact the development or deployment of updated targets. LKIM requires a precise description of the data structures used by the software it is measuring. This means that legitimate updates to the target may cause LKIM to become confused and generate false alarms. We partially address this problem by separating the target-dependent aspects of LKIM into a configuration file that should be deployed in sync with updates to the target software. This solution imposes some management cost to deploying LKIM; we are working in pilot deployments to better understand how high this cost is and how it can be best addressed (see the ***LKIM Transition*** section for more detail).

HOW LKIM WORKS

Although LKIM provides a generic measurement capability, the majority of work on applying LKIM has focused on the Linux kernel itself. This section provides an overview of LKIM s measurement algorithm using the configuration developed for the Linux kernel as an example. Efforts to retarget LKIM to measure other software, including other operating system kernels and application-level software, have followed the same process with minor changes to account for differences in file formats and software architecture.

LKIM divides the task of verifying the Linux kernel s integrity into three distinct phases:

- [illegible]: the identi cation of valid program states
- [illegible]: the collection of evidence
- [illegible]: the evaluation of the collected evidence against the baseline

Figure 1 indicates how these three phases work together to determine the integrity of the system. The baselining phase combines an expert understanding of the Linux kernel s behavior with information found in the kernel s executable file. The measurement phase inspects the state of a running instance of the kernel to summari e those aspects of its state relevant to the integrity decision; this summary constitutes a measurement of the running kernel s state. The appraisal phase

Appraisal

The appraisal phase consumes these data and compares them with data computed from the on-disk representation of the kernel to determine whether the observed state is consistent with the original file. This determination relies on an expert understanding of how the kernel operates. Ultimatel, LKIM provides a result indicating either that no modifications were detected or exactl which data in the kernel have been modified in an unexpected wa.

LKIM s appraisal phase is specified as a series of logical predicates that are evaluated over both the baseline and measurement graphs. These predicates can refer to the graph node representing data at a particular address in memor, all nodes representing data of a particular tpe, or the relationships among multiple graph nodes. In the example given in Fig. 2, an example constraint is that all read function nodes that descend from the given file sstem node must point to the correct executable code for reading a file from that FileSstem.

During LKIM apomp4ilph no emplM(()

of challenges to LKIM, and we hope to ease the integration of LKIM as we gain operational experience.