CC3: An Identity Attested Linux Security Supervisor Architecture

Richard Engen MSFS, Johannes Grosen MS,
Scott Stofferahn, Greg Wettstein R.Ph., Ph.D.
IDfusion, LLC
{rick,john,scott,gw}@idfusion.org

Abstract

Ubiquitous global networking and the economic incentives of commodity hardware and operating systems have conspired to produce a crisis of unprecedented status in information security. Of particular concern is security for systems containing data or controlling infrastructure where no *ex-post-facto* redress is available for information disclosure. Recent compromises suggest classic defensive systems based on intrusion protection and detection technologies are failing by leaving systems compromised for months before detection. Integrity measurement architectures (IMA), in combination with dynamic root of trust, offer the means to implement systems which support behavior compromise detection, if the challenges of managing and modeling system behavior can be addressed. In this paper we introduce the concept of an iso-identity integrity measurement architecture for implementing the detection of compromised platform behavior and present a Linux based security supervisor based on this design. We conclude this system represents the limit of detectability for integrity measurement architectures.

1 Introduction

Global networking has produced a system of unparalleled value to modern society. Extracting value from this system of information exchange is based on the ability to implement intelligent endpoint nodes on the network consisting of systems ranging in size from supercomputers through wearable systems. The premise of the *Internet-Of-Things*, popularized as IOT, suggests this trend will continue.

Unfortunately this proliferation of intelligent endpoints has created an environment unparalleled in the history of information security and risk management. Each endpoint represents a potential failure point with respect to the desired system behavior being subverted to the goals of an aggressor seeking to capture privileged information or disrupt the intended functionality of the endpoint. The global network which synergizes the utility of these endpoints also provides the framework for launching attacks against the endpoints from anywhere in the world, with the added complication of little or no possibility for attributing the origin of the attack.

Since the inception of global networking, the strategy has been to protect intelligent endpoints by sequestering them from access by the network at large through the use of firewall technology. Simple firewalls have given way to stateful inspection systems which seek to recognize and optionally interdict attempts to subvert the functionality of the protected systems. The effect of these systems has been to produce architectures which are effectively soft targets once the perimeter protection systems have been breached.

In order to maximize the effect of a compromised system, the focus by malicious actors has been on the development of *Advanced Persistent Threat* technologies which seek to introduce long term behavioral modifications to the endpoint targets. This provides a mechanism which persists the ability to exfiltrate information from the compromised systems long past the initial compromise.

This strategy is particularly effective in the firewall model since it allows other 'soft' targets in the interior of the protection domain to be attacked and infiltrated without interdiction by the perimeter defense systems. The response has been to employ additional protection systems to monitor internal network traffic in an attempt to interpret whether or not illicit behavior is being

demonstrated by any internal network endpoints.

If aggressors avoid detection by perimeter systems it is likely that internal network monitoring will also fail. Any type of traffic monitoring also faces challenges associated with steganographic methods which shroud illicit traffic in ever increasing quantities of legitimate traffic. An industry movement toward the use of strong encryption will lead to increasingly random data streams which will be used to ambiguate illicit network traffic.

Major system compromises in 2014-2015 in the federal government, entertainment, retail and healthcare industries have demonstrated the ability of attackers to persist information exfiltration attacks for long periods of time without detection. In these attacks aggressors have demonstrated the ability to export hundreds of gigabytes of data without being detected by internal or perimeter defense systems.

Maintaining the security of network endpoints has classically involved continually applying updates to security vulnerabilities in operating systems and application platforms. This strategy, is by definition, unreliable in the face of *zero-day* exploits which leverage previously undiscovered software vulnerabilities to implement both the initial compromise and subsequent persistence of attack systems

In spite of ever more sophisticated attack systems, the primary threat to effective security is the economics of information technology. Vendors seek to appease markets that demand platforms which implement the value proposition of ubiquitous networking but which do not reward attention to the security implications of such systems. Addressing the modern information security challenge requires attention to the economics of security which requires cost and complexity minimization on network endpoints.

It has been doctrine in the security industry that security and complexity are mutually incompatible. The recent attention to *containerization* strategies is an attempt to reduce the complexity and attack surface of service providing endpoints. While such systems provide isolation they do not provide a system for determining whether or not the behavior of the encapsulated system is consistent with the design intent of the system.

In this work we address these security challenges through the development of a *security supervisor,* hereafter referred to as CC3, which implements verification of the behavior of service-providing network endpoints. The focus of devices based on this technology are the large number of single intent systems, where the most cost effective strategy for deployment is to use a standard Linux or Windows operating system as the application platform. Our work addresses the thousands of network endpoints which handle privileged information and/or which may be used to harbor malicious software and were deployed with an emphasis on reducing the cost of the endpoint.

Examples of such applications include but are not limited to:

- Firewalls/routers (L3-L4 appliances)
- Point of sale systems
- Digital signage
- Information exchange/transport
- System Control / Data Acquisition
- Cloud microservices

The primary responsibilities of the security supervisor are to implement and maintain the system in a pre-defined behavioral state and to support the execution of applications needed to achieve the desired service function of the endpoint.

An important contribution of this work is the notion of basing the ability of the endpoint to implement its functionality on a mathematical measurement of the desired functional behavior of the platform. The CC3 supervisor implements the POSSUM protocol which is a minimalistic system for implementing autonomous mutual remote attestation between two verifying platforms. Completing a POSSUM protocol exchange

requires that two platforms prove their current behavioral measurement is consistent with what has been designated as acceptable for the platform.

Since the supervisor is designed to deploy into environments where there may be hundreds or thousands of endpoints the behavioral status measurement is linked to the intrinsic identity of the device. This scheme provides a framework for guaranteeing no two devices have the same behavioral measurement, even if they possess identical operating systems and/or application frameworks. Violation of the integrity of any one device thus does not lead to information which would be of value in spoofing or attacking the integrity status of other devices in the service delivery framework.

2 Intended Security Environments

The CC3 supervisor was originally developed to address the security needs of devices which were managing information which had no *ex-post-facto* redress for information disclosure. These environments are defined as situations which lack a reversibility path for information disclosure or system compromise. Examples of such environments include:

- Privileged healthcare information
- Personally sensitive information
- Life safety implications

These are distinct from disclosures, such as credit card information, which are reversible through financial remuneration. It is arguable that early experiences with environments which were reversible in nature may have minimized the potential security risks of poorly protected endpoints. Since financial institutions acted to restore the financial impact of compromised systems on affected consumers, there was no perceived cost to the system compromise and subsequent disclosure.

Since the disclosure of information cannot be 'unlearned' a breach of privileged information is catastrophic within the impact domain of the information. For example, the compromise of systems in the Office of Personnel Management led to a situation where the utility of personnel in security sensitive positions may be forever compromised by the disclosure of personally sensitive information.

In the case of healthcare, knowledge regarding potentially sensitive information such as genetic predisposition to disease, may threaten an individuals future employment or cost of insurance.

It has now become apparent that automotive and perhaps flight control systems may be inadequately protected at their functional endpoints. Compromise of intended behavioral characteristics of such systems may result in loss of life, clearly an irreversible process.

It is anticipated that insurance or re-insurance forces will intervene and serve as a governing influence in the security market. One of the design intents of the CC3 supervisor was to provide a system with quantifiable guarantees to evaluating parties on the integrity of insured systems.

3 Identity Model

The design of the CC3 supervisor is based on an *identity-attestation* architecture. This model assumes each device will have a unique identity assigned to it by a provisioning organization. The attestable measurement value of the platform is a manifestation of the intrinsic identity assigned to the device.

The identity model is based on the generation of an N-bit number used to represent an identity. The identity is generated by applying a cryptographic hash function over two components; a range selector and a credential. [1]

This is formalized by the following description:

$$I_M = H_M(R_M || H_M(C))$$

Where:

I_M = Identity
R_M = Range value of size M
C = Credential
|| = Concantenation
H_M = Hash function digest size M

The R_M value serves to select the possible range of functional values which can be achieved by any given credential mapping. Absent knowledge of the range selector, the actual credential used to formulate the identity is protected from disclosure by the infeasibility of determining the $2*M$ sized function pre-image.

An organization provisioning devices has a top level identity defined by some credential value such as a tax identification number. Once the organizational identity is generated, it serves as the R_M value for subordinate identities within the context of the organization.

The subordinate identity space is logically defined into three separate groupings:

- Users
- Services
- Devices

For the purposes of this work, attention is restricted to the device identity which is derived by applying the basic identity expression in the following form:

$$D_M = H_M(O_M || H_M(D_C))$$

Where:

D_M = Device identity
O_M = Organization identity
D_C = Device credential
|| = Concantenation
H_M = Hash function digest size M

In our current work SHA256 is used for H_M which yields the device identity as a 256 bit injective mapping into a range selected by the organizational identity.

The CC3 supervisor uses a functional expression of this identity. This is a three component expression which implements the following three elemental functions of an identity:

- Identity assertion
- Identity implementation
- Identity authentication

The identity assertion is a 512 bit representation of the identity which allows the device to indicate it is in possession of an implementation of the identity which is represented as a 2048 bit number. The identity authentication element is a 256 bit number used to authenticate the assertion and implementation of the identity.

An ASCII representation of the functional expression of the identity was developed for tooling and expression needs. The following is an example of the format used, in reduced form for publication demands:

```
-----BEGIN IDENTITY TOKEN-----
-----BEGIN ASSERTION-----
feedbeaf
-----END ASSERTION-----
-----BEGIN IMPLEMENTATION-----
feadbeef
-----END IMPLEMENTATION-----
-----BEGIN AUTHENTICATION-----
deadbeef
-----END AUTHENTICATION-----
-----END IDENTITY TOKEN-----
```

An ASN.1 representation of the identity is used internally in the supervisor.

4 Iso-Identity Integrity Measurement

The initial implementation of the CC3 supervisor used the native Integrity Measurement Architecture (IMA) as implemented in the Linux kernel by the IBM Trusted Computing Group[2] and extended by community collaboration.

Field experience with CC3 deployments led to a structural modification of the existing integrity

measurement model which is being introduced in this work as the Iso-Identity Integrity Measurement Architecture (I3MA). This alternate formulation derived easily and naturally from the existing IMA implementation and uses the previously described identity model to express a deterministic measurement value for platform behavior.

The theory for I3MA is delivered in the form of the following three premises:

Premise 1:

The system behavior identity of an actor process is expressed by the functional projection of the identity factors of the process over the identity factors of an acted upon subject identity.

Premise 2:

The functional projections of Premise 1 represent a mutually exclusive and collectively exhaustive set of contours which represent an $A*S$ set of values. These define the desired behaviors of the platform, where A represents the total number of unique actors and S represents the total number of unique subjects.

Premise 3:

Neglecting inter-contour and extra-contour time dependencies, a single deterministic measurement of the platform behavior is given by the extension hash sum of an arbitrary ordering of the contour points from premise 2, each projected into a range selected by the device identity.

The conceptual notion for the iso-identity contour model is heavily influenced by the primary author's experience in the field of quantum molecular orbital theory. Each actor identity can be thought of as precessing through an 'orbit' defined by its interaction with a field of subject identities. Each of the mutually exclusive iso-identity contours represents the allowed behaviors of an actor identity.

The sum of all the contours or *behavioral orbits* represents a bounded measurement of the gross platform behavior. This is analogous to the notion of the energy of a molecular system being modeled as a linear combination of all the atomic orbitals used to describe the atoms in the molecule.

Premise 3 reflects the notion of the gross measurement of system behavior as the sum of the behaviors of the individual actor identities. The important model simplification of time invariance is inherent in this premise. Intra-contour time independence implies the notion of neglecting the order in which actor identities interact with subject identities. Inter-contour time independence implies the notion of neglecting any dependency in the ordering of the actions of different actor identities.

Accepting the simplification of time independence implies that such a system is $(A*S)-1$ degenerate. As an example, a system with two actor and two subject identities yields four behavior measurements, any combination of which reflects the deterministic system behavior value. This is once again analogous to molecular orbital theory where multiple molecular wave functions can yield the same molecular energy value.

The notion of degenerate representations of a single system behavior measurement has implications with respect to both system management and the functional integrity of the gross system measurement.

The acceptance of time independence implies the security supervisor can load the complete set of acceptable iso-identity contours at system initialization time. A system reference quote taken at this time reflects the desired behavioral measurement of the system over its functional lifetime. Any additional actor/subject contour projections result in a perturbation of the system behavior measurement which is a detectable event.

From a security perspective, the simplification of

time invariance implies an acceptance of some loss in fidelity of the gross system measurement representing the desired system behavior. Examples would be a situation where the integrity of the system is based on either the order in which an actor acts on specific subjects or the order in which separate actors operate.

Increasing the precision of the gross system measurement requires reducing the degeneracy level of the measurement model. This requires ordering of the contour points so they are representative of the actor/subject trajectory path in the system being modeled. This can be accomplished either in-situ when a measured system environment is designed, or by capturing the contour points at some point in the system initialization process.

The factors used to compose the identities in a measured system must exhibit closure over the system being measured. An example of this issue is in designing attested systems which interact with other attested systems. A cyclic dependency is inherent in attempting to include the measurement state of the counter-party system as a subject identity factor secondary to that system depending, in turn, on the measurement status of the system being constructed. A strategy for addressing this issue and the challenge of writable files is discussed in the I3MA implementation section.

The final issue to be addressed is the formulation of the fundamental actor and subject identities discussed in premise 1. The actor identity is given in general form by the following:

$$A_M = H_M(F_1 \ldots F_N)$$

Where:

A_M = Actor identity
F = Actor identity dimensions
H_M = Hash function digest size M

The dimensions used to create the actor identity are selected to reflect properties which are important in regulating the actions of the actor over its subject field. Dimensionality suitable for this role include the following:

- Discretionary access control values
- Mandatory access control labels
- Security capabilities

The subject identity is given in general form by the following:

$$S_M = H_M(F_1 \ldots F_N)$$

Where:

S_M = Subject identity
F = Subject identity dimensions
H_M = Hash function digest size M

The dimensions used to create the subject identity are selected to reflect the characteristic properties of the identity. Dimensionality suitable for this role include the following:

- Filename
- Cryptographic sum of file contents
- File inode number
- Filesystem UUID
- Discretionary access control values
- Mandatory access control labels

The iso-identity contour projection point for a given actor identity operating on a subject identity is given by the following:

$$C_M = H_M(A_M \| S_M)$$

Where:

C_M = Contour point
A_M = Actor identity
S_M = Subject identity
$\|$ = Concantenation
H_M = Hash function digest size M

It is important to note the C_M identity is the generic projection of the actor/subject identity interaction and can be calculated in-situ by the system designer given a knowledge of the identity factors used to represent the actor and subject. As

premise 3 notes, this point is projected into a device specific identity through the following function:

$$P_M = H_M(D_M \| C_M)$$

Where:

D_M = Device identity
C_M = Contour point
$\|$ = Concantenation
H_M = Hash function digest size M

This final identity projection allows the measurement contours to be prepared as part of the system image without security concerns over their content. The final platform measurement is protected by managing the security of the device identity.

5 I3MA Implementation Details

As was noted previously, the implementation of iso-identity contour modeling was a natural extension to the standard Linux IMA architecture. The integration of this functionality was enhanced through recent community work to implement customizable templates for defining event measurement values.

From a conceptual perspective, the primary difference between IMA and I3MA is the notion of basing a measurement event on the interaction of subject/actor identities rather then a change in file content. This is demonstrated by the finding in our work that I3MA benefited from the addition of an additional verb to the existing IMA policy actions. The **MAP** directive, and its negation **DONT_MAP,** were added to the measure, appraise and audit actions and their negations.

This change was in part driven by the fact the existing IMA code is based on a one measurement per inode model. In I3MA the number of measurements per inode is based on the number of independent actor identities which operate on the inode.

To minimize impact on the existing code structure and to support desired enhanced functionality an identity contour cache was implemented which is interrogated for a cache hit when a **MAP** action is requested. If the contour point is not found in the cache, a call to the ima_store_measurement() function is made to generate a new measurement value which is used to extend the Platform Configuration Register (PCR) based hardware measurement value.

Since the **MAP** action is functionally a super-set of the **MEASURE** action, a new integrity measurement policy named ***iso-identity*** was created which replaces the measure action with the map action. In addition, the UID and EUID based action selectors were replaced with action selection by capability masking.

5.1 Capability Based Policy Triggers

The notion of using capability masking as a condition for triggering a map or measure action was fostered by a regression found in the original IMA implementation. The uid directive triggered an action based on a check of the real uid of the process involved in the measurement. This is most commonly triggered by the FILE_CHECK policy directive:

```
measure func=FILE_CHECK mask=MAY_READ uid=0
```

This resulted in a condition where a setuid binary run by a normal user would not trigger a file measurement but would when run by the root user. This behavior was obviously not optimal if the reason for triggering a file measurement is based on the assumption that a process with administrative privileges should be monitored when accessing files. Since the Discretionary Access Control (DAC) check is based on the effective identity of the process, a setuid binary run as a normal user would circumvent DAC security checks but its action in doing so would not trigger a measurement action.

Based on our findings, the euid directive was added to the IMA policy which triggers a policy action based on the effective identity of the process. In addition, having the CAP_SETUID bit set in the process capability mask triggers the

identity check against all three of the user identities (real, effective and saved). Since the CAP_SETUID capability allows the process to assume, with respect to DAC checks, any of the three identities, this modification triggers measurement for any root capable process.

Experience with this anomaly suggested a superior method for triggering measurements would be based on the capability mask of the process rather then any specific user identity. This is consistent with the fact that permission checks throughout the Linux kernel are based on the capability mask of a process rather then its user identity.

A new 'capability=' policy directive was added which supports either the keyword 'any', which specifies a full capability mask, or a reduced capability mask specified in hexadecimal.

An action is triggered if the following action mask contains any non-zero bits:

$$A_{\text{MASK}} = P_{\text{MASK}} \wedge (Eff_{\text{MASK}} \vee Per_{\text{MASK}})$$

Where:

A_{MASK} = Action mask
P_{MASK} = Policy mask
Eff_{MASK} = Effective capabilities
Per_{MASK} = Permitted capabilities

The iso-identity integrity policy implements a mapping action based on the following FILE_MMAP and FILE_CHECK function checks:

```
map func=FILE_MMAP mask=MAY_EXEC capability=any

map func=FILE_CHECK mask=^MAY_READ capability=any
```

The authors believe capability masking in integrity policies will become more important as the concept of *ambient capabilities* is implemented in the Linux kernel security architecture. Ambient capabilities are an attempt to address well understood functional issues with the current capabilities model. The goal of ambient capabilities is to allow increased extension of granular permissions to binaries.

The use of capabilities has been consistently plagued with subtle security behaviors. Capability based policy triggers enable the detection of possibly undesired and subtle privilege violations irregardless of the DAC based identities assigned to an actor process.

5.2 Actor/Subject Identities

In the current I3MA implementation the SHA256 hash function is used as the identity projection function. All identities are thus 32 bytes in length. The SHA256 function was chosen in order to minimize concerns by clients over the SHA1 algorithm being 'insecure' even though the first pre-image resistance strength of the function is currently not in question.

As previously noted, the IMA template infrastructure was leveraged to provide support for creation of the actor and subject identities as components of the event measurement. The 'actor' and 'subject' field names were added to the list of supported measurement fields. The initialization functions assigned to these fields generate the A_M and S_M identity projections.

To support the extension of the C_M identity into the P_M identity an additional member, named genhash, was added to the ima_template_desc structure. This structure member contains a pointer to a function which is responsible for generating the digest value over the template fields which will be used to extend the PCR measurement register.

In order to support the POSSUM protocol, which will be discussed later, the I3MA implementation maintains a kernel based 256 bit 'soft' measurement register. The measurement generation function computes the host specific contour point identity P_M and extends the soft measurement register with this value. For compatibility with 1.x based Trusted Platform Module (TPM) hardware, a SHA1 digest value is generated over the P_M identity and returned to the caller to be used to extend the IMA PCR

measurement register.

The 'actor' field initialization function computes the A_M identity based on the generation of a SHA256 digest over a packed structure which includes the following DAC identities; uid, euid, suid, gid, egid, sgid, fsuid and fsgid. In addition, the capabilities of the process are captured by including a bitmask representing the intersection of the effective and permitted capabilities mask, ie $(\mathrm{Eff}_{MASK} \vee \mathrm{Per}_{MASK})$

The 'subject' field initialization function computes the S_M identity by computing the identity digest value over a packed structure which includes the SHA256 checksum of the file contents, uid, gid and inode number. The filename is included, largely for diagnostic value, via the standard filename field descriptor.

5.3 Securityfs Interface

Management interfaces to I3MA are surfaced via the securityfs pseudo-fileystem. The management pseudo-files are grouped in the following pseudo-directory:

/sys/kernel/security/ima/iso-identity

Within this directory the following files implement the management interfaces to the system:

- contours (r)
- forensics (r)
- host_identity (w)
- map (w)
- measurement (r)
- pseudonym (w)
- sealed (w)

The 'host_identity' file is write-only and used to set the device identity (D_M) value which will serve as the range value for projecting the individual iso-identity contour points. The device identity is set by writing a 64 character hexadecimal value to the file.

The 'contours' file outputs the contents of the identity cache as a series of C_M values in hexadecimal format, one value per line. The contents of this file represent the history of subject/identity interactions which lead to the current behavioral state of the platform.

The 'map' file is write-only and is used to populate the I3MA identity cache. The file accepts a series of C_M values in hexadecimal format, one value per line. In addition to being added to the identity cache, the identity projections are extended with the D_M identity to yield P_M identities which are used to update the current soft measurement state of the platform as well as the PCR based hardware measurement value.

The 'measurement' file is read-only and displays the current soft measurement status of the platform. This is a 256 bit measurement value which represents the extension hash sum of the P_M identities. The purpose is to provide a high performance interface to the current platform behavioral measurement. Since this measurement is not anchored in hardware it must be considered advisory unless it is validated with a hardware based reference quote.

The 'sealed' file is write-only and is used to seal the behavioral measurement state of the platform. Writing a 1 to this file removes the host_identity, contours, map and pseudonym files, thus disabling any further modifications to the platform behavior model. Internally it enables a counter which limits the number of subsequent actor/identity interactions which will be recorded.

Once the iso-identity measurement status of a platform is sealed, any subsequent $A_M S_M$ interactions are recorded and made available in the read-only 'forensics' pseudo-file. This file provides tagged entries which document the policy event and the identity factors which were involved in generating the event. The following is an example of a forensic event in a sealed system:

function: 1
process: su
pathname: /etc/group
uid: 0
euid: 0
suid: 0
gid: 50
egid: 50
sgid: 50
fsuid: 0
fsgid: 50
capabilities: 0x1fffffffff

This event was generated by a FILE_CHECK policy action by an actor process running the su binary accessing the /etc/group subject file. The process was running with a full set of capabilities but with a mixed set of DAC identities, secondary to the fact it is a setuid binary which has used its CAP_SETUID capability to modify its DAC access profile.

As will be discussed in the section on system architecture the CC3 supervisor runs from a RAM based block device. In the event of a platform compromise the forensics pseudo-file is designed to provide a history of the events involved in such a compromise.

The 'pseudonym' file is write-only and is used to declare identity factor pseudonyms for subject identities. The notion of pseudonyms was developed for I3MA to address files which have variable contents or may be involved in Time Of Measurement/Time Of Use (TOMTOU) or open-writer violations as part of their acceptable use pattern.

A pseudonym is declared for a subject by writing the pathname of the file to the 'pseudonym' file. This causes an integrity inode cache entry to be allocated for the file. The flags entry for the inode has the IMA_PSEUDONYM and IMA_COLLECTED flags set. The ima_hash digest value for the structure is set to a value which is serially extended from the D_M identity.

Once defined, any S_M subject identities generated from this inode use the derived digest value rather then a checksum over the file contents. This yields a constant identity value for files whose contents are allowed to be variable by the system designer. In addition any TOMTOU or open-writer violations involving inodes with the IMA_PSEUDONYM set are disregarded.

To provide support for I3MA pseudonyms the security_inode_unlink function was modified to include a call to the integrity_inode_free function. This insures the integrity inode cache entry for a subject declared as a pseudonym is destroyed if the file is removed.

Subject pseudonyms are designed to be declared by the security supervisor as part of the initialization of a measured platform. Once the iso-identity state of the platform is sealed, the pseudonym digest value cannot be restored which would result in an off-contour projection for any interactions involving this subject.

The securityfs interfaces are designed to simplify the design and implementation of a measured application platform. The iso-identity contour points which implement the platform behavior measurement can either be captured at a known point in the system initialization process by reading the contents of the 'contours' file or generated by tooling developed to support the CC3 supervisor. A security supervisor or an equivalent entity defines the desired platform measurement by writing the contours into the 'map' pseudo-file followed by sealing of the platform measurement state.

In a dynamic root of trust environment a hardware quote taken at this point provides a reference to the identity state of the platform anchored in the hardware state of the device. Any subsequent actor/subject interactions which have not been defined or whose identity factors are modified will generate a perturbation in the platform measurement which can be verified by subsequent reference quotes. The nature of the violation can be determined through the contents of the 'forensics' pseudo-file.

6 Security Supervisor Architecture

The CC3 supervisor architecture is an extremely minimal platform built on top of I3MA which implements a measured application platform for hosting applications or execution environments with documented system behavior. CC3 supports the following execution environments:

- Native binaries
- Virtual machines
- Container environments

The platform consists of the security supervisor which implements a replacement for the classic init process and is referred to as sinit, plus a minimal set of support binaries. The current implementation, which includes hardware 2-factor support for management authentication, is approximately 13 megabytes in size.

CC3 is a native Trusted eXecution Technology (TXT) environment which extends the Dynamic Root of Trust Measurement (DRTM) into the iso-identity state of the platform. In a virtual launch environment, CC3 runs in dom0 under the Xen hypervisor and retains physical control of the TPM.

The CC3 platform is architected to run entirely from RAM based block devices. A block device driver named HugePageDisk (HPD) was developed which implements a dynamically sizable block device using the two megabyte extended page size infrastructure in the kernel [3]. As compared to ramfs based implementations this device driver implements true block device semantics with bounded memory consumption behavior.

Three separate block based filesystems are used to support a CC3 instance:

- Root (/)
- System configuration (/etc/system)
- Platform configuration (/etc/platform)

The contents of the /etc/system filesystem is created at system provisioning time and define the operational function the system will perform. The /etc/platform filesystem is designed to hold site specific configuration information and is field configurable via 2-factor authentication technology.

An encrypted filesystem image format was developed for CC3 which implements an AES256-CBC encrypted system image with HMAC-SHA256 integrity protection. The format includes anti-forensic provisions for the encrypted images.

The filesystem image loader checks for the presence of files named contours and pseudonyms in the /boot directory of the loaded filesystem. If found, the loader writes these files into the iso-identity securityfs configuration files of the same name.

The filesystem images used by a CC3 implementation are stored in the /boot directory of the boot device. Each image file has a companion file with a .seal suffix which contains the encryption key and initialization vector for the encrypted image. The image sealing file is symmetrically encrypted with a key which is PCR sealed to the measurement state of the platform at the time it is to be loaded.

6.1 Supervisor Launch Sequence.

The outline for a system launch sequence is as follows:

- Kernel
- Security bootloader
- Security supervisor
- Counter-party validation

The trusted launch sequence is initiated from an initramfs filesystem compiled into the kernel image. The integrity of the launch platform is protected since the kernel image is covered under the trusted boot policy loaded into TPM NVram during the system provisioning process.

After kernel launch the system is prepared for execution of sinit by the security bootloader,

sboot. The loader is responsible for initializing the system to a point where the tcsd TPM management daemon will function.

Once the trusted platform resources are initialized, the security bootloader reads an ASN1 encoded functional device identity from TPM NVram which was PCR read sealed to the bootloader measurement state of the platform. A reduced form of the implementation of the identity is written to the host_identity pseudofile to set the device identity. The identity is then loaded into a time delimited keyring for passage to the security supervisor

With the device identity loaded, the bootloader configures an appropriately sized HPD block device and unseals the keying material for the filesystem image. The loader decrypts the filesystem image and loads it into the provisioned block device. When the load is completed the bootloader quiesces the system and switches the root namespace to the block device. The bootloader then executes the security supervisor as its replacement.

The security supervisor then re-starts the TPM support infrastructure. The device identity manager is started which transfers the device identity into process memory for subsequent use. The identity manager provides support for the POSSUM protocol.

The supervisor then loads and configures itself from the system configuration filesystem. It completes the system initialization based on information from the configuration filesystem and then enters platform management mode.

The platform management mode loads the platform configuration filesystem. Support is implemented in this mode for detection of a 2-factor USB authentication token to request a configuration dialogue for the system. The contents of the dialogue are persisted, if necessary, by creation of a new platform configuration filesystem image.

Once this final configuration is completed the security supervisor enters platform management mode based on the system and platform configuration information which has been accrued through the launch process. Since the system images which were loaded contain contour and pseudonym definitions, the system behavior measurement state has also been defined. The security supervisor seals the iso-identity state measurement and begins executing the platform role definition

In this mode the supervisor is responsible for maintaining the service providing status state of the device. Before carrying out application and usage dependent actions the supervisor conducts an identity attestation exchange with an appraising counter-party to verify its behavioral conformance status.

6.2 POSSUM

CC3 based supervisor systems operate in either one-to-one or one-to-many collaboration environments. Implicit in the collaboration model is the ability to carry out device authentication and mutual platform attestation.

The 'POSSUM' protocol was designed to provide a simple and easily verifiable protocol for authenticating a device and verifying its platform behavior measurement. Initially deployed field devices used IPsec as their collaboration transport layer and the POSSUM protocol was designed to replace the IPsec RACOON authentication and key negotiation system.

The POSSUM protocol was designed to provide support for implementing an Elliptic Curve Diffie-Hellman (ECDH) key exchange between two devices, predicated on the mutual identities and platform measurement status of the devices. The ECDH exchange is based on Curve25519 previously described by Bernstein and uses the implementation found in later versions of OpenSSH [4].

The protocol is based on a packet oriented send/acknowledgement model. All packets are encrypted with AES256-CBC and the resultant payload is integrity protected with an HMAC-

SHA256 checksum. The payload and verifier are sent atomically as a single packet.

6.3 POSSUM Authentication

The authentication of the POSSUM protocol exchange is implemented through One Time Epoch Differential Key Scheduling (OTEDKS). The OTEDKS key scheduling algorithm is based on a reduced form of the functional device identity described earlier in this paper.

The reduced form of the device identity implementation and the authentication element of the identity are treated as arrays of eight separate 32-bit values. The requested authentication time is subtracted from each 32 bit value to yield the epoch differential vector for the identity element. The authentication differential vector is exclusively or'ed with the identity implementation vector and vice-versa to give two starting points for key scheduling.

The total number of key iteration rounds is produced by treating each 32 bit value of the identity implementation starting vector as a standard epoch date. The day of the month represented by each epoch date is summed and added to a floor value of 250 to determine the number of key iteration rounds.

The starting round value from the identity implementation is hashed with HMAC-SHA256 with the key supplied by the value derived from the identity authentication vector. The output of the hash serves as the input to the next key scheduling round. The output is also exclusively or'ed with the current HMAC key and then hashed with SHA256 to serve as the HMAC key value for the next round.

The initialization vector for CBC encryption is taken as the midpoint value in the key scheduling rounds.

As noted in the platform architecture section, one of the daemons maintained by sinit is the identity manager. An application which desires generation of an OTEDKS key passes a buffer to be encrypted along with the desired authentication time to the daemon which generates the keying material and then encrypts the buffer. This model is designed to thwart the identity manager from being used as a key oracle in the case of a compromise of the platform.

6.4 Identity Verification Files

The POSSUM protocol is a departure from the Open AttesTation (OAT) system which implements database driven exchanges of platform integrity status[5]. The POSSUM approach was chosen to provide a very simplistic framework which does not require any infrastructure outside the confines of two participating parties to be involved in the validation of platform behavior state.

In order to simplify the logistics of this approach, the notion of identity verification (IVY) files was developed. These files have a .ivy extension and are ASN1 encoded repositories of the following information:

- Reduced device identity
- Platform public key
- Soft platform measurement
- Hardware reference quote

The device provisioning process generates an IVY file which can be used in a POSSUM protocol exchange to verify the operational state of a device.

Optimal security suggests IVY files for a device be privileged to the devices which are using them to attest platform behavior. However, the compromise of an IVY file does not allow the construction of a device which can spoof the behavior of the platform represented by the IVY file, since the unreduced implementation of the identity is required to implement and thus fully validate the behavior of the platform. The only copy of that identity is entrained within the PCR sealed NVram state of the device.

6.5 POSSUM Exchange

A device requesting authentication with a counter-party generates an authentication packet with the

following elements:

- Replay nonce
- Quote nonce
- ECDH public key

The replay nonce is used to support the replay avoidance guarantee of the protocol and the quote nonce is used by the receiving party for the generation of the platform quote. This packet is encrypted with the OTEDKS keying material for the requested authentication time.

A challenge packet is then constructed with the following elements:

- Authentication time
- Identity challenge
- Authentication challenge

The identity challenge is derived from the identity assertion component of the functional identity.

The resulting packet is sent to the desired communication party, integrity protected by an HMAC-SHA256 checksum. The checksum key is derived from the OTEDKS epoch key and the value in the soft platform measurement state pseudo-file.

The receiving party uses the identity challenge to locate the IVY file of its communication counter-party. The soft platform measurement state and authentication time are used to carry out integrity verification and decryption of the authentication challenge. If this process is successful, an identity challenge packet for the current device is constructed to send to the initiating party. The ECDH key returned in this challenge is formed from the shared key provided by the initiating party.

The initiating party upon receipt of the identity challenge from its counter-party validates the authentication challenge. If validation is successful, the ECDH derived shared key is used to encrypt a platform measurement packet containing a hardware reference quote based on the nonce supplied in the counter-party challenge.

Upon receipt of the platform measurement packet, the counter-party decrypts the reference quote with the ECDH session setup key. If the decryption and validation of the reference quote is successful a platform measurement packet is generated and returned to the initiator.

Upon validation of the counter-party platform behavior, the initiator signals completion of the circuit setup by sending a circuit rekeying request. The counter-party replies with an ECDH key response and the shared key from this acknowledgement phase is used as the session key for any further communication.

The POSSUM protocol implementation requires both parties to the attested circuit validate their hardware reference quotes before a final shared key is established. Since the reference quote is derived from the iso-identity platform measurement a functional guarantee is produced that both participating parties have verified platform behavior.

6.6 Possum Pipe

The realities of field deployments, which include NAT, firewalls and configuration of devices in DMZ network zones, demonstrated the need to have a non-IPsec based transport layer. The POSSUM protocol was thus extended to be a general purpose, message oriented communications framework. A full coverage of this protocol will not be considered in this document, only a brief functional summary.

The POSSUM protocol was designed to implement a system authenticating the identity of a device and its operational behavior status. Successful execution of a POSSUM exchange leaves both devices with a mutually agreed upon shared secret. In IPsec based implementations, this secret is used to derive the authentication key and initialization vectors for an ESP based tunnel.

PossumPipe is an extension of this protocol designed to support subsequent device

communications when IPsec is not a viable option. The ECDH based shared secret is used as an ephemeral key to authenticate and integrity verify subsequent packet exchanges.

This protocol implements two unique characteristics with respect to the exchange of secured packets.

Each packet encryption key and initialization vector are derived from the session shared secret by hash extension summing of that secret by the unencrypted contents of the packets which have been sent to date. Each party to the circuit maintains both a sent and received register with these values.

A compromise of the session key is thus not useful in decrypting any individual packet unless the entire unencrypted history of the communication stream up to that packet is available. Each transmitted frame includes a nonce of random size which is immediately discarded by both the sender and receiver. This measure thwarts the deduction of subsequent packet keys even if the complete contents of a communication stream is known.

The other innovation of the protocol is to perturb the HMAC-SHA256 key used to generate the integrity checksum of a packet with the current soft measurement state of the platform. Compromise of a platform is thus immediately represented as an inability of a counter-party to verify the integrity of communications from an affected device.

Implementation of this latter feature was the rationale for surfacing the 'soft' measurement state of a platform in the securityfs filesystem. High packet transmission rates require the ability to obtain platform measurements at a rate which would otherwise be constrained by the hardware performance of the TPM.

7 Discussion

The CC3 security supervisor provides a minimalistic implementation of a platform built for the purpose of attesting the behavioral integrity of a service providing network endpoint. It offers a departure from classic network protection strategies in that it requires a device be able to validate its functional behavior as a prerequisite for its ability to provide services.

Platform integrity measurement is implemented through iso-identity contour modeling. This system models the functional behavior of a system as the sum of the actor/subject identity interactions which implement the desired platform behavior.

Effective use of integrity measurement architectures as the basis for validating system behavior requires an understanding of the type of behavioral deviations which can be reliably detected versus illicit behavior which is undetectable. Understanding this differentiation is critical to effectively using integrity metrics to design trusted systems

We introduce the following two concepts to classify system compromise behavior:

- Intra-dimensional compromise
- Extra-dimensional compromise

The transition from an extra-dimensional to an intra-dimensional compromise represents the mathematical limit of detection for integrity measurement systems.

More precisely, this interface serves as the transition point where detection systems must evolve from deterministic to probabilistic methods. This is again consistent with the modeling of physical systems which have demonstrated the need to transition from deterministic newtonian to probabilistic quantum mechanical models as particle size decreases.

An intra-dimensional compromise is a modification in system behavior which does not result in a contour projection different from those designated by the system designer and/or captured at system measurement time. Conceptually, this can be thought of as a modification in behavior which does not cause an actor identity to depart

from its acceptable 'behavioral orbit'.

Due to the avalanche effect of the cryptographic hashes used to formulate the participating identities, an intra-dimensional compromise implies no variation could have existed in any of the dimensional factors used in the composition of an actor or subject identity. An example of an intra-dimensional compromise would be a situation where Bob finds Mary's password on a sticky-note under Mary's keyboard. Bob uses Mary's credentials to login and access information which Mary is authorized to view but Bob is not. The system is operating within its iso-integrity contour mappings but with its desired information disclosure behavior compromised.

An extra-dimensional compromise is a situation where an actor identity goes '*off-contour*' secondary to a change in any of the dimensional factors used in the composition of the A_M or S_M identities. This results in a C_M identity not included in the gross system behavior measurement and is thus a detectable event. These types of compromises are thus highly amenable to detection by deterministic behavioral modeling.

Intra-dimensional compromises are ill suited for detection by integrity measurement architectures since, by definition, there is no violation of system integrity, function or behavior. Detecting this type of compromise requires the application of behavioral prediction techniques which seek to determine if the application or system is being used in a manner inconsistent with a normal pattern of usage.

While intra-dimensional compromises are challenging to detect, the use of integrity measurement systems is not without utility. The Office of Personnel Management breach of 2015 appears to be an example of an intra-dimensional compromise where user login and passwords were stolen. Reports suggest administrative privileges were then used to persist the malware which, in a suitably modeled system, would be a detectable event.

From a mathematical perspective, an intra-dimensional system compromise represents the limit of detection of an integrity monitoring system. In the absence of a transition to completely probabilistic behavior, extending integrity measurement detection limits requires increasing the dimensionality of the elements used to model system behavior. By modeling behavior based on actor/subject identity interactions, each of which are composed of multiple dimensional elements, I3MA yields a higher level of sensitivity then systems, such as classic IMA, which models system integrity primarily on file contents.

The modeling of actor/subject interactions is similar in concept to mandatory access control models based on type enforcement systems which base access control decisions on subject and object label intersections. The C_M contour points are, in essence, the identities of the intersection points in a type enforcement system. Expressed in this manner, the iso-identity contour of an actor is the set of allowed access conditions for a subject over an object label field in a type enforcement system.

Current development work with I3MA involves adding the MAC labels of an actor and subject to the set of identity factors used in the composition of the A_M and S_M identities. This will integrate type enforcement behavior into the overall platform behavioral measurement.

Iso-identity integrity modeling thus offers synergies to type enforcement systems by linking the defined enforcement model to a quantifiable measurement of the platforms conformance to that model.

Regardless of the identity factors used, iso-identity contour modeling has at its basis the notion of a white list of permitted behaviors. Specifying an iso-identity contour map is the equivalent of defining the set of behaviors which the system will be allowed to demonstrate and then deriving a single measurement value which

reflects the sum of these desired behaviors. Any behaviors outside the bounds of the model perturbs this single measurement of platform behavioral state.

An important emerging issue is the notion of *containerization* which involves providing private implementations of system resources to a process and its subordinates. Technically the concept of containerization is manifested in I3MA as a reduction in subject dimensionality.

While containerization is important with respect to isolation and subject dimensionality reduction, it does not address the problem of unintended actor/subject interactions or behavior. For example, a WEB server running in a container can still experience a security violation which could be used to subvert the behavior of the container and/or the application it is supporting.

Given this, it is anticipated that behavioral modeling will remain important as the use of container technology accelerates. Design work is underway by the authors to provide namespace support for the identity cache in the current I3MA implementation. Unsharing the identity cache allows the specification of an alternate set of contour points to define the desired behavior of the actor/subject interactions in a container.

One of the open issues is addressing how to reflect the behavior of the container in the overall platform measurement status. By design, in a perfectly isolated container, there should be no expression at the platform level, of anomalous container behavior.

The current I3MA model uses TPM PCR extension to express the platform behavior in the form of a hardware measurement. An alternate method will be required for containers since it would be undesirable to have a perturbation in container behavior change the overall platform measurement status.

One possibility would be to use a resettable PCR register as the target for the measurement extensions. Unfortunately this is a limited hardware resource and thus a constraining commodity in the face of environments hosting hundreds of containers.

The concept of a virtual TPM is well understood in full virtualization models. Extending this architecture to containers will require an assessment of complexity and the notion of a 'supervisor' process for each container which is responsible for implementing the virtual TPM measurement state of the container.

8 Conclusion

Field experience with the iso-integrity measurement architecture in the form of the CC3 supervisor demonstrates the utility of this approach to the practical application of securing limited service network endpoints. The inclusion of Trusted Execution Technology by Intel in the vPro feature set suggests the necessary hardware technology for sophisticated behavioral attestation will be available in commodity platforms, particularly those focused on enterprise deployments.

A Linux based behavioral assessment system provides an economically viable way to use this commodity hardware to implement service providing endpoints which are capable of self detecting compromise. Advances in isolation technologies, such as containerization and virtualization, provide a framework to minimize the application development costs associated with using such platforms.

Measured application platforms represent a paradigm shift in how security architectures are developed. The notion of systems self-detecting an alteration in their behavior will be critical given the effectiveness being demonstrated by current detection and prevention systems.

The iso-identity integrity measurement architecture implements an extremely sensitive system for detecting variations in platform behavior. The CC3 implementation demonstrates that management interfaces can be implemented which assist in the tooling and development of integrity modeled systems.

Harnessing the effectiveness of integrity modeling will benefit from integration of these systems with type enforcing mandatory access control systems. Extending the identity paradigm to other subjects, such as network socket connections, will provide a mechanism to enable the detection of anomalous behavior which may be intra-dimensional in nature given only file based subject dimensionality.

References

1: G. Wettstein, Digital identity creation and coalescence for service authorization., United States Patent 7,325,143, Assignee: Linux Foundation, 2008
2: R. Sailer, X. Zhang, T. Jaeger, L. Van Doorn, Design and Implementation of a TCG-Based Integrity Measurement Architecture, 2004
3: G. Wettstein, ftp://ftp.enjellic.com/pub/hpd/hpd_driver-latest.tar.gz, 2015
4: D. Bernstein, Curve25519: new Diffie-Hellman speed records, 2006
5: Trusted Computing Group, TCG Infrastructure Working Group Integrity Report Schema Specification, 2006