trustifier

Inside the Design of Trustifier **KSE**™

ABSTRACT: *Insider attacks are the main cause of loss and theft of sensitive information, and abuse of information system assets in organizations.*

The only truly effective way to protect against insider threats is to use a "positive" security model, which denies access by default and enforces only the minimum necessary access to the users.

While conventional approaches such as, Multi-level Security, Domain Separation and Formal Methods have been theoretically well understood for decades, practical limitations in complexity, administrative overhead, and expense in traditional implementations have prohibited their wide-spread adoption.

In this paper, we introduce ***Trustifier KSE*** *(Kernel Security Enforcer)—a modular, injectable, kernel-level security reference monitor. KSE can be used to provide the deep levels of control needed in modern systems and networks, but without the usability issues that infest traditional positive-security models. We provide an overview of KSE internal design, provide definition of Security Primitives available in KSE and tie them to the first-principles, and discuss how the design approach helps with the ease of use in modern multi-homed, multi-OS, networked environments.*

Introduction

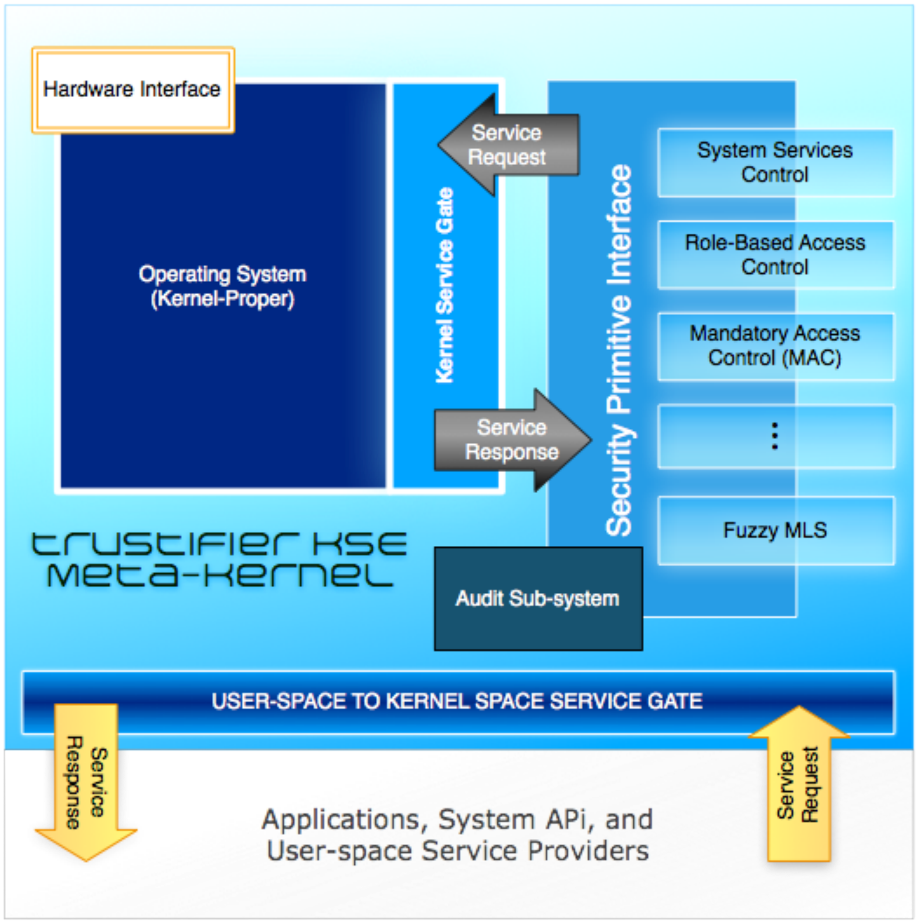
Trustifier KSE™ is a Kernel-level Security Enforcer that gives the *security owner* a new mechanism with which to define access and operational authorization within a system and across the network.

As the name suggests, Trustifier KSE enforcement occurs at the kernel-level within the operating environment. All existing and future applications, software systems, data and libraries are subject to the over-arching security rules within Trustifier KSE.

KSE: A Security System That Makes Business-Sense

In businesses, information security is still regarded as a necessary evil. This is further frustrated by a lack of comprehensive solutions. Most of the existing solutions only focus on perimeter security, and do not address the biggest threat of them all: insider-attacks. The handful that do, are extremely difficult to use and do not scale well.

KSE is designed to specifically address practical problems in modern cyber-security such as providing insider-attack protection including protecting against evil system administrators.


Hardware Interface
Operating System (Kernel-Proper)
Kernel Service Gate
Service Request
Service Response
Security Primitive Interface
System Services Control
Role-Based Access Control
Mandatory Access Control (MAC)
⋮
Fuzzy MLS
trustifier KSE meta-kernel
Audit Sub-system
USER-SPACE TO KERNEL SPACE SERVICE GATE
Service Response
Service Request
Applications, System APi, and User-space Service Providers

For example, consider a system with a user, Bob, whose work is deemed sensitive. We want to ensure that no one, including the system administrator can access Bob's files and data, and that Bob cannot maliciously share his sensitive information with anyone else.

Discretionary Access Control Systems cannot address this problem. Traditional Mandatory Access Control (MAC) security systems do address this problem, but are extremely difficult to manage. The primary reason for this is the complexity in specifying the rules. For instance, SE Linux requires 90,000+ rules by default on a per workstation or server environment in order to just get going.

Yet, if you need to stop an on-going attack immediately—by revoking the offending user's access privileges—SE Linux provides no practical way of achieving it, without requiring you to shut down the entire system.

This problem is not just limited to SE Linux. Pick any modern IPS, IDS, Role Based Access Control system, or a Mandatory Access Control system of your choice. They all suffer from either lack of capability or complexity of implementation, or both.

Now consider how KSE may be able to perform the same task:

The *security owner* can use the KSE Ranking Module to disallow the **root** user all access to a Bob's objects and directories in a single step:

```
$ trustifier kse rank user:bob \
  user:bob=1s⏎
```

Using the command above, the *security owner* can immediately make everything that Bob owns on the system subject to Mandatory Access Control rules and stop everyone, including **root** from accessing Bob's content while the system is running.

This, and all other KSE rules, are enforced in real-time. Which means that if someone is in the middle of looking at Bob's content, the moment that command is entered, access to that content is removed and any programs that are "looking" at Bob's files / directories are closed automatically.

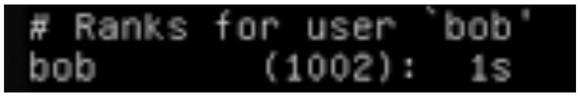
This is an example of the use of a KSE *Security Primitive*. In this case it's the *User to User Ranking Primitive*–one of the several KSE mandatory access control enforcers.

Security Primitives are mathematical objects inside Trustifier KSE which ensure that business security rules can be algebraically mapped to system security rules.

The representation of a business rule as an algebraic mapping is the key to KSE's security guarantees. Since the business rule is transformed into a reversible map AS-IS, its meaning is not lost in translation and is enforced as intended. This property is a unique feature in KSE and is necessary to build verifiable and consistent security rules.

As a consequence, when a *security owner* queries KSE about user:bob the result that comes back is identical to the rule that was added:

```
$ trustifier kse show user:bob⏎
```



```
# Ranks for user `bob'
bob          (1002):   1s
```

The information that Bob's documents are secret is represented by exactly one ubiquitously enforced rule; all of Bob's files, directories, any software Bob is running, and all memory that is used by Bob is protected by the system from everyone.

Security owners no longer have to use several thousand rules—as would be typical of other systems—in order to isolate sensitive information, and then still worry about whether or not they are secure.

KSE Access Control Security Facilities

Trustifier KSE provides these Security Primitives for several well understood security facilities:

- Least-Privilege System Services
- Labeled Security
- Domain Separation
- Mandatory Access Control
- Mandatory Integrity Control
- and more...

These common access control facilities allow *security owners* to create access rules using familiar and intuitive concepts. Since these facilities are implemented as security primitives, business rules don't become opaque when implemented. This, in-turn, ensures that data in your systems and networks is contained within well understood boundaries of use, processing, distribution and access.

In KSE, business rules are represented using security primitives. Each security primitive is an algebraic mapping function. KSE guarantees that if we have a set of Business Rules $\boldsymbol{R}_{\mathrm{b}}$, there exists a security primitive transform *S*, such that

$$S{:}R_b \mapsto R_s \qquad \textbf{(A)}$$

where $\boldsymbol{R}_{\mathrm{s}}$ is a bijection of $\boldsymbol{R}_{\mathrm{b}}$: In other words, every KSE security primitive *S* has a reverse transform S^{-1}:

$$S^{-1}{:}R_s \mapsto R_b \qquad \textbf{(B)}$$

that maps the security primitive rules back to the business rules.

KSE Authorization Security Facilities

In addition to several *Access Control* facilities, KSE provides *Authorization* Security Primitives that extend the concepts of access control to operations control **post access** grants.

We know that with *Access Control* we can ensure that while Bob can manipulate his sensitive data and not share it with un-authorized users. However, KSE goes further with *Authorization Rules.*

Authorization Rules allow the *security owner* to specify more interesting and practical bounds on Bob. In particular, a *security owner* can limit what Bob can and cannot do while manipulating the sensitive data.

For example:

- Bob can read all his files, but not read all his files in one day, so he can't copy them to a USB stick and take them out of the office; KSE can detect if Bob is attempting this over a longer period surreptitiously.
- Bob cannot read a particular file in combination with selected others because once he does, he can combine all that information in his head to gain intelligence beyond his scope—think knowing some of the ingredients of a secret recipe on-the-fly, once Bob knows A, B and D. KSE can stop Bob from knowing C.
- Bob can only make phone-calls to specific numbers after reading sensitive data.

etc…

KSE and Multi-User Multi-Homed Applications:

KSE integrates with the operating system at a fundamental level, and provides security, access control and authorization over users, groups, files, directories, networks, shared folders, and anything else you can think of, within the context of a normal operating system. In addition KSE also provides its security facilities to applications of multiple users within them.

KSE has plugin modules and APIs that can be used to extend access and authorization boundaries to within an application.

Let's look at an example of protecting a LAMP stack web-based application: KSE can interact at a deep level with Apache Web-server processes, provide audit and control plugins to the MySQL database, and security primitives for PHP scripts at the Linux OS layer. This ensures that flow of sensitive information from the back-end database to the Apache server, through the PHP processor out over the socket to an end-user is not going to be usurped by a malicious system admin, web-admin, PHP admin or database admin.

Concluding Remarks

Achieving cyber-security is a costly endeavor without the proper tools and capabilities. KSE greatly simplifies the process of implementing cyber-security within an organization, not only cutting down the time and cost to implement, but also the life-cycle cost to manage security. Its design enables *security owner*s to create rules that can be verified by both technical and non-technical stake-holders, and that delivers real value in the cyber-defense of your organization.

Contact Information

For more information contact us at sales@trustifier.com or visit us on our website at **www.trustifier.com**

Terms and Definitions

Discretionary Access Control

Discretionary Access Control is a type of access control where access to any content is granted at the discretion of the owner of the content.

See also: *Mandatory Access Control*

Mandatory Access Control

Mandatory Access Control is a type of access control where access to any content (formally ***objects***) is **mandated** by the *security owner*s and not the owners of the content (***objects***).

See also: *Discretionary Access Control*

Role-Based Access Control (RBAC)

The central notion of RBAC is that permissions are associated with roles, and users are assigned to appropriate roles.

Security Owner

Security Owners are the Security Officers and Security Administrators within a business unit.

Security Officer

Security Officer is the person responsible for creating security policies within a business unit.

Security Primitive

Security Primitive is a KSE security component that provides security owners to map business security rules to system security rules.