



ARCHITECTING HIGH-SECURITY SYSTEMS FOR MULTILATERAL COOPERATION

In cooperation with:



Federal Office
for Information Security

secunet

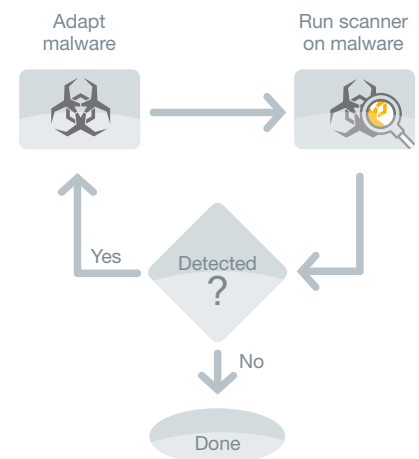
Threats to critical infrastructure and sensitive information

WEAKNESSES in operating systems, web services and user applications are regularly discovered and may be exploited to obtain sensitive information or to disrupt critical systems. Most recent incidents like the discovery of the Stuxnet worm demonstrated that attackers with great technical expertise and extensive resources are more than just an abstract threat. It is possible to analyze systems to find so-called zero-day threats which are unknown to system vendors and undetectable by current anti-malware products. General purpose systems are usually defenseless against this kind of directed attacks.

The public release of vast quantities of classified documents suggests that the security architectures currently employed do not sufficiently protect sensitive information from flowing off on a large scale. Not only is this a threat to classified governmental information but also to valuable intellectual property and research results of commercial enterprises. The appealing competitive advantage and the low risk of discovery render computer-based industrial espionage a serious threat.

Can reactive security measures protect us?

Countermeasures suggested to provide protection from malware include the use of a current virus scanner, an intrusion detection system and regular software updates. While those measures may have their justification, they do not reliably protect from directed attacks. An attacker is able to reproduce the targeted system and improve an attack until it succeeds. Such an adaptive approach makes it very likely that directed attacks are successful and remain undetected.



Large software cannot be secure

THE MARKETS' constant demand for new features and functionality leads to a continuous increase of commercial software complexity. Moreover, software vendors' schedules are usually very tight and time-to-market periods must be short to gain advantages over competitors. Hence, producers of document viewers, web browsers, operating systems and other complex pieces of software often do not have appropriate resources available to build sufficiently secure software for critical environments.

Regardless of the development process larger and more complex software is more likely to contain security-critical errors. While the probability of flaws can be reduced by a suitable process, very large software components statistically expose numerous critical errors.

Users frequently require large and complex applications in order to work productively, regardless of the fact that flaws in this software may provide an attacker with a target. Hence security must be maintained even when vulnerable and defective software has to be used. As we cannot depend on the correct enforcement of security properties by untrusted applications it is essential to be able to separate unrelated software and to securely manage information flows.

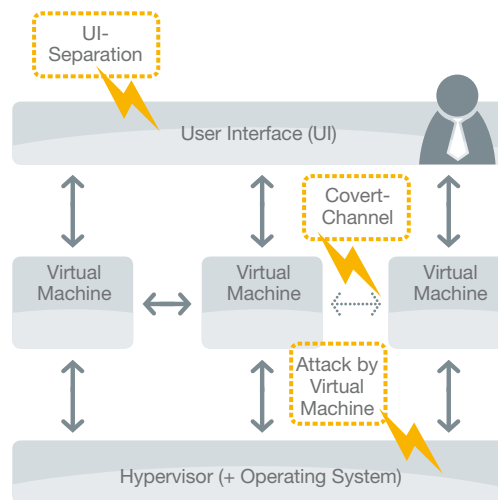
Who is accessing your bank account?

The problem with large and defective software and weak separation is easily demonstrated by a simple scenario: A user has legitimate access to both a spreadsheet application and a web browser. The spreadsheet application is used to review external files, while the browser is used to manage financial data. In this setup an attacker could manipulate a spreadsheet so that a flaw in the spreadsheet application is exploited to trigger arbitrary actions on behalf of the user. This includes for example access to a user's bank account through the browser. Can you trust an architecture that allows any spreadsheet to manage your bank account?

Virtualization alone does not solve the problem

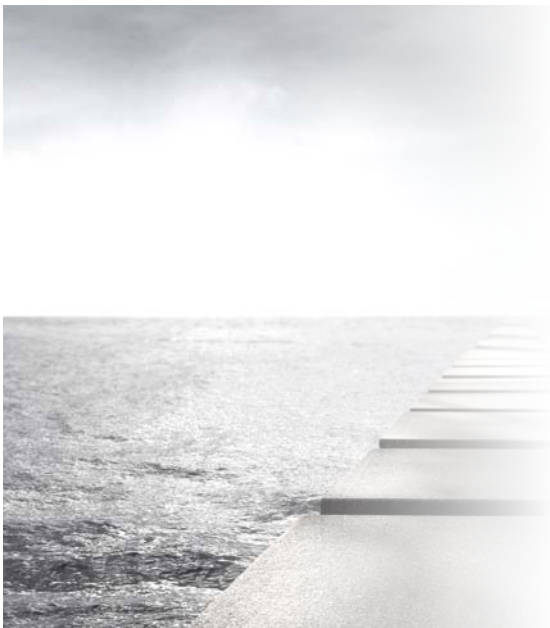
USING HYPERVISORS to isolate large, defective operating systems and the applications running on top of them is a step in the right direction. However, virtualization is not sufficient to achieve and does not imply separation. Many solutions are integrated into large and potentially defective operating systems which may be attacked. Even standalone hypervisors consist of a vast quantity of highly specialized code which is likely to contain errors that are exploitable by an attacker.

While policies for controlling the information flow between Virtual Machines (VMs) may exist in commercial hypervisors, the problem of so-called covert channels is rarely addressed. Covert channels do in fact pose a serious threat to system security as they are unintended communication paths within the system that may be used to violate the security policy. Furthermore, the graphical user interface is often hosted by one large, potentially erroneous virtual machine. If compromised by an attacker, this VM may forge windows of other VMs, record keystrokes or take screenshots of confidential information.



How serious is the covert channel problem?

In high-security systems covert channels become a serious issue. They may allow attackers to establish high-bandwidth communication between unrelated VMs or to retrieve secret cryptographic keys. One of many examples is an attack against a VM performing the RSA algorithm. By analyzing the behavior of the CPU's branch prediction cache, an attacker can observe the key-dependent execution of the encryption process. This will enable him or her to derive significant portions of the secret RSA key.



An architecture and methodology suitable for high-security systems

FOR SECURE processing of sensitive information of different origins using a single computer system that is connected to an untrusted network, reliable separation is an important precondition. In our architecture, security critical portions can be implemented in a trustworthy fashion. The communication between individual components is well-defined and in accordance with a security policy.

Our architecture enables the enforcement of critical security properties while preserving productivity by strictly separating security decisions from the user's ordinary workflow. The risk of operating errors is thereby reduced and training costs are minimized. Virtualization combined with reliable separation allows the user to work within a familiar environment and permits a cost-efficient integration of legacy software. While reliably maintaining different security domains, the architecture allows for independent management of security domains by different unrelated organizations.

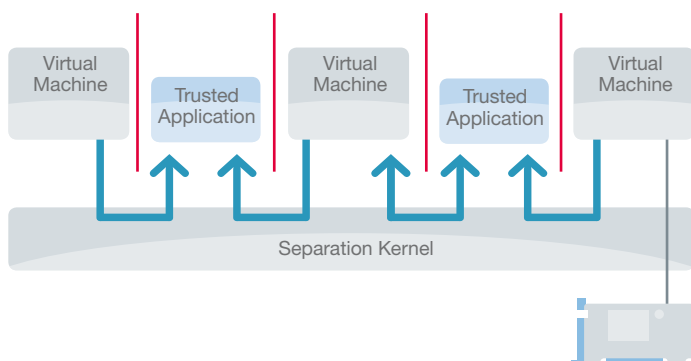
Multiple Independent Levels of Security (MILS)

MILS IS AN emerging security architecture which employs a so-called Separation Kernel (SK). This minimal operating system kernel enforces a strong isolation between components and permits communication and hardware access strictly according to a well-defined security policy. The key advantage of the MILS architecture is the opportunity to securely run trusted and untrusted applications on the same system. This opens up the possibility to move security critical functionality into very small components which are developed trustworthily and which undergo extensive evaluation. Non-critical parts of the system are implemented in large untrusted or less trusted software which allows for cost-efficient reuse of exchangeable off-the-shelf components.

Why is MILS suitable for high-security?

Separation Kernels have undergone formal verification of their security properties and consist of less than 10 kSLOC. They can be trusted to isolate components and to enforce communication and resource access policies. SKs allow the implementation of typical OS functionality, of device drivers or of security policies inside isolated components.

System decomposition into a MILS system results in a set of small trusted applications cooperating with large untrusted ones. If done properly, it leads to an overall complexity reduction of the trusted parts of a system. This is the precondition for cost-efficient formal verification of trusted components and therefore the key for both security and trustworthiness of a platform.



Small components – the key to trustworthiness

EVERY NON-TRIVIAL piece of software does have flaws. The number of defects a software contains depends on various factors like, for example, the development process, the language and tools as well as the experience of designers and developers. Independent of the actual error rate it is always true that the larger the software, the more flaws it does have. This leads to the conclusion that critical components of a MILS system must be as small as reasonable.

For several subsystems of our MILS platform we have successfully realized a decomposition into small, trusted components. One of those subsystems is an implementation of the ESP protocol defined in the Internet protocol security suite IPsec. A rough comparison of the significant lines of code (SLOC ¹) of our network encryption with a minimized Linux setup shows that the approach is highly effective. Our trusted components exhibit a complexity of less than a 50th of the 770 000 SLOC a minimized Linux setup would at least consist of:

Component	SLOC	Component	SLOC
Cryptographic algorithms	2 300		
Encryption/decryption subsystem	1 300		
Communication library	260		
Kernel API	600	Minimized	
Initialization	70	Linux	
Separation Kernel	~10 000		
Total	~15 000	Total	>770 000

How to create small critical components systematically

Designing trusted components is a systematic process in which all parts of a software component are categorized into security-critical and non-critical functionality. A functionality is critical if confidentiality, integrity or authenticity can be compromised when controlled by an attacker. For ESP we can decompose the replay window algorithm to make it truly minimal. The critical part only checks the sequence number of packets to increase strictly monotonically, while the untrusted part maintains a packet cache and feeds packets into the trusted component. This reduces the critical code size to a single line while maintaining security.

Formal verification is feasible with the right architecture

Techniques to be used for lightweight verification

The SPARK programming language ² was specifically designed to support the development of critical applications and formal verification. We use a standard Ada compiler (GNAT) to build SPARK applications with minimal runtime complexity. To be able to prove more advanced properties of our code we integrated SPARK with Isabelle, a powerful interactive theorem prover ³. Our extension to Isabelle is able to read in and reason about the verification conditions the SPARK tools produce and is freely available in the official Isabelle distribution.

THE MILS ARCHITECTURE and a strict decomposition into trusted and untrusted components give us the opportunity to target a formal verification of all critical software. Each critical component has got a size between a few hundred and a few thousand lines of code, for which a formal verification proves to be commercially feasible. Depending on the criticality of a component and the level of assurance that is required the depth of the verification can be adjusted. Our approach enables us to prove the absence of run-time errors for a piece of software and, optionally, to prove partial correctness:

Absence of run-time errors

For critical components we are using the SPARK programming language which does not rely on heuristics but is based on a profound mathematical model. Before compiling our source code we prove that the program always is in a well-defined state and does not show any undefined behavior. This eliminates a large class of errors like buffer overflows, integer overflows or use-after-free conditions.

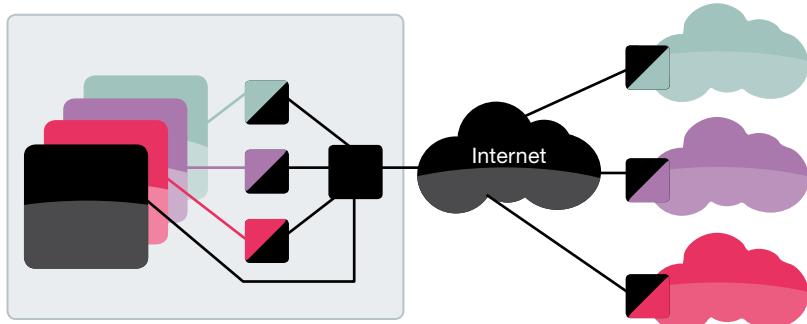
Partial correctness

When rigorous mathematical proofs of security properties are required, we can use SPARK's proof capabilities and show even more complex properties using the Isabelle theorem prover. This enables us to formalize and prove, for example that no network packet leaves the ESP encryption component unencrypted.

¹ All SLOC measurements generated using 'SLOCCount' by David A. Wheeler. ² <http://altran-praxis.com/spark.aspx>. ³ <http://isabelle.in.tum.de>

An interactive multilevel workstation

USING THE INTRODUCED architecture and methodology, we are developing an interactive high-security multilevel workstation which is able to concurrently handle multiple, strictly separated sessions. In fact it enables the user to securely access multiple, differently classified networks from within a single system. Nonetheless, the separation of those networks is maintained at all times. As different networks are not necessarily under control of one single organization our workstation design involves a concept to distribute responsibility. As a result, each session can be managed by a different organization independently. A secure graphical user interface (GUI) ensures an unambiguous correlation of user interaction to one well-defined session.



Network encryption – a case for small components

The network encryption part of the workstation greatly benefits from the small-components approach. Classical IPsec implementations are integrated into the IP stack of a monolithic operating system kernel. This makes the trusted code base as large as multiple hundreds of thousands lines of code.

Our security-critical ESP component only performs cryptographic operations while enforcing security-relevant properties like traffic limits for security associations. Other functionality such as routing, hardware drivers and the Internet protocol are completely handled outside ESP. This makes the TCB of our implementation 50 times smaller than a minimized Linux setup.

All cryptography is implemented inside small, trustworthy components, while the actual network stack and the device drivers are completely untrusted regarding confidentiality and integrity. Consequently, the workstation can be securely connected to arbitrary, untrusted networks like the Internet. Networking is completely handled by an untrusted virtualized environment, which facilitates an easy integration of new networking protocols, new hardware and a dynamic configuration process.

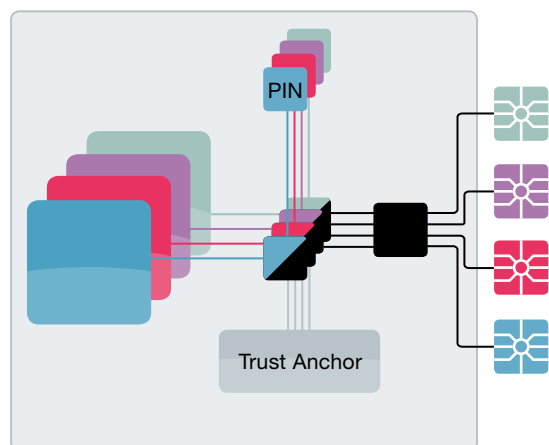
As software contained in untrusted parts of the system cannot compromise security no special development methodology is required even in high-security scenarios. This allows for very cost-efficient development and extension of networking capabilities.

Multilevel networking

OUR MULTILEVEL-CAPABLE workstation allows concurrent access to multiple, completely separated networks from within a single computer. Network traffic originating from the workstation is transparently encrypted and integrity protected employing strong cryptographic algorithms. Each network is accessed through a remote security gateway which decrypts traffic and enforces a security policy.

Multilevel management and authentication

MANAGING A MULTILEVEL SYSTEM can be challenging, especially if access to networks of different, independent organizations is required. Therefore our platform strictly separates the management of different sessions. Each session is operated with an individual, distinct smartcard. The user has to authenticate against each session using a trusted PIN dialog.



To authenticate the platform against the user smartcards and vice versa, a trust anchor is built into the system. That mechanism can ensure cryptographically that only authorized smartcards can be used with the system and that smartcards are used in authorized platforms only. This has the advantage that no dependency is created between organizations managing different sessions of a multilevel workstation and that different organizations do not have to trust each other.

Advantages of a secure channel

For each session a distinct smartcard is used which communicates with a session-specific endpoint. An endpoint is a trusted application that uses the platform's trust anchor to perform mutual authentication against a smartcard using public key cryptography. During authentication a tunnel is established between an endpoint and a smartcard, providing confidentiality and integrity protection.

The mutual authentication and tunneling concept has the major advantage that device drivers, smartcard readers and other infrastructure do not need to be trusted. Man-in-the-middle attacks against the smartcards can thus be prevented effectively.

Multilevel user interface

Threats to classical user interfaces

Classical user interfaces are inherently unsuitable to separate input and output of different sessions. Even if input and output were separated, on many systems a rogue application could create a new window which could receive keystrokes intended for another window (focus stealing).

Another attack is a malicious application or virtual machine forging a perfect lookalike of another session's window. Without suitable countermeasures a user may provide sensitive information to the wrong session as he or she has no chance to realize such an attack.

GRAPHICAL USER INTERFACES of multilevel secure systems have requirements that do not exist in common desktop environments. As a multilevel graphics subsystem potentially has access to input and output of different sessions it must be trusted not to intermix the information it is processing and to be free of covert channels between separate clients. In addition to the information-flow aspect, a secure GUI must enable the user to reliably distinguish screen output of different sessions and to know to which session keyboard and mouse events are redirected.

For our multilevel workstation we chose the most simple and most obvious concept available: By pressing a secure attention key (SAK) a user can switch the whole screen from one session to another. A titlebar at the top of the screen unambiguously identifies the active session, while the rest of the screen is freely available for the session itself. The titlebar is not accessible by any session and is changed by the trusted graphics application only if the user requests a change of the session. Keyboard and mouse input are exclusively redirected to the currently active session.

The concept provides secure, unforgeable separation of different sessions and is implemented with very low complexity. It is easily understood by the user and requires little training.

MULTILATERAL-SECURE SYSTEMS

Most existing software is known to be insecure but is essential in today's workflows. Reactive measures cannot sufficiently protect from ever-increasing threats like data theft or information warfare. The conflicting demand for close cooperations requires a system architecture that provides

- multiple compartments, protected from one another
- independent management of compartments by different entities
- seamless integration of existing applications and business processes

These requirements can be met cost-efficiently by our high-security architecture because

- security only depends on a very small amount of software
- riskless integration of existing software is feasible
- only little effort for formal verification of critical properties is required
- extension and adaption to specific customer requirements are easily possible



Federal Office
for Information Security

Bundesamt für Sicherheit
in der Informationstechnik (BSI)
Postfach 20 03 63
53133 Bonn, Germany

Contact

Dr. Thomas Östreich
Phone: +49 (0) 228 99 9582 5466
thomas.oestreich@bsi.bund.de

secunet

secunet Security Networks AG
Kronprinzenstraße 30
45128 Essen, Germany

Phone: +49 - 201- 54 54 - 0
Fax: +49 - 201- 54 54 -1000
E-mail: info@secunet.com

Contact

Dr. Kai Martius
Head of Business Unit High Security
Phone: +49 201 54 54-3504
kai.martius@secunet.com

Alexander Senier
System Security Architect
Phone: +49 201 54 54-3544
alexander.senier@secunet.com