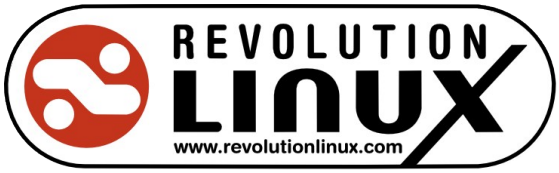


Report W7701-101396/001/QCC

Department	Consulting
Authors	Jean-Philippe Luiggi, Julien Desfossez
Date	December 22th 2010

Table of Contents

1. Statement of Work

DRDC Valcartier must conduct a state-of-the art study and a comprehensive analysis of existing Linux-based monitoring and security systems, which are utilized at runtime to monitor the software activity and states (as well as static files) of its computing devices (and not of its network).

The result should be a comprehensive analysis of Linux-based monitoring and security systems.

2. Introduction

Information security concepts have become an essential component of our current usage of computing technology.

An information system's security can be viewed as a group of various elements that are more or less resistant to external threats.

The multiplicity of software solutions available on the market combined with increasingly diverse needs creates many security issues. To address these, a multi-level security ("MLS") policy must be implemented.

Although there are many different ways to achieve this, the defence in depth approach is one of the most effective. More than a simple perimeter protection, this line of defence builds several levels of security and isolates them from one another, quite similarly to the way ancient fortified castles were built.

In order to conduct the requested study, we will base ourselves on this principle and focus our research on the various parts of a Linux system, starting with the lower levels (network) and gradually working our way up to the applications. We will then extrapolate this information to categorize the systems' security tools based on the risks they address.

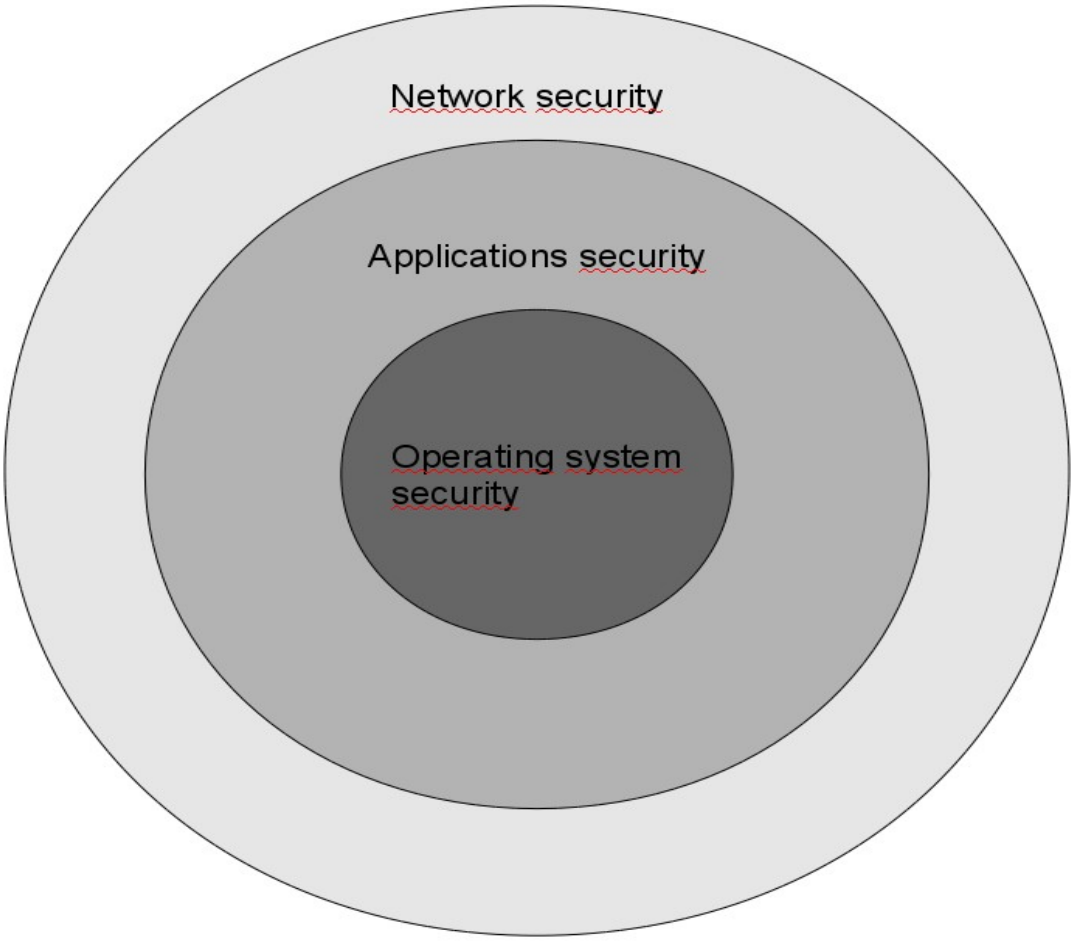
The following aspects of information system security will be studied:

- network and telecommunications security;
- application and data security;
- operating systems security;

- system access security.

The following diagram illustrates the chosen approach:

Security zones



3. Network Security

3.1. Network Filtering: The Firewall

Introduction

The firewall, which is often the first line of defence of a computing infrastructure, has significantly evolved since its beginnings [W2-1].

Its function is to provide secure, controlled network connections between various parts of a network (both external and internal) [W2-2].

There are different types of firewalls [W2-3] that act at different levels within the infrastructure [W1-4], including:

- Network firewalls (stateless and statefull)
- Application firewalls
- Identity-based firewalls

The first generation of filter systems used basic criteria and simply filtered packets without keeping track of their state (to this day they are still referred to as stateless firewalls). The concept consists of taking each packet independently from the others and comparing to a list of preconfigured rules.

This method had its limitations as it did not offer the ability to precisely determine what needed to be blocked. Therefore, the next step was to implement statefull firewalls, which successfully resolved these issues.

One way to study state management concepts is to take a closer look at how the Transmission Control Protocol (TCP) works. Unlike UDP, TCP handles actions related to connections, such as identifying who created them, making sure each packet follows the previous one in sequence, and so forth.

This simple notion greatly simplifies the process, as it is easy to tell the system to grant access to all incoming packets that are associated with the initiated connection. It is to be noted that the Netfilter firewall used in Linux can keep track of states with UDP and ICMP.

Application firewalls were implemented for a very specific purpose: to filter packets based on their contents (the data itself) rather than their container (the connection data).

The FTP protocol is a prime example of this. This protocol works (in active mode) by opening ports dynamically and exchanging IP or TCP level information — addresses and network ports, respectively — at the application level. To achieve this, the concept of “application” must be understood and each application must be known in order to be managed by the security solution.

Another type of application-level firewall verifies if the contents of the packets complies with the configured protocol. This is increasingly becoming important, as it is more and more common for applications to use certain network ports for convenience purposes. Indeed, software clients connecting to peer-to-peer networks are often configured to use TCP port 80 (normally dedicated to HTTP) in order to bypass filtering rules that are too restrictive.

An identity-based firewall possesses two different functions. The first is to enable a connection based on the person who is attempting to connect, rather than just based on the connection data. The second consists of determining whether a user has the right to access a particular application.

The latter function is very useful as it provides the ability to decide that “John Doe can connect to the Internet using Firefox, but not Chrome,” for example.

Port knocking is another restriction functionality based on firewall usage; this method may be used to modify a firewall’s behaviour.

The advantage of port knocking is that it opens/closes ports dynamically and in real time. The method consists of temporarily establishing connections to a set of specific network ports according to a particular logic. The port knocking process runs in background mode and checks (for example) the network connections that were attempted on the firewall. If a particular sequence is found, the server memorizes the source IP address (the one which initiated the sequence) and changes the firewall’s rules in order to open the corresponding network port (described in the configuration file) for this sequence.

An interesting thing to note is that only the machine that initiated the request can verify if the operation was successful. If the identification process fails, no new port will be opened and no information will be sent. In the same manner, the port can be closed using a connection sequence that differs from the first one; its function will be to block access to the previously opened network port using the firewall.

The major advantage of this type of manipulation is the complexity it creates for an attacker; unless the correct method is known to the attacker, he or she will have to test 655354 packets (65535^3 for the three port sequence, multiplied by 65535 attempts each time to discover the port that is possibly open).

History

Firewalls go back a long way (1987) and various RFCs exist on the subject (RFC 1636, etc.) [W2-2].

Most experts attribute authorship of this solution to Digital Equipment Corp (DEC); the corporation produced their first implementation of this type of system towards the end of the 1980's.

The solution was named gatekeeper.dec.com and was designed by Jeff Mogul, Brian Reid, and Paul Vixie. DEC SEAL was the first commercial product of this sort, but authorship for the first statefull firewall is attributable to Nir Zuk from Check Point, who created the technology towards the middle of the 1990's.

Tools

- Ebtables: http://ebtables.sourceforge.net/
- Netfilter: http://www.netfilter.org/
- NuFw: http://www.nufw.org/
- L7-filter: http://l7-filter.sourceforge.net/
- knockd: http://www.zeroflux.org/projects/knock

References

- [W2-1] The roots of the firewall: http://www.avolio.com/pres/FirewallsHistory_files/v3_document.html
- [W2-2] Freed, N. (2000). Behavior of and requirements for internet firewalls. Retrieved (2010, August 10) http://www.ietf.org/rfc/rfc2979.txt
- [W2-3] A History and Survey of Network Firewalls: http://www.cs.unm.edu/~treport/tr/02-12/firewall.pdf
- [W2-4] - Overview of firewalls at the various ISO levels: http://www.cs.unm.edu/~treport/tr/02-12/firewall.pdf

- [W2-5] Whitehouse W, Yamamoto M (2004) Knock Knock, Sandstorm Enterprises Yarden J (2005) http://techrepublic.com.com/5100-1009-5798871.html
- [W2-6] Jeanquier S (2006) An Analysis of Port Knocking and Single Packet Authorization. http://www.securethoughts.net/spa/An Analysis of Port Knocking and Single Packet Authorization (Sebastien Jeanquier).pdf

3.2. Network detection: NIDS

Introduction

Blocking packets is not enough to ensure the legitimacy of the network traffic passing through the equipment mentioned in the previous chapter.

Indeed, it is possible to conceal malevolent sequences even within legitimate traffic (on authorized and validated ports).

Let's take the HTTP (TCP/80) protocol as an example. Logically, connections to a web server will be authorized by the firewall, but what about requests attempting to take advantage of cross-site scripting security vulnerabilities?

The packets are legitimate at the network and even at the application level (they comply with the HTTP protocol). However, they are not legitimate in terms of system security as they attempt to exploit an application vulnerability. Clearly, an additional device must be implemented to complement the action of the firewall.

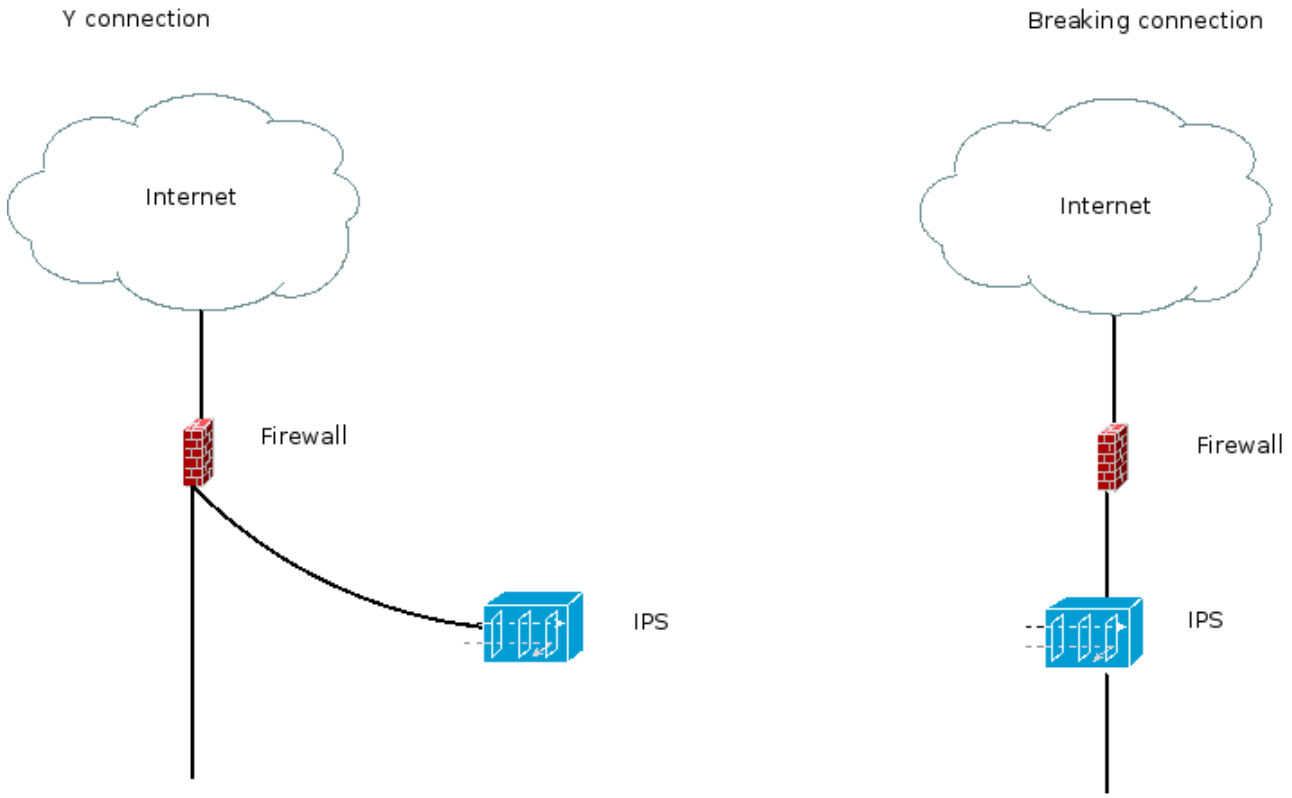
This type of equipment is called an intrusion detection system (IDS). Here, the word intrusion is used in a broad sense because a simple CSS type attack would be detected as an intrusion.

Network intrusion detection systems can be passive or reactive (IPS), software or hardware-based, and they can either use a library of signatures or check for protocol irregularities.

Their function is to search for specific attack sequences by analyzing network traffic and verifying its innocuousness. If a particular sequence is identified as potentially malicious (true or false), the device sends an alert to the security administrator or, in the case of an IPS, blocks the packet in question.

There are several things to keep in mind when it comes to an IPS. First of all, it is important to understand that packets will be blocked if the need arises, which means it is essential to eliminate as many false positives (false alarms) as possible to avoid blocking legitimate requests.

When implementing an IPS, special consideration must also be given to its placement. The following diagram illustrates the two main setup methods.



The Y connection is the easiest to set up, and if a physical failure (breakage) of the IPS occurs, it will have very little impact on the overall operation of the solution. In this operating mode, the equipment must interact with the firewall in order to block traffic. The IPS must therefore reconfigure the rules as needed in order to minimize exposure to threats.

In the second scenario, the equipment cuts through the network and can directly block packets considered to be malicious.

It is very important to make sure a physical failure would not compromise the integrity of the network connection setup. For instance, a simple electrical power failure must not prevent packets from passing

through the system, as it is preferable to run the risk of letting malicious packets get through rather than compromise the entire operation of the system because of a faulty connection.

Whether an IDS or IPS is used, both solutions will use one of the following methods to search for security issues.

The first method is identical to the functionality performed by most antivirus systems and consists of searching a database of signatures to find the binary patterns observed on the network. If a match is found, an alert will be sent out or the packet will be blocked (in IPS mode).

One of the basic premises of a signature-based approach is that it will only be effective against known threats; all unknown threats will pass through the system without any difficulty. This is an important consideration that is often overlooked by administrators.

The other approach consists of searching for functional anomalies, including the following:

- Unusual network traffic (in terms of volume, type, date/time of occurrence, etc.)
- Inconsistent usage of computer resources compared to what is normally observed (CPU load, number of processors, etc.)

In sum, normality is a key concept of this approach — identifying what is normal and what isn't.

Thus, when using this type of technology, one must first and foremost identify what constitutes a normal mode of operation in order to be alerted when activity outside of this scope is observed.

One of the benefits of this approach compared to a signature-based system is that attacks that have not yet occurred can possibly be detected.

History

The work carried out by Mr. James P. Anderson in 1980 established the foundation of IDS. [W3-1]

In 1986, Dorothy E. Denning, assisted by Peter G. Neumann, published [W3-2] a work that formed the basis for many of the systems we use today. The model described in their work used various types of statistics to detect functional anomalies; its result was the IDES intrusion detection system offered by SRI International, which ran on Sun's workstations.

Towards the end of 1998, a product called Snort was conceived and has since become the most widely used intrusion detection system, with nearly 300,000 registered users.

In 1999, the LBNL, in Berkeley (CA), released a new open-source tool named Bro [W3-3]. Although it is not as widely known and used as Snort, the tool is worthy of mention, one of its strengths being its dynamic protocol analysis functionality [W3-4], that is, its ability to perform precise protocol analysis without regard to the number of the network port used.

More recently, The Open Information Security Foundation (OISF) [W3-5], an organization created using funds from various institutions including the US Department of Homeland Security (DHS) [W3-6], developed and released a tool called Suricata [W3-7], which offered several new and highly useful Features (use of multiple processors, CUDA [W3-8]).

Tools

- Bro: Anomaly based, http://bro-ids.org
- Snort: Misuse based, http://www.snort.org
- Suricata: http://www.openinfosecfoundation.org/

References

- [W3-1] Anderson, James P., "Computer Security Threat Monitoring and Surveillance" Washing, PA, James P. Anderson Co., 1980.
- [W3-2] Denning, Dorothy E. "An Intrusion Detection Model," Proceedings of the Seventh IEEE Symposium on Security and Privacy, May 1986, pages 119–131
- [W3-3] “Bro: A System for Detecting Network Intruders in Real-Time” - ftp://ftp.ee.lbl.gov/papers/bro-CN99.ps.gz
- [W3-4] Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection - http://www.icir.org/robin/papers/usenix06/
- [W3-5] OSIF - http://www.openinfosecfoundation.org/
- [W3-6] DHS - http://www.dhs.gov/index.shtm
- [W3-7] Suricata - http://www.openinfosecfoundation.org/index.php/download-suricata

- [W3-8] CUDA - http://www.nvidia.com/object/cuda_home_new.html

3.3. Network Monitoring

Introduction

When it comes to information systems monitoring, there are two different areas to consider:

- Computer systems security;
- Security as it relates to reliability

Let’s start by examining the latter. Reliability details the implementation of devices used to supervise one or more elements of the IT infrastructure and alert the administrator in case of problems.

These problems include system overloads, server shutdowns, faulty network connections, security issues, etc.

Today, a wide variety of solutions [W4-1] address the aforementioned issues at various levels. Unfortunately, these are not the only things that can go wrong within an infrastructure. Some equally damaging threats are much more insidious.

The standard way to monitor an infrastructure is to test its elements; if one of these does not respond or if its response time is longer than usual, an alert must be sent out. The same goes for irregular network traffic, which can be observed when a significant and constant load is measured on the TCP 443 (HTTPS) port. Although this may seem harmless at first, it can prove to be a major security issue. Indeed, this behaviour is not typical of the protocol and may therefore indicate the presence of a network tunnel. Another example is when new equipment is detected as being connected to the network. In the worst-case scenario, a warning occurs; in the best-case scenario, no access whatsoever is granted.

In either of the aforementioned situations, detecting these problems is no easy task; it requires implementing various solutions and possessing the appropriate knowledge to manage them. An effective strategy requires a combination of system and network skills, analytical skills, knowledge of the field (types of attacks and tools), and so forth.

History

It is difficult to identify the first tool that was specifically designed to monitor behaviour since a simple ping command can achieve this.

A document [W4-2] published in 1995 shows that many solutions already existed at that time.

Tools

- Arpwatch: http://ee.lbl.gov/
- Mon: https://mon.wiki.kernel.org/index.php/Main_Page
- Munin: http://munin-monitoring.org/
- Nagios: http://www.nagios.org/
- Opennms: http://www.opennms.org/

References

- [W4-1] http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html
- [W4-2] http://www.slac.stanford.edu/~cottrell/tcom/survey3-results.html

3.4. Network Reporting

Introduction

Blocking or detecting network access is not enough in itself; to ensure accurate protection, it is essential to have an effective tool that can display usage reports or alerts.

A software solution whose main purpose is to display information in a simple and effective way will greatly simplify the task of the system administrator, who must deal with the proliferation of various data coming from many different sources.

Whether the activity consists of reading the connection logs of a server or firewall or the alerts sent by an intrusion detection system, it is essential to extract the essence of what comes in to ensure alerts are sent out if required — and as soon as possible.

Different types of software solutions exist for this purpose, and the method used to display data also varies. Display modes include spreadsheets, excerpts from connection logs, graphs, fixed or animated images, etc.

All solutions must offer the most exhaustive view possible.

History

There are no specific dates to mention here because recent operating systems offer the ability to log out what goes on in the system.

Even a simple syslog and its related connection logs can constitute a reliable source of data.

Tools

- Lire: http://www.logreport.org/
- Webfwlog: http://devel.webfwlog.net/index.php

3.5. Network Analysis

Introduction

Reading a network usage report is often enough to suspect or realize that something is wrong. One can then decide to perform a complementary analysis and implement the relevant methods to achieve this.

Not only does this approach require the proper tools, but it is also important to plan ahead in implementing them. In this case, the strategy consists of capturing network packets in a short term perspective.

Indeed, the proliferation of high-speed architectures [W5-1] has made it difficult to capture all frames and also store and analyze them effectively (without any loss) unless advanced technologies are used [W5-2], [W5-3].

Therefore, the contents of this section will be limited to short-term solutions; long-term solutions will be discussed in the following chapter.

In Linux, this process is almost always based on the use of the PCAP programming interface and its associated library, LIBPCAP [W5-4].

This library is frequently used by many open source software programs destined to analyze traffic, including protocol analyzers, network supervision tools, intrusion detection tools, etc.

Although a lot of products can be used from the command line, many solutions also have a graphical interface, both for implementation and presentation purposes.

History

PCAP is the de facto standard, but other similar tools exist. An example is Snoop for the Solaris operating system, which has a different file format compared to PCAP and was defined in RFC 1761 (February 1995).

Tools

- Ngrep: http://ngrep.sourceforge.net/
- Tcpdump: http://www.tcpdump.org/
- Wireshark: http://www.wireshark.org/

References

- [W5-1] http://en.wikipedia.org/wiki/10-Gigabit_Ethernet
- [W5-2] http://www.ntop.org/TNAPI_HwFiltering.html
- [W5-3] http://www.ntop.org/PF_RING.html
- [W5-4] http://www.tcpdump.org/

3.6. Network Audit

Introduction

This section is the continuation of the previous one and will focus on the network audit. Although a network audit can be performed using the tools mentioned in the previous section, this approach would have its limitations.

The reason is that the work involved here is very time consuming. The analysis performed as part of a network audit can span several days (or weeks). The actions are nearly identical to those of a network analysis, but their scope is different. The tasks of a network analysis often involve the packet level; an audit generally uses a more general approach in order to gain a global view of what is going on.

This process demands a whole different logic and the implemented solutions will therefore differ from or complement the tools of the network analysis.

The Netflow protocol [W6-1] is an example of a frequently used solution in this context. Developed by Cisco for its network routers and switches, the protocol is now supported by a variety of platforms, including Linux.

This solution has become the de facto standard to perform network audits. Indeed, it is very easy to implement Netflow probes that will analyze the activity occurring on the equipment and report it to a collector, which will provide a precise visual picture of what is going on.

History

NetFlow is a network protocol developed by Enterasys Networks (formerly known as Cabletron) and Cisco Systems to collect IP traffic data. Although it was initially a proprietary solution, NetFlow is now supported by many other network and software systems.

The industry greatly apprehended the possibilities the technology had to offer [W6-2].

Netflow was described in RFC 3954 and has since been amended by the IPFIX protocol, which has become one of the industry's standards. IPFIX is described in RFC 5101, RFC 5102, etc.

Tools

- Argus: http://www.qosient.com/argus/
- Flow-tools: http://www.splintered.net/sw/flow-tools &
- http://code.google.com/p/flow-tools/
- Nfdump: http://nfdump.sourceforge.net/
- Nmap: http://www.insecure.org
- Flowviewer/FlowTracker/FlowGrapher: http://ensight.eos.nasa.gov/FlowViewer

References

- [W6-1] Cisco netflow, definitions:
  http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html
- [W6-2] Cisco Netflow, a defacto standard:
  http://www.qosient.com/argus/argusnetflow.shtml

4. Software Security

4.1. Introduction to Application-Related Issues and Their Solutions

Over time, network defence solutions have become more powerful and reliable. As a result, piracy threats have carried over to the application level.

The reasons for this are simple and intuitive: although it is easy enough to define a network architecture with the appropriate filters, controlling what happens at the application level is much more complex. This is because applications are created by developers from everywhere in the world, who use different languages and operating systems (note that secure coding practices will not be discussed here).

4.2. Application Firewall

Introduction

An application firewall is a sort of “software lock” that controls an application or service’s access, input, and output.

Its function consists of monitoring and, when necessary, blocking applications according to established operating rules. Firewalls can have one of two different approaches: a network-based approach or a host-based approach.

In a network-based approach, the device is used as a network firewall but acts at the application level. A network firewall operates at Layer 7 of the OSI model [W7-1] and therefore has the ability to inspect the contents of network traffic, block certain requests and/or viruses and even prevent client software vulnerabilities from being exploited.

Examples of these solutions include proxy servers, which in most cases are designed for web services.

In a host-based approach, the software monitors an application's operations with greater precision. It examines the data passed through system calls instead of or in addition to the network stack and decides whether to send an alert or block a request, if applicable.

It is to be noted that a system firewall will only protect the workstation on which it is installed; it is useless to the neighbouring machines.

Application-level filters are certainly beneficial when it comes to monitoring activity because they understand the semantics of the applications used. In other words, they understand the language and/or protocol associated with these actions, allowing for easy detection of anomalies.

For various reasons, it is not infrequent for applications to be on network port addresses they normally shouldn't use. A regular network firewall cannot take any action against this. A software firewall, on the other hand, is precisely designed to verify whether the data it receives matches the established rules.

It is to be noted that other elements besides HTTP can be filtered here, including database access.

History

In June of 1991 [W7-2], Digital Equipment Corporation (DEC) offered the first commercial product of this kind, called DEC SEAL. Another product named Firewall Toolkit (FWTK) was released in October of 1993 [W7-3].

Tools

- Squid: http://www.squid-cache.org
- Delegate: http://www.delegate.org
- mod_security: http://www.modsecurity.org/
- GreenSQL: http://www.greensql.net

References

- [W7-1] OSI Model: http://en.wikipedia.org/wiki/OSI_model
- [W7-2] History of the firewall: http://www.cisco.com/web/about/ac123/ac147/ac174/ac200/about_cisco_ipj_archive_article09186a00800c85ae.html

- [W7-3] FWTK: http://www.fwtk.org

4.3. Intrusion Detection: HIDS

Introduction

Host-based intrusion detection systems (commonly referred to as HIDS) are, as their name implies, placed on servers and/or workstations and dedicated to monitoring a piece of hardware or an operating system.

Easy to deploy and available on many operating systems, HIDS are an essential complement to their network-based counterparts.

This type of software operates as a system service (or “daemon”in Unix terminology) and analyzes various types of data, including connection logs, files, directories, etc.

The system’s functional architecture is often based on a client-server model.

A system is installed on each machine and monitors the information coming from the hardware or operating system. These systems then report to another system, which analyzes the situation according to the rules defined by the administrator.

Various aspects can be monitored, including the machine’s activity, current processes, utilization of system resources or changes brought to the file system.

Indeed, it is possible to record the state of a directory or of its contents and to send out an alert if a change occurs, for example, if an element is added or modified.

The monitoring process is the following: a library of signatures is built, which contains the cryptographic fingerprints of the files destined to be monitored. These signatures are then compared to the data recorded over time. If a discrepancy occurs (regarding the file, its date, its access privileges, etc.), an alert is sent out.

User behaviour is by no means ignored by these security measures; indeed, anomalies pertaining to system usage can easily be detected by scanning the connection logs. The monitoring module will detect when people log on at unusual hours, use particular commands, etc.

This type of tool is useful for monitoring everything system-related and is a solution of choice to complement network intrusion detection tools.

History

Intrusion detection systems were first developed in the 1990's. Haystack Labs' Stalker products were the first of these solutions to appear on the market; these were later licensed to Sun [W8-1].

Tools

- Ossec: http://www.ossec.net/
- AIDE: http://sourceforge.net/projects/aide/
- Tripwire: http://www.tripwire.com/
- Sec: http://simple-evcorr.sourceforge.net/
- Prelude-ids: http://www.prelude-technologies.com/fr/bienvenue/index.html

References

- [W8-1] Haystack Labs Licenses Its Active Security Technology To Sun http://www.cs.indiana.edu/~kinzler/pubs/webstalker.html

4.4. Application Monitoring

Introduction

Most companies and system administrators would agree that application monitoring is indispensable [W9-1]. In a Unix environment, the system load and CPU usage are usually taken into account, but often overlooked is the proper operation of applications.

A mistake that is often made, for example, is to consider a messaging system functional based on the fact that a service is listening on network port SMTP (TCP/25). Thus, application monitoring is often necessary.

This type of security has an impact on all levels of the application system [W9-2], starting with the operating system at the very bottom. In a previous chapter, we studied host-based intrusion detection systems. It is essential to make sure these systems function properly and to have the ability to know when a service has been stopped or never started in the first place, otherwise our security model is impaired.

This is not the only problem system administrators must deal with. Data volume is another issue that can be misleading if it is too intrusive.

Managing large volumes of data coming from multiple sources increases the risk of obtaining too much information. Although this may not be an issue in many situations, it must be considered in the context of information security.

This means that various data sources must be taken into account and that their essence must be extracted, otherwise something important could be missed.

Possible means to address this include (partial list) data correlation tools and application test tools.

History

Since the very beginnings of the Unix operating system, tools have been developed to test its operation. For example, a simple shell script can be used to monitor a messaging system by sending an email and making sure it gets delivered to the mailbox.

Tools

- Nagios: http://www.nagios.org/
- Splunk: http://www.splunk.com/
- Ossim: http://www.alienvault.com/

References

- [W9-1] IBM - http://www.ibm.com/developerworks/websphere/library/techarticles/0304_polozoff/polozoff.html
- [W9-2] Application monitoring: http://adminschoice.com/application-monitoring

4.5. Application Reporting

Introduction

Having security tools is essential, however, it is equally important to have an overall view of the data they retrieve.

The system administrator must be aware of what goes on within the system, which includes knowing when packets are blocked by the firewall, when users attempt to connect to secure resources and even keeping track of data unrelated to security (web server traffic).

This phase differs from the previous one in that it focuses on presenting information. This is done by creating dashboards that offer an overall view of the current situation.

These dashboards can exist in different forms:

- Spreadsheets
- Excerpts from connection logs
- Graphs
- Etc.

Whatever format is used, the goal here is to provide the most comprehensive view possible of what is going on within the system. A simple syslog [W10-1] file issued by the eponymous daemon is one of the many tools that can be used to achieve this.

History

It is difficult to map out the exact chronology of this type of technology as log systems have existed since the very beginnings of the Unix operating system.

Tools

- Lire: http://www.logreport.org/

References

- [W10-1] Syslog - http://en.wikipedia.org/wiki/Syslog

4.6. Application Analysis

Introduction

The purpose of an application security test is to evaluate an application's security level. Application tests can be performed on web-based applications or on rich applications (for example, applications using a heavy client-server model).

The test is normally divided into two parts:

Black-box testing: done without accessing the source code, without any identifiers or any technical information on the software.

Gray box testing: a user account is required to verify the application's security from the perspective of a legitimate user having access to the application.

Different methods can be used depending on whether the subject of the audit is a web application or a traditional desktop application.

In the former case, the test will focus on verifying technical vulnerabilities such as:

- XSS type vulnerabilities [W11-1],
- SQL injections [W11-2],
- CSRF vulnerabilities [W11-3],
- Remote/local file includes[W11-4],
- Attacks targeting sessions and cookies [W11-5],
- Vulnerabilities specific to certain technologies,
- Management of ViewState parameters of the .NET language [W11-6].

In the second case, vulnerabilities pertaining to the audited application will be tested:

- Heap overflow [W11-7],
- Stack overflow [W11-8],
- Format string [W11-9],

- Race condition [W11-10],
- etc.

Searching for bugs is another type of verification performed here. These can include “logical” bugs, that is, those that occur due to an error in the design of the algorithm, as well as bugs associated with authentication and access privilege management (ex. a user who can access certain data he/she normally shouldn’t have access to, based on his/her privileges).

Now that we have defined the concept of software analysis, let’s take a look at how it’s done. A software analysis can either be performed via the software (fuzzing [W11-11] is an example of this) or by more conventional means, such as auditing the source code (which will be described in the following chapter).

Fuzzing is a software testing technique that consists of providing (random) data to the inputs of a program. If the program fails (by crashing or generating an error), this usually proves that defects are present and must be corrected.

Examples of a program’s inputs:

- the network
- peripheral devices (keyboard, mouse, etc.)
- environment variables
- files
- resource limitations (memory, CPU, etc.)
- etc.

One of the particular aspects of fuzzing is that the tests are very simple to write; no knowledge of the system’s operation is required. Finding vulnerabilities is therefore easy, which is why fuzzing is commonly used for this purpose.

Given the wide variety of fuzzers available [W11-12], one can very realistically conceive that hackers will take advantage of the possibilities offered by these tools.

History

The first publication mentioning fuzzing dates back to December 12th, 1990: An Empirical Study of the Reliability of UNIX Utilities [W11-13], written by Barton P. Miller, Louis Fredriksen, and Bryan So.

Note: During tests, 25 to 33% of utility programs from any version of Unix failed.

Tools

- Nessus: http://www.nessus.org/
- Metasploit: http://www.metasploit.com/
- Openvas : http://www.openvas.org/
- Spike: http://www.immunitysec.com/resources-freesoftware.shtml
- Webscarab: http://www.owasp.org/index.php/OWASP_WebScarab_NG_Project
- Fusil: http://bitbucket.org/haypo/fusil/wiki/Home

References

- [W11-1] XSS - http://en.wikipedia.org/wiki/Cross-site_scripting
- [W11-2] SQL injections - http://en.wikipedia.org/wiki/SQL_injection
- [W11-3] CSRF - http://www.cgisecurity.com/csrf-faq.html
- [W11-4] Remote file inclusion - http://en.wikipedia.org/wiki/Remote_file_inclusion
- [W11-5] Cookie - http://en.wikipedia.org/wiki/HTTP_cookie & http://www.imperva.com/resources/glossary/cookie_poisoning.html
- [W11-6] View state - http://msdn.microsoft.com/en-us/library/ms972976.aspx
- [W11-7] Heap overflow - http://en.wikipedia.org/wiki/Heap_overflow
- [W11-8] Stack overflow - http://en.wikipedia.org/wiki/Stack_overflow
- [W11-9] Format string - http://en.wikipedia.org/wiki/Format_string_attack

- [W11-10] Race conditions - http://www.sans.edu/resources/securitylab/race_cndtns.php
- [W11-11] Fuzzing attacks - http://www.owasp.org/index.php/Fuzzing
- [W11-12] List of fuzzers - http://www.infosecinstitute.com/blog/2005/12/fuzzers-ultimate-list.html
- [W11-13] History of fuzzing - http://www.cs.wisc.edu/~bart/fuzz/fuzz.html

4.7. Application Audit

Introduction

White-box testing is another category of application tests. Unlike black-box testing, white-box testing is done in a cooperative manner. Performed with the consent of the application's creator, these tests focus on analyzing the way the application is made (its code and construction).

A source code audit [W12-1] also constitutes a comprehensive analysis of an application's code. The audit is done by combining tools that will automate the process and allow for mass data treatment. These tests also rely on the skills of the IT experts who audit the code.

These tests are based on a source code audit, which can be carried out in one of two ways:

Manual analysis: in this case, the audit is performed by a single person. The main problem with this approach is the limited capacity of work one human being can achieve per day (in terms of auditing code lines).

Static code analysis: uses formal methods based on the source code to evaluate an application's behaviour without actually executing it. The testing process is initiated using an automated tool; the results it yields are then verified and qualified by an IT expert.

Notes:

White-box testing can cover a wider scope of vulnerabilities than black-box testing.

An abundance of tools currently exist to analyze the various types of source code (languages).

History

It is difficult to trace the exact historical references of these practices as programmers have always carefully verified the quality of code. More recently, an open source BSD type operating system [W12-2] experienced a significant lag compared to Linux because its entire system libraries had to be re-written [W12-3] to ensure their security. This issue is not only related to BSD operating systems; Linux distributions can also be affected [W12-4].

Tools

- Lint: http://docs.sun.com/source/806-3567/lint.html
- Splint: http://splint.org/
- Valgrind : http://valgrind.org/

References

- [W12-1] Audit code - http://en.wikipedia.org/wiki/Code_audit
- [W12-4] OpenBSD – http://www.openbsd.org
- [W12-1] BSD code - http://www.freebsdworld.gr/freebsd/bsd-family-tree.html
- [W12-1] Debian audit FAQ - http://www.debian.org/security/audit/faq

5. Operating System Security

5.1. Hardening

Introduction

Hardening is a process that aims to secure the operating system. In an ideal scenario, the content installed on the system is reduced to an absolute essential minimum. The premise of this approach is that the less objects are installed, the smaller the vulnerability surface will be.

The process will affect software, software libraries, and tools as well as users and access privileges that are configured for the various parts of the system.

Software, libraries, and tools were studied previously; we will now take a look at user privileges and access control.

Unix offers a set of basic user privileges (Read/Write/Execute) for the owner of a file and the group to which this user belongs. This user-based approach can lead to problems if the legitimacy of the actions performed — either at the user or administrator level — is not supervised.

For example, an operating system cannot easily differentiate a malware [W13-1] from a user attempting to access a resource. A malware can easily assume the digital identity of a legitimate person and attempt to usurp its actions.

Another known issue in the Unix environment is the omnipotence of the root user, towards whom all of the system's privileges are directed. Most system tasks can only be performed by the root user, therefore all of this user's actions must be carefully scrutinized.

It is a known fact that the system administrator can access each element of a system, however this can create problems when piracy occurs. In order to avoid these risks, kernels now make it possible to configure user privileges using rules called capabilities [W13-2].

Capabilities can differentiate access levels, but they fall short when it comes to determining the exact nature of objects that are targeted. Therefore, they cannot offer any kind of access granularity.

AppArmor, RSBAC, and SELinux are just a few of the solutions that can be used to increase the security of a system.

Developed by Novell, AppArmor [W13-3] offers the ability to attach a security profile to each program in order to restrict its access privileges; this type of access control is referred to as MAC [W13-4].

AppArmor proactively protects the operating system and applications against various threats (including zero days). The use of profiles offers the ability to precisely define who has access to what and, most importantly, to set access levels.

Alike AppArmor, RSBAC offers enhanced access control compared to the basic Unix security settings. RSBAC also offers various security models. Its capabilities include:

- Kernel user management (no more /etc/passwd)
- Network control support

- Symlink redirection (symlinks can redirect to another location by role, by uid, by security level or by remote address)
- Secure delete (mandatory secure deletion per file, directory or whole filesystem)
- Hides processes easily using a kernel option

Lastly, SELinux is a tool that uses an approach similar to that of AppArmor, though it functions differently.

SELinux identifies system objects using their inode rather than their path name and separates the application from the access policy and its definition.

This approach has its advantages when compared to AppArmor. With AppArmor, a restricted file can become accessible when a hard link is created to it, but SELinux would prevent this type of access. On the other hand, in SELinux, data that is inaccessible may become accessible when applications update the file by replacing it with a new version, while AppArmor would continue to deny access to the data.

History

The practice of hardening an operating system to meet specific needs was apprehended early on by various institutions, including the National Security Agency (NSA) [W13-5] in the U.S.

In 1992 and 1993, the NSA worked with the Secure Computing Corporation (SCC) [W13-6] to develop two prototypes called DTMach and DTOS [W13-7]. The NSA continued this work in collaboration with the University of Utah and these efforts led to the development of the Flask security architecture; its name was changed to SELinux once it was integrated to the Linux kernel. Several other major contributors took part in the project, including NAI Labs and MITRE.

Tools

- AppArmor: http://www.novell.com/linux/security/apparmor/
- Grsecurity: http://grsecurity.net/
- RSBAC: http://www.rsbac.org/
- SELinux: http://www.nsa.gov/research/selinux/index.shtml
- MAC: http://en.wikipedia.org/wiki/Mandatory_access_control

- DTOS: http://www.cs.utah.edu/flux/fluke/html/dtos/HTML/dtos.html
- FLASK: http://www.cs.utah.edu/flux/fluke/html/flask.html

References

- [W13-1] Malwares - http://en.wikipedia.org/wiki/Malware
- [W13-2] Capabilities - http://linux.die.net/man/7/capabilities
- [W13-3] Novell - http://www.novell.com/home/
- [W13-4] MAC - http://en.wikipedia.org/wiki/Mandatory_access_control
- [W13-5] NSA - http://www.nsa.gov/
- [W13-6] SCC - http://en.wikipedia.org/wiki/Secure_Computing
- [W13-7] DTOS - http://www.cs.utah.edu/flux/fluke/html/dtos/HTML/dtos.html

5.2. Virtualization

Introduction

Virtualization encompasses various methods that add an extra layer of security by allowing several versions of an operating system to run in parallel.

How does this enhance the security of a system? First of all, the kernel is often modified in order to allow several software versions to run at the same time. Thus, when a threat is attempted, this strategy will either block it or yield different results from one version to another.

This approach also involves isolating one server from another; as a result, different instances of a service (ex. a web server) can run in parallel on the same physical server, which reduces the surface exposed to a global system compromise incident in the event that one of the virtual machines presents a vulnerability.

Let's take a look at the security of virtual machines.

Virtual machines are becoming more and more popular mainly because of the benefits they provide in terms of management and ease of use. There are different kinds of virtualization technologies; the four main implementations are:

- full virtualization using dedicated processor instructions.
- para-virtualization, where the guest OS is designed to run on a virtualized architecture.
- full emulation, where each instruction is interpreted by the physical host.
- context-based virtualization, where only the processes are isolated; the guest OS possesses the same kernel, memory, and devices as the host.

Full virtualization is the most common approach; leading products such as VMware [W14-1], Xen [W14-2], VirtualBox[W14-3], and KVM [W14-4] make use of the virtualization extensions added by Intel and AMD in their recent processors, VT-x [W14-5] and AMD-V [W14-6], respectively.

Using these extensions, a host can let a guest execute all of its unprivileged code natively; in this mode, the processor is completely dedicated to the guest for a period of time. As soon as the guest needs to execute a privileged instruction (such as accessing a privileged register or a device), the control is given back to the host, which completes the execution and sends the result back to the guest. This method offers good performance, especially when combined with para-virtualized drivers and hardware dedicated to virtualization. Para-virtualization is one of the approaches that can be used with Xen [W14-3]; here a Linux kernel is provided and is dedicated to running over another Linux kernel. This approach offers great performance because the guest knows exactly how to execute the privileged instructions in the best possible way.

Full emulation technologies such as Qemu [W14-9] are mainly used for test and development purposes. They are rarely used in production environments because of their performance. Indeed, each instruction needs to be interpreted by the host OS.

The context-based method approaches virtualization in a different way. A context is a set of processes running on the host OS with restricted capabilities. The processes running inside a guest can only interact with each other. They run inside a chroot and are usually assigned a dedicated virtual network interface. On top of that, the host OS can restrict the contexts with quotas on CPU, disk, and memory usage.

The main technologies in this area are Linux-Vserver [W14-10], OpenVZ [W14-7], and LXC [W14-8]. Given that all the resources are shared and the processes are normal Linux processes, the performance of these “guests” is relatively similar to that of the host [FIXME: ref Fernando's Thesis ?].

Security

The problem associated with virtual machines is the inherent risk of a VM compromising the host or another VM located on the same physical host. When using the full virtualization technology, the hypervisor is designed to give the guest OS a complete native environment. The guest does not need to know it is running in a virtual machine: it is running its own kernel, has a complete dedicated memory address space and sees the devices as its own.

The first step for an attacker would be to identify the hypervisor. As we saw in the introduction, a variety of software solutions provide the same technologies. There are a lot of methods to determine if a machine is running on bare metal or on a hypervisor. Some methods are obvious, such as checking the CPU model name (which is "QEMU CPU" by default in KVM and Qemu) or looking at device names that are commonly emulated (for example, Intel ICH AC'97 or a SoundBlaster 16 sound card commonly emulated in VirtualBox and VMware). Also, when KVM runs with para-virtualized drivers, it can export special CPU flags, which could be used to identify the virtualization technology. Advanced techniques based on timing can be used to determine if a machine is running on a hypervisor, but these cannot precisely determine which hypervisor is used. Johanna Rutkowska also introduced the Red Pill technique [W14-12] to detect the presence of a virtual machine monitor based on the way interrupts are handled inside a machine.

There are a few known techniques that can be used to compromise a virtual machine environment but most of them are outdated. The Bluepill rootkit [W14-11] is a project to create an invisible malware based on the hypervisor technology. The proof of concept was presented at Black Hat in 2006; although it is designed for Windows Vista, the concept could be used with any other OS. The project has since been abandoned and the source code does not seem available anywhere.

Given the robustness of today's virtualization technologies, most of the attacks targeting virtual machine environments attempt to take advantage of configuration problems. One common vector is the network; since all VMs share the same physical network card, an attacker can use this vector to bypass the firewall's rules.

History

System virtualization appeared in the 1960's as a means to partition IBM's mainframe computers and thus streamline hardware usage. Nowadays, computers based on Intel's x86 technology face the same optimization and flexibility limitations as their ancestors from the sixties.

In 1990, VMware designed a virtualization technology that is still being used on the vast majority of computers today.

Tools

- secvisor: http://www.cylab.cmu.edu/partners/success-stories/SecVisor.html
- sHype:http://www.research.ibm.com/secure_systems_department/projects/hypervisor/
- sVirt: http://vidéoprojecteur/page/SVirt
- KvmSec: A Security Extension for Linux Kernel Virtual Machines

References

- [W14-1] VMware: http://www.vmware.com
- [W14-2] VirtualBox: http://www.virtualbox.org
- [W14-3] Xen: http://www.xen.org
- [W14-4] KVM: http://linux-kvm.org
- [W14-5] Intel VT-x: http://www.intel.com/technology/virtualization/

- [W14-6] AMD AMD-V: http://www.amd.com/us-en/0,,3715_15781_15785,00.html
- [W14-7] OpenVZ: http://openvz.org
- [W14-8] LXC: http://lxc.sourceforge.net/
- [W14-9] Qemu: http://www.qemu.org
- [W14-10] Linux-Vserver: http://linux-vserver.org
- [W14-11] Blue Pill: http://theinvisiblethings.blogspot.com/2006/06/introducing-blue-pill.html
- [W14-12] Red Pill: http://invisiblethings.org/papers/redpill.html

5.3. Tracking Data

Introduction

Tracking data consists of monitoring any given program's activity and can be performed at the application level (ex: a web server) or at the system level (ex: the Linux kernel).

Tracking data is a common means of resolving various issues. Among other things, it can be used for debugging purposes [W15-1] or to monitor a particular behaviour within a program.

Tracking is basically following a program's operations every step of the way, similarly to recording all of its actions and verifying them, but at a very low level (near the system).

The software input data used to monitor what goes on can originate from the operating system's kernel (input/output of system calls, network activity, etc.) or from the applications. Although the output data can be provided in various formats, it is common to use software specifically designed for this purpose in order to maximize processing performance and data readability.

It is imperative to make sure that these activities do not interfere with the proper operation of the system. This means the tracking Features must not be enabled on a continuous basis (i.e. during the normal operation of the system). Even if these tools are optimized, using them on a continuous basis during production, for example, can considerably cripple the system's operation.

History

The original version of the strace utility was written by Paul Kranenburg for the SunOS operating system. It was inspired by the trace tool used for the same OS. The Linux version of this tool was created at the beginning of the 1990's. Systemtap and LTTng date back to 2005.

Tools

- strace: http://en.wikipedia.org/wiki/Strace
- gdb: http://www.gnu.org/software/gdb/
- ftrace, kprobe, ltrace, strace: included in Linux distributions
- LTTng: http://lttng.org/
- perf: https://perf.wiki.kernel.org/index.php/Main_Page
- systemtap: http://sourceware.org/systemtap/

References

- [W15-1] http://en.wikipedia.org/wiki/Debugging

5.4. Encrypting Data

Introduction

Data encryption is a method used to make a document unreadable to anyone who does not possess the key to invert it (the reverse process is called decryption).

On an information system, encryption is used to prevent data from being lost or compromised.

This means that a user who has the ability to access the physical media (hardware) does not necessarily have access to the logical media (the data).

When applied to computers, encryption can be performed at two levels:

- Filesystem level encryption [W16-1]
- Full-disk encryption [W16-2]

The first method consists of encrypting each of the files and/or directories found on the storage device individually; this phase is performed by the file system itself.

Note that this task only protects the data itself. Everything related to the directory structure, the names, sizes, and access/modification dates is untouched. This can be an issue in some cases; although it is impossible to read a drive's contents, a user can still know what is stored on the drive.

The second method consists of encrypting the drive in its entirety, which solves the aforementioned problem. One of the advantages of this method is its "all-or-nothing" approach; indeed, everything stored on the drive will be protected from unauthorized access, including the swap space.

It is also possible to combine both approaches, i.e. encrypting the entire disk and using filesystem encryption for certain parts of the system. A thing to keep in mind is that once an encrypted disk is decrypted, anyone can access the data. Therefore, the combined approach can be useful to restrict access to certain data (ex. when encrypting directories with different authorization levels).

History

The first version of the TrueCrypt software dates back to 2004. Over the years, many new versions of the software were developed with significant enhancements; Linux support became available as of 2006. Version 6.0a received a First-Level Security Certificate (Certificat de Sécurité de Premier Niveau, or CSPN) from the French Network and Information Security Agency (FNISA) [W16-3].

Tools

- Truecrypt: http://www.truecrypt.org/
- dm-crypt: http://www.saout.de/misc/dm-crypt/
- dmcrypt/LUKS: http://code.google.com/p/cryptsetup/
- EcryptFS: https://launchpad.net/ecryptfs

References

- [W16-1] http://en.wikipedia.org/wiki/On-the-fly_encryption
- [W16-2] http://en.wikipedia.org/wiki/Encrypting_File_System
- [W16-3] http://www.ssi.gouv.fr/site_rubrique54_certificat_cspn_2008_03.html

5.5. Using the System

Introduction

There are many different ways to connect to an information system. One of the most common methods is the login/password combination, but this approach obviously has its shortcomings.

- If the password is chosen by the user, there is no guarantee as to its complexity, which therefore makes it vulnerable to brute force attacks.
- Even if a password seems complex, its likeliness of being uncovered will increase over time, thereby decreasing its strength.
- The risk of having someone listen on network/keyboard activity is significant; this applies to both clear and encrypted data.

To address these issues, researchers developed a new approach architected around challenge-response concepts [W17-1]. The idea is to use a password for a single session.

An important consideration here is that the password is no longer chosen by the user, but rather a particular method is used to generate a list of passwords.

This approach virtually eliminates all of the aforementioned issues because:

- A password will only be used once,
- Passwords will be determined by the computer,
- The password changes constantly and therefore cannot be uncovered by brute force.
- It is also impossible to listen on the password, as it is not reusable.
- This concept is commonly referred to as the one-time password (OTP)[W17-2].

This is another reason why we recommend the port-knocking technique studied in chapter 2 [W17-3]: it provides selective access to a machine via the network.

History

The OTP concept was developed by Bellcore (now known as Telcordia).

Tools

- opie-server: this software is included in Linux distributions
- knockd: http://www.zeroflux.org/projects/knock

References

- [W17-1] Challenge Response - http://en.wikipedia.org/wiki/Challenge-response_authentication
- [W17-2] OTP - http://en.wikipedia.org/wiki/One-time_password
- [W17-3] Port Knocking - http://techrepublic.com.com/5100-1009-5798871.html

6. Product Analysis

6.1. Introduction

Before describing the products we believe deserve consideration, here is a list of criteria to provide an objective view of the requested recommendations.

The original list described in document/contract W7701-101396/001/QCC will be provided, with some additional criteria we consider relevant.

6.2. List of Criteria to Consider:

1. Available Features
   a) Main Features of the technology described
   b) Its strengths
2. Types of threats addressed (cyber, design flaws, other)
   a) The risks it prevents
   b) The type of protection it offers
3. Scalability of the technology (how can it be evolved)
   a) Its upgradability
   b) At what cost?

c) How can it be upgraded?

4. Theories, techniques, paradigms, and approaches of the technology (e.g. signature-based, anomaly-based, other types of approaches and paradigms).

a) Operating principle

b) Techniques used

5. Type of monitoring, protection

a) What type of monitoring is used?

b) How is the protection designed?

6. Purpose/occurrence of monitoring activities

a) What is monitored?

b) When?

7. Type, form, size, (etc.) of generated raw data

a) Description of output data

b) Format of output data (text/binary)

8. Overview of how it works (the main features), known flaws, limitations, problems

a) How does the technology work?

b) Known limitations

c) Known shortcomings

9. Requirements, required systems (hardware, software) to run each technology

a) Hardware required to run the technology

b) System required to run the technology

10. Data fusion methods, analysis methods, rules, other methods used to process and transform raw data

a) Data aggregation

b) Can the data be correlated?

11. Can traces of attacks be kept for further analysis? In what form? When? How?

a) How is the data maintained?

12. What type(s) of action(s) can be taken? In what circumstances? Using what components?

a) Can the technology take any action? If so, which parts of it perform the actions?

13. What knowledge bases (or other types of data) are needed/provided? When and how should they be updated (how do they work)?

a) Is it necessary to use a data source?

14. What programming language/script is involved?

a) Programming language used:

15. Ease of use and installation (for the user and administrator)

a) User-friendliness

16. Quality of the documentation

a) Quality of the documentation

6.3. Network firewall

Description of technology		Monitor and block network access	
Name of technology :	Netfilter	L7-filter	Nufw
Main features :	Support Ipv4/Ipv6/Statefull	Work at level 7 (OSI)	Able to set rules/applications/users
Features :	packet filtering framework	identify packets based on application layer data.	ID-based firewalling
Scalability :	High	High	High
Installation of component :	Easy	Medium	Easy
Remote's usage :	Via SSH	Via SSH	Yes
Impact of technology (kernel space) :	Yes	Yes	Yes
Impact of technology (user space) :	No	No	Yes
Object of surveillance	Network packets	Network packets	Network packets
How surveillance is conducted	Checking rules	Checking rules	Checking rules + user access
Type of data scanned	Network packets	Network packets	Network packets
Type of analysis on the data scanned	Network headers	Network headers + application data	Network headers + user access
Rules of surveillance	User defined	User defined	User defined
Actions of the technology	Block packets	Block packets	Block packets
Impact on performances	Small	Medium	Medium
Constraints : utilization, administration	Knowledge of networks concepts + operating system	Knowledge of networks concepts / application data + operating system	Knowledge of networks concepts + operating system + identifying users
Major limitations : functionality, efficiency, effectiveness, system coverage	No applications filtering	Not available in recent kernel	Price for a significant number of clients behind the tool
Improvements	Adding application filtering	Inserted into the current kernel branch	N/A

Usage in virtualization	yes	yes	N/A
Configuration options	Able to use patch-o-matic addons	depending the kernel	
System recovery			
Recovery techniques	console access	console access	console access
Impact on performances	none	none	none
System reports			
Type of reports	syslog or binary logs	syslog or binary logs	syslog or binary logs
Classification of attacks	N/A	N/A	N/A
Manage alarms	N/A	N/A	N/A
Notifications	Syslog	Syslog	Syslog
Interoperability			
Can exchange with other systems	Yes	Yes	Yes
easy use with other systems	Yes	Yes	No
Is an API available	Yes	Yes	Yes
Comparative analysis			
Important aspects	Easy to use	Application filtering	User filtering
Weakness	Lots of options	Not synced with current kernel	High cost for windows
Strengths	Lots of options	Able to classify applications	Filtering not only done on IP headers
Advantages	Included with Linux	The only one tool	Easy to manage
Inconvenient	Command line usage	Need an appropriate kernel	Lot of work depending the number of users
Constraints	Deep knowledge of options	Not all protocols are recognized	Deep knowledge of user/applications
Remarks	N/A	Need to be improved	N/A

6.4. Network detection

Description of technologies	Detecting security problems at network level	
Name of technology :	Snort	Suricata
Main features :	network intrusion prevention and detection system	network intrusion prevention and detection system
Features :		
Scalability :	High	Medium
Installation of component :	Easy	Medium
Remote's usage :	Via SSH	Via SSH
Impact of technology (kernel space) :	None	None
Impact of technology (user space) :	Yes	Yes
Object of surveillance	Network	Network
How surveillance is conducted	Analysis of network flow	Analysis of network flow
Type of data scanned	Network flow	Network flow
Type of analysis on the data scanned	signature, protocol and anomaly-based inspection	signature, protocol and anomaly-based inspection
Rules of surveillance	Match rules	Match rules
Actions of the technology	use rulesets to identify problems and attacks in incoming traffic	use rulesets to identify problems and attacks in incoming traffic
Impact on performances	Depending of hardware	Depending of hardware
Constraints : utilization, administration	Network positioning	Network positioning
Major limitations : functionality, efficiency, effectiveness, system coverage	Need to have an updated ruleset	Need to have an updated ruleset
Improvements	Able to use GPU	Able to use sn
Usage in virtualization	Yes	Yes
Configuration options	Use IPS functionality, PF_Ring	Use IPS functionality, PF_Ring
System recovery		
Recovery techniques	N/A	N/A

Impact on performances	N/A	N/A
System reports		
Type of reports	Log file	Log file
Classification of attacks	Yes	Yes
Manage alarms	Yes	Yes
Notifications	Syslog/Email	Syslog/Email

6.5. Network Monitoring

Description of technologies	Monitor the network	
Name of technology :	Nagios	Munin
Main features :	Checking availability	Graphing data
Features :	Check services	Graph all sort of data
Scalability :	High	High
Installation of component :	Easy	Easy
Remote's usage :	Yes	Yes
Impact of technology (kernel space) :	N/A	N/A
Impact of technology (user space) :	Small	Small
Object of surveillance	Services and system	Systems
How surveillance is conducted	Services and systems probes	System probes
Type of data scanned	Network port/system data	System probes
Type of analysis on the data scanned	Check for availability	N/A
Rules of surveillance	Check for availability depending on criterias	Data source breaching its defined limits
Actions of the technology		
Impact on performances	High with lots of device to manage	Small
Constraints : utilization, administration	N/A	N/A
Major limitations : functionality, efficiency, effectiveness, system coverage	No serious limitations	No serious limitations
Improvements		
Usage in virtualization	Yes	Yes
Configuration options	N/A	N/A

System recovery		
Recovery techniques	Restore	Restore
Impact on performances	N/A	N/A
System reports		
Type of reports	Email/Web	Email/Web
Classification of attacks	N/A	N/A
Manage alarms	Yes	Yes
Notifications	Yes	Yes

6.6. Network reporting

Name of technology :	**Lire**
Main features :	versatile log analysis software
Features :	transform raw data in network/computer system log files into valuable information for you
Scalability :	High
Installation of component :	Easy
Remote's usage :	Via external web server
Impact of technology (kernel space) :	None
Impact of technology (user space) :	Yes
Object of surveillance	Log files
How surveillance is conducted	N/A
Type of data scanned	Log files
Type of analysis on the data scanned	N/A
Rules of surveillance	N/A
Actions of the technology	Reporting
Impact on performances	Small
Constraints : utilization, administration	Use specific data log
Major limitations : functionality, efficiency, effectiveness, system coverage	N/A
Improvements	Add more plugins
Usage in virtualization	Yes
Configuration options	N/A
System recovery	
Recovery techniques	N/A
Impact on performances	N/A
System reports	
Type of reports	N/A

Classification of attacks	N/A
Manage alarms	N/A
Notifications	N/A

6.7. Network analysis

Description of technologies	Analyze the network	
Name of technology :	Tcpdump	Wireshark
Main features :	Network analysis	Network analysis + graphic interface (GUI)
Features :	Dissect network flow	Dissect network flow and present them via plugins
Scalability :	High	High
Installation of component :	Packaged in most distributions	Packaged in most distributions
Remote's usage :	Via SSH	Via display remote
Impact of technology (kernel space) :	Depends on traffic	Depends on traffic
Impact of technology (user space) :	Medium	Medium
Object of surveillance	Network data	Network data
How surveillance is conducted	Network analysis	Network analysis
Type of data scanned	Data flows	Data flows
Type of analysis on the data scanned	Depends on user request	Depends on user request
Rules of surveillance	Depends on user request	Depends on user request
Actions of the technology	Present data	Present data
Impact on performances	Small	Medium to High
Constraints : utilization, administration	Need to know command line parameters and TCP/IP headers	Need to know TCP/IP headers
Major limitations : functionality, efficiency, effectiveness, system coverage	Difficult to use depending on number of parameters + Need to know TCP/IP	Need to know TCP/IP
Improvements	N/A	Plugins

Usage in virtualization	Yes	Yes
Configuration options	N/A	N/A
System recovery		
Recovery techniques	N/A	N/A
Impact on performances	N/A	N/A
System reports		
Type of reports	Binary log	Depending on analyzed traffic
Classification of attacks	N/A	N/A
Manage alarms	N/A	N/A
Notifications	N/A	N/A

6.8. Network audit

Description of technologies	Audit the network		
Name of technology :	Argus	Flow-tools	Flowviewer suite
Main features :	Analyzing Packet Files and network streams	Tool set for working with NetFlow data.	Tool set for working with NetFlow data.
Features :	Conduct deep analysis of data	Make analysis on netflow flux	make analysis on netflow flux and display it
Scalability :	High	High	Medium-High
Installation of component :	Packaged inside Linux distributions	Packaged inside Linux distributions	Easy
Remote's usage :	Via SSH	Via SSH	Yes – via web external web interface
Impact of technology (kernel space) :	N/A	N/A	N/A
Impact of technology (user space) :	Medium	Low	Medium – depending on analysis
Object of surveillance	Network streams	Netflow streams	Netflow streams
How surveillance is conducted	User defined	User defined	User defined
Type of data scanned	Streams of octets	Netflow packets	Netflow packets
Type of analysis on the data scanned	User defined	User defined	User defined
Rules of surveillance	User defined	User defined	User defined
Actions of the technology	Reporting	Reporting	Reporting
Impact on performances	Low	Low	Low

Constraints : utilization, administration	Defining the things to check	Defining the things to check	Defining the things to check
Major limitations : functionality, efficiency, effectiveness, system coverage	If used in Network auditing, need to install the tool on each host. If used in Netflow mode, need to install Netflow probes	Need to install Netflow probes	Need to install both Flow-tools and netflow probes
Improvements			
Usage in virtualization	Yes	Yes	Yes
Configuration options	N/A	N/A	N/A
System recovery			
Recovery techniques	N/A	N/A	N/A
Impact on performances	N/A	N/A	N/A
System reports			
Type of reports	N/A	N/A	N/A
Classification of attacks	N/A	N/A	N/A
Manage alarms	N/A	N/A	N/A
Notifications	N/A	N/A	N/A

6.9. Application firewall

Description of technologies	Block unattended access to applications	
Name of technology :	mod_security	GreenSQL
Main features :	provide protection from a range of attacks against web applications	Protection against SQL injections, CSS and CSRF attacks
Features :	Provide protection from a range of attacks against web applications and allows for HTTP traffic monitoring, logging and real-time analysis.	The logic is based on evaluation of SQL commands using a risk scoring matrix. Moreover it can block known db administrative commands (DROP, CREATE, etc.)
Scalability :	High	High
Installation of component :	Easy	Easy
Remote's usage :	N/A	N/A
Impact of technology (kernel space) :	N/A	N/A
Impact of technology (user space) :	Low	Low
Object of surveillance	HTTP requests	HTTP requests
How surveillance is conducted	Analyze request	Analyze request
Type of data scanned	HTTP requests	HTTP requests
Type of analysis on the data scanned	Match specific rules	Match commands
Rules of surveillance	User or product defined	evaluation of Sql commands
Actions of the technology	Block requests	Block requests
Impact on performances	Low	Low
Constraints : utilization, administration	Need of good ruleset	Need of good ruleset

Major limitations : functionality, efficiency, effectiveness, system coverage	As it's a rule's based product, we need to have them accurate	The scoring Matrix needs to be accurate
Improvements	Need to be improved over time	Need to be improved over time
Usage in virtualization	Yes	Yes
Configuration options	N/A	N/A
System recovery		
Recovery techniques	Restore	Restore
Impact on performances	N/A	N/A
System reports		
Type of reports	Type of attacks	Type of attacks
Classification of attacks	Yes	Yes
Manage alarms	Yes – console available	Yes
Notifications	Yes	Yes

6.10. Host intrusion detection system

Description of technologies	Detect security problems on hosts		
Name of technology :	Ossec	Prelude-ids	Sec
Main features :	OSSEC is an Open Source Host-based Intrusion Detection System. It performs log analysis, file integrity checking, policy monitoring, rootkit detection, real-time alerting and active response.	Prelude is a Universal "Security Information Management" (SIM) system. Prelude collects, normalizes, sorts, aggregates, correlates and reports all security-related events	SEC is an open source and platform independent event correlation tool. It can be employed as an event correlator for any application that is able to write its output events to a file stream.
Features :	Analysis, Checks, the standard in open-source HIDS		Powerfull correlation engine
Scalability :	High	High	High
Installation of component :	Easy	Medium	Easy
Remote's usage :	N/A	N/A	N/A
Impact of technology (kernel space) :	N/A	N/A	N/A
Impact of technology (user space) :	Memory usage	Memory usage	Memory usage
Object of surveillance	System process/directory/log	System wide	System log
How surveillance is conducted	Rules based	Rules based	Rules based
Type of data scanned	Log, process, directory	System data	Log
Type of analysis on the data scanned	Match rules	Match rules	Match rules

Rules of surveillance	Program/User defined	Program/User defined	User defined
Actions of the technology	Alerts and actions	Alerts and actions	Alerts and actions
Impact on performances	Low	Low	Low
Constraints : utilization, administration	No packaging, install from sources	Some complexity of the product	Deep knowledge of regular expressions
Major limitations : functionality, efficiency, effectiveness, system coverage	N/A	Investment in time	Monitoring of system log only
Improvements	Need to follow product's development	Need to follow product's development	N/A
Usage in virtualization	Yes	Yes	Yes
Configuration options	Yes	Yes	N/A
System recovery			
Recovery techniques	Backup/Restore	Backup/Restore	Backup/Restore
Impact on performances	N/A	N/A	N/A
System reports			
Type of reports			
Classification of attacks	Yes	Yes	Yes
Manage alarms	Yes	Yes	Yes

Notifications	Yes	Yes	Yes

6.11. Host monitoring

Description of technologies	Monitoring at various level the applications		
Name of technology :	Nagios	Splunk	Ossim
Main features :	By using defined plugins, able to monitor everything	Data centralization	Complete, includes lots of programs
Features :	Can check everything	Centralize data	Provide a full view of the systems
Scalability :	Yes	Yes	Yes
Installation of component :	Easy	Easy	Easy
Remote's usage :	yes	Yes	Yes
Impact of technology (kernel space) :	N/A	N/A	N/A
Impact of technology (user space) :	Need to be tuned if a lot of hosts ans services are monitored	Memory used	Memory used
Object of surveillance	Services / system	Data log	System / network / Log
How surveillance is conducted	Using plugins	Using rules	Using rules
Type of data scanned	Service, programs, data	Data	Service, programs, data
Type of analysis on the data scanned	Match rules	Match rules	Match rules
Rules of surveillance	User defined	User defined	User defined
Actions of the technology	Alert	Alert and correlate	Alert and correlate
Impact on performances	Possibly high	High	High

Constraints : utilization, administration	Need to configure both all the client and the server	Need to configure all the clients for feeding Splunk with data	Deep knowledge of the system
Major limitations : functionality, efficiency, effectiveness, system coverage	Number of system and services monitored		Deep knowledge of the system
Improvements	N/A	N/A	N/A
Usage in virtualization	Yes	Yes	Yes
Configuration options	N/A	N/A	N/A
System recovery			
Recovery techniques	Backup Restore	Backup Restore	Backup Restore
Impact on performances			
System reports			
Type of reports	Visual/Mails	Visual/Mails	Visual/Mails
Classification of attacks	N/A	Yes	Yes
Manage alarms	Yes	Yes	Yes
Notifications	Yes	Yes	Yes

6.12. Application reporting

Description of technologies	Reports what's going with the applications running
Name of technology :	Lire
Main features :	Able to understand a lot of technologies
Features :	Read systems log and makes report about them
Scalability :	Yes
Installation of component :	Easy
Remote's usage :	Yes
Impact of technology (kernel space) :	None
Impact of technology (user space) :	N/A
Object of surveillance	Log files
How surveillance is conducted	Read log files
Type of data scanned	Present the log with a specific format
Type of analysis on the data scanned	N/A
Rules of surveillance	Depending on log
Actions of the technology	Parse log and reports on them
Impact on performances	Depending on the size of the log files
Constraints : utilization, administration	We need to import into the software the log file
Major limitations : functionality, efficiency, effectiveness, system coverage	Use log files the software can handle
Improvements	N/A
Usage in virtualization	Yes
Configuration options	N/A

System recovery	
Recovery techniques	Backup and restore
Impact on performances	N/A
System reports	
Type of reports	Web pages
Classification of attacks	N/A
Manage alarms	N/A
Notifications	N/A

6.13. Analyze the security of applications

Description of technologies	Analyze the security of applications		
Name of technology :	Nessus	Metasploit	Fusil
Main features :	The Nessus® vulnerability scanner is the world-leader in active scanners,	Metasploit provides useful information and tools for penetration testers, security researchers, and IDS signature developers.	Fusil the fuzzer is a Python library used to write fuzzing programs.
Features :	high-speed discovery, configuration auditing, asset profiling, sensitive data discovery and vulnerability analysis of your security posture.	This project was created to provide information on exploit techniques and to create a functional knowledge base for exploit developers and security professionals. The tools and information on this site are provided for legal security research and testing purposes	It helps to start process with a prepared environment, start network client or server, and create mangled files. Fusil has many probes to detect program crash: watch process exit code, process stdout/syslog for text patterns, session duration, cpu usage, etc.
Scalability :	Yes	Yes	Yes
Installation of component :	Easy	Easy	Easy
Remote's usage :	Yes	Yes	Yes
Impact of technology (kernel space) :	N/A	N/A	N/A
Impact of technology (user space) :	First load of plugins is time consuming	N/A	N/A
Object of surveillance	Vulnerability analysis	Network, program	Program problems
How surveillance is conducted	Probe the target	Probe the target	Probe the target

Type of data scanned	network services and applications	Network services and apllications	Applications
Type of analysis on the data scanned	Check for vulnerabilities	Check for vulnerabilities	Check for vulnerabilities
Rules of surveillance	User defined	User defined	User defined
Actions of the technology	Check for errors	Check for errors	Check for errors
Impact on performances	N/A	N/A	N/A
Constraints : utilization, administration	Need to carefully choose target (some tests are dangerous)	Need to carefully choose target (some tests are dangerous)	Need to carefully choose target (some tests are dangerous)
Major limitations : functionality, efficiency, effectiveness, system coverage	Need to keep synchonisation with product's updates (plugins)	Need to keep synchonisation with product's updates (exploits)	Need to carefully choose target
Improvements	N/A	N/A	N/A
Usage in virtualization	Yes	Yes	Yes
Configuration options	N/A	N/A	N/A
System recovery			
Recovery techniques	Backup restore	Backup restore	Backup restore
Impact on performances	Medium	Medium	Low
System reports			
Type of reports	Web	Web	Console
Classification of attacks	Yes	Yes	N/A
Manage alarms	Yes	Yes	N/A

Notifications	Yes	Yes	Yes

6.14. Application audit

Description of technologies	Source code audit	
Name of technology :	Splint	Valgrind
Main features :	Splint is a tool for statically checking C programs for security vulnerabilities and coding mistakes.	Valgrind is an instrumentation framework for building dynamic analysis tools.
Features :	Splint can perform stronger checking than can be done by any standard lint.	There are Valgrind tools that can automatically detect many memory management and threading bugs, and profile your programs in detail.
Scalability :	Yes	Yes
Installation of component :	Easy	Yeasy
Remote's usage :	N/A	N/A
Impact of technology (kernel space) :	N/A	N/A
Impact of technology (user space) :	Light	Light
Object of surveillance	N/A	N/A
How surveillance is conducted	N/A	N/A
Type of data scanned	Source code audit	Source code
Type of analysis on the data scanned	Splint does checks including unused declarations, type inconsistencies, use before definition, unreachable code, ignored return values, execution paths with no return, likely infinite loops, and fall through cases.	
Rules of surveillance	Errors in source code	Errors in source code
Actions of the technology	Report problems	Report problems

Impact on performances	N/A	N/A
Constraints : utilization, administration	Language is 'C'	Languages are 'C' and C++
Major limitations : functionality, efficiency, effectiveness, system coverage	Only one language	Only 2 languages
Improvements	N/A	N/A
Usage in virtualization	N/A	N/A
Configuration options	N/A	N/A
System recovery		
Recovery techniques	Backup and restore	Backup and restore
Impact on performances	N/A	N/A
System reports		
Type of reports	Problems detected	Problems detected
Classification of attacks	N/A	N/A
Manage alarms	N/A	N/A
Notifications	Yes	Yes

6.15. Operating system security

Description of technologies	Hardening of the Linux kernel. Add security profiles for the programs used.	
Name of technology :	Apparmor	Selinux
Main features :	Allow the system administrator to associate with each program a security profile that restricts the capabilities of that program	provides a mechanism for supporting access control security policies
Features :	Proactively protects the operating system and applications from external or internal threats, even zero-day attacks, by enforcing good behavior and preventing even unknown application flaws from being exploited.	- Clean separation of policy from enforcement - Controls over file systems, directories, files, and open file descriptors - Controls over sockets, messages, and network interfaces - Controls over use of "capabilities"
Scalability :	Yes	Yes
Installation of component :	Easy	Easy
Remote's usage :	N/A	N/A
Impact of technology (kernel space) :	Yes	Yes
Impact of technology (user space) :	Yes	Yes
Object of surveillance	Operating system	Operating system
How surveillance is conducted	Monitoring operating systrem	Monitoring operating system
Type of data scanned	N/A	N/A
Type of analysis on the data scanned	N/A	N/A
Rules of surveillance	Depending on administrator	Depending on administrator
Actions of the technology	Block/Allow access	Block/Allow access
Impact on performances	Yes	Yes

Constraints : utilization, administration	Need a fine tuning	Need a fine tuning
Major limitations : functionality, efficiency, effectiveness, system coverage	N/A	N/A
Improvements	N/A	N/A
Usage in virtualization	Yes	Yes
Configuration options	N/A	N/A
System recovery		
Recovery techniques	N/A	N/A
Impact on performances	N/A	N/A
System reports		
Type of reports	N/A	N/A
Classification of attacks	N/A	N/A
Manage alarms	N/A	N/A
Notifications	Yes	Yes

6.16. Virtualization

Description of technologies	Operating system virtualization	
Name of technology :	OpenVZ	KVM
Main features :	Provide contextualization for Linux	full virtualization solution for Linux
Features :	OpenVZ creates multiple secure, isolated containers (otherwise known as VEs or VPSs) on a single physical server enabling better server utilization and ensuring that applications do not conflict.	Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images.
Scalability :	Yes	Yes
Installation of component :	Depend on distribution	Depend on distribution
Remote's usage :	N/A	N/A
Impact of technology (kernel space) :	Yes	Yes
Impact of technology (user space) :	N/A	N/A
Object of surveillance	N/A	N/A
How surveillance is conducted	N/A	N/A
Type of data scanned	N/A	N/A
Type of analysis on the data scanned	N/A	N/A
Rules of surveillance	Separation of process	Separation of operating system
Actions of the technology	Isolate process	Isolate operating system
Impact on performances	Low	Medium
Constraints : utilization, administration	Depend on distribution, need to have a kernel matching the patch used	Depend on distribution

Major limitations : functionality, efficiency, effectiveness, system coverage	Only Linux operating system may be used	Need a good physical server (depending on the number of operating system used).
Improvements	N/A	N/A
Usage in virtualization	N/A	N/A
Configuration options	N/A	N/A
System recovery		
Recovery techniques	N/A	N/A
Impact on performances	N/A	N/A
System reports		
Type of reports	N/A	N/A
Classification of attacks	N/A	N/A
Manage alarms	N/A	N/A
Notifications	N/A	N/A

6.17. Tracing the execution of programs

Description of technologies	Follow the action of the various process		
Name of technology :	LTTNg	Strace	SystemTap
Main features :	Its helps tracking down performance issues and debugging problems involving multiple concurrent processes and threads	Monitor the system calls used by a program and all the signals it receives	SystemTap provides infrastructure to simplify the gathering of information about a running Linux kernel
Features :	Monitor operating system operations	using strace may reveal that the program is attempting to access a file which does not exist or cannot be read.	Systemtap assists in identifying the underlying cause of a performance or functional problem
Scalability :	Yes	Yes	Yes
Installation of component :	Easy	Easy	Easy
Remote's usage :	N/A	N/A	N/A
Impact of technology (kernel space) :	Yes	N/A	Yes
Impact of technology (user space) :	Yes	Yes	Yes
Object of surveillance	Monitor operating system operations	Process	Depend on scripts used
How surveillance is conducted	Follow traces	Monitor system calls	User defined (script)
Type of data scanned	Operating system operations	Systems call	User defined (script)
Type of analysis on the data scanned	Actions done by the system	Actions done	User defined (script)
Rules of surveillance	N/A	N/A	User defined (script)
Actions of the technology	Trace data	Monitor access	Depend on scripts used
Impact on performances	Medium – High	Low	Medium – High

Constraints : utilization, administration	Installation and analyse	N/A	Need to develop scripts
Major limitations : functionality, efficiency, effectiveness, system coverage	May be hard to understand	May be hard to understand	Scripts must be carefully designed
Improvements	N/A	N/A	N/A
Usage in virtualization	Yes	Yes	Yes
Configuration options	N/A	N/A	N/A
System recovery			
Recovery techniques	N/A	N/A	N/A
Impact on performances	N/A	N/A	N/A
System reports			
Type of reports	Traces	Actions done	Actions done
Classification of attacks	N/A	N/A	N/A
Manage alarms	N/A	N/A	N/A
Notifications	N/A	N/A	N/A

6.18. Encrypt the data

Description of technologies	Encryption of the data in order to deny reading by intruders
Name of technology :	TrueCrypt
Main features :	- Creates a virtual encrypted disk within a file and mounts it as a real disk. - Encrypts an entire partition or storage device such as USB flash drive or hard drive. - Encrypts a partition or drive where Windows is installed (pre-boot authentication). - Encryption is automatic, real-time (on-the-fly) and transparent. -Provides plausible deniability
Features :	Secure encryption of data
Scalability :	Yes
Installation of component :	Easy
Remote's usage :	N/A
Impact of technology (kernel space) :	N/A
Impact of technology (user space) :	Low
Object of surveillance	Data
How surveillance is conducted	Encryption
Type of data scanned	N/A
Type of analysis on the data scanned	N/A
Rules of surveillance	N/A
Scalability :	Secure encryption
Impact on performances	Low
Constraints : utilization, administration	Depend on utilization
Major limitations : functionality, efficiency, effectiveness, system coverage	N/A

Improvements	N/A
Usage in virtualization	Yes
Configuration options	N/A
System recovery	
Recovery techniques	N/A
Impact on performances	N/A
System reports	
Type of reports	N/A
Classification of attacks	N/A
Manage alarms	N/A
Notifications	N/A

6.19. Managing the connection to the system

Description of technologies	Managing the connection to the system	
Name of technology :	Knockd	Opie-server
Main features :	knockd is a port-knock server. It listens to all traffic on an ethernet (or PPP) interface, looking for special "knock" sequences of port-hits	OPIE is a free implementation of the S/KEY.
Features :	Depending of the « knock », it open network access	The idea is that each password is only usable once so it doesn’t matter if anyone grabs it as it’ll be useless when they try to use it.
Scalability :	Yes	Yes
Installation of component :	Easy	Easy
Remote's usage :	Yes	Yes
Impact of technology (kernel space) :	N/A	N/A
Impact of technology (user space) :	Yes	Yes
Object of surveillance	Network ports	Password
How surveillance is conducted	Monitor network access	Monitor password used
Type of data scanned	Network datagrams	Password
Type of analysis on the data scanned	Must match rules	Use differents passwords
Rules of surveillance	Used defined	User defined
Actions of the technology	Open network ports	Allow access
Impact on performances	N/A	N/A
Constraints : utilization, administration	Use a network client	Need to generate list of password

Major limitations : functionality, efficiency, effectiveness, system coverage	Need to fully design firewall	N/A
Improvements	N/A	N/A
Usage in virtualization	Yes	Yes
Configuration options	N/A	N/A
System recovery		
Recovery techniques	N/A	N/A
Impact on performances	N/A	N/A
System reports		
Type of reports	N/A	N/A
Classification of attacks	N/A	N/A
Manage alarms	N/A	N/A
Notifications	Yes (system log)	Yes (system log)

7. Conclusion

In this study, we tried to remain as objective as possible while describing the technological solutions that can protect the various points of a computer system. The approach used to present data has the advantage to reveal a logical split between different parts and at the same time to propose a comprehensive approach.

If one were to build a solution based on some of the elements proposed, here is that we would use (level by level):

- Network Firewalls: “Netfilter” because it is the basic tool provided with Linux and has proved its worth;
- Intrusion Detection: “Snort” without hesitation and taking out a subscription in order to be as proactive as possible;
- Network Monitoring: “Nagios” and “Munin” as they both play a different but complementary role;
- Network and application reporting: “Lire” as it can handle many formats;
- Network analysis: “Tcpdump” and “Wireshark” again because the two meet different needs;
- Network Audit: either “Argus” or "Flow-tools", both are valuable in order to present what is happening on the network. Even if it is quite possible to work from the command line with the latter, it will be more effective to take "FlowView" which will present the data via a web server;
- Application Firewall: “Mod_security” and “Greensql” are complementary in their handling of the issue of web security;
- Intrusion detection system: The tool “OSSEC” is now the most widely open source tool used in that role;
- Application monitoring: “Nagios” and “Splunk” (possibly “OSSIM”). The first excels in monitoring systems and services. The second is a tool to centralize log files. The third is it just a great tool but its functional richness is matched only by its complexity to master;
- Application analysis: “Nessus” and “Metasploit” to look for flaws in software systems;

- Hardened operating system: Either “AppArmor” or “SELinux”, both solutions are valuable supplements to add an additional layer of protection;
- Virtualization: “OpenVZ” or “KVM” depending on the type or virtualization needed. The first one is a only a Linux container, and the second one can virtualize different operating systems;
- Data Encryption: “TrueCrypt” because it is currently the most successful.
- Remote Usage: Using a “OTP” to open network ports selectively and have unique passwords.

8. Appendix A - list of all products identified

Acronyms:

Scope	Usage	Goal
loc – locale org – organisation net – network	foren – forensics oper – operations R & D – Research and development	remov – removal forec – forecasting prev – previsional mon – monitoring dis – display detec – detection harden – hardening

Scope	Usage	Goal	Notes	URL	Type of project
loc	foren	forec		distributions Linux (Ubuntu)	open
loc	foren	forec	Wireshark is the world's foremost network protocol analyzer, and is the de facto (and often de jure) standard across many industries and educational institutions.	http://www.wireshark.org/	open
org	oper	foren	With Splunk you can search, report, monitor and analyze real-time streaming and historical IT data generated by all your IT systems from one place.	http://www.splunk.com/	cots
loc	R&D	foren	*Ngrep is a pcap-aware tool that will allow you to specify extended regular expressions to match against data part of packets on the network.*	ngrep.sourceforge.net	open
loc	R&D	foren	tcpdump is a common packet analyzer that runs under the command line. It allows the user to intercept and display TCP/IP and other packets being transmitted or received over a network to which the computer is attached	*www.tcpdump.org*	open
org	R&D	forec	*SEC is an open source and platform independent event correlation tool*	*simple-evcorr.sourceforge.net/*	open
org	oper	mon	ntop is a network traffic probe that shows the network usage	www.ntop.org	open
loc	oper	remov	Clam AntiVirus is an open source (GPL) anti-virus toolkit for UNIX, designed especially for e-mail scanning on mail gateways.	www.clamav.net	open
loc	foren	forec	chkrootkit is a tool to locally check for signs of a rootkit.	http://www.chkrootkit.org/	open

net	foren	forec	Argus is a Real Time Flow Monitor that is designed to perform comprehensive data network traffic auditing.	http://www.qosient.com	open
net	foren	forec	p0f is a versatile passive OS fingerprinting and masquerade detection utility, to be used for evidence or information gathering on servers, firewalls, IDSes, and honeypots	http://freshmeat.net/projects/p0f/	open
loc	oper	mon	*Arpwatch keeps track for ethernet/ip address pairings. It syslogs activity and reports certain changes via email*	ftp://ftp.ee.lbl.gov/arpwatch.tar.gz	open
net	oper	forec	"mon" is a tool for monitoring the availability of services, and sending alerts on prescribed events.	https://mon.wiki.kernel.org	open
net	oper	dis	NfSen is a graphical web based front end for the nfdump netflow tools.	nfsen.sourceforge.net	open
loc	R&D	foren	Rootkit scanner is scanning tool to ensure you for about 99.9%* you're clean of nasty tools. This tool scans for rootkits, backdoors and local exploits	http://www.rootkit.nl/projects/rootkit_hunter.html	open
net	R&D	prev	dsniff is a collection of tools for network auditing and penetration testing	*monkey.org/~dugsong/dsniff*	open
net	R&D	forec	*Flow-tools is a software package for collecting and processing NetFlow data*	*www.splintered.net/sw/flow-tools/*	open
net	R&D	forec	*a collection of netflow tools developed by the CERT/NetSA (Network Situational Awareness) Team to facilitate security analysis.*	*tools.netsa.cert.org/silk/*	open
app	oper	prev	GreenSQL is an Open Source database firewall used to protect databases from SQL injection attacks.	http://www.greensql.net/	open
app	oper	prev	ModSecurity is a web application firewall that can work either embedded or as a reverse proxy. It provides protection from a range of attacks against web applications and allows for HTTP traffic monitoring, logging and real-time analysis.	http://www.modsecurity.org/	open
net	oper	prev	packet filtering framework inside the Linux 2.4.x and 2.6.x kernel series.	http://www.netfilter.org/	open
net	oper	prev	NuFW is an application which adds identity-basedfiltering to Netfilter	http://www.nufw.org/	open
net	R&D	prev	The ebtables program is a filtering tool for a Linux-based bridging firewall. It enables transparent filtering of network traffic passing through a Linux bridge.	http://ebtables.sourceforge.net/	open
loc	oper	prev	AppArmor is a Mandatory Access Control (MAC) system which is a kernel (LSM) enhancement to confine programs to a limited set of resources	distributions Linux (Ubuntu)	open
loc	oper	forec	Fail2ban scans log files like /var/log/pwdfail or /var/log/apache/error_log and bans IP that makes too many password failures. It updates firewall rules to reject the IP address.	*www.fail2ban.org/*	open
loc	oper	prev	Security-Enhanced Linux (SELinux) is a Linux feature that provides a mechanism for supporting access control security policies, including U.S. Department of Defense style mandatory access controls, through the use of Linux Security Modules (LSM) in the Linux kernel.	http://www.nsa.gov/research/selinux/	open
org	oper	prev	Puppet is an open source data center automation and configuration management framework. Puppet provides system administrators with a simplified platform that allows for consistent, transparent, and flexible systems management.	*www.puppetlabs.com/*	open
org	oper	prev	Cfengine is an automation framework for system administration or IT Management.	http://www.cfengine.org/	50/50

org	oper	prev	Bcfg2 helps system administrators produce a consistent, reproducible, and verifiable description of their environment, and offers visualization and reporting tools to aid in day-to-day administrative tasks	http://trac.mcs.anl.gov/projects/bcfg2	open
loc	oper	forec	AIDE (Advanced Intrusion Detection Environment) creates a database for checking the integrity of the files.	http://www.cs.tut.fi/~rammer/aide.html	open
loc	oper	foren	OSSEC is an Open Source Host-based Intrusion Detection System. It performs log analysis, file integrity checking, policy monitoring, rootkit detection, real-time alerting and active response.	http://www.ossec.net/	open
loc	oper	forec	Tripwire® Enterprise provides IT configuration control by combining real-time change detection, comprehensive configuration auditing, continuous compliance assessment, and rapid configuration remediation in a single product suite.	http://www.tripwire.com/	cots
loc	oper	forec	Tripwire® Log Center delivers next-generation log and security event management without the complexity of traditional SIEM systems. The result is continuous compliance and non-stop security that reduces the breach-to-detection gap from the industry average of months, to minutes.	http://www.tripwire.com/	cots
net	R&D	forec	Honeyd is a small daemon that creates virtual hosts on a network. The hosts can be configured to run arbitrary services, and their personality can be adapted so that they appear to be running certain operating systems. Honeyd improves cyber security by providing mechanisms for threat detection and assessment. It also deters adversaries by hiding real systems in the middle of virtual systems.	http://www.honeyd.org/	open
net	R&D	forec	Nepenthes is a versatile tool to collect malware. It acts passively by emulating known vulnerabilities and downloading malware trying to exploit these vulnerabilities.	http://nepenthes.carnivore.it/	open
net	oper	detec	Snort® is an open source network intrusion prevention and detection system (IDS/IPS) developed by Sourcefire. Combining the benefits of signature, protocol and anomaly-based inspection, Snort is the most widely deployed IDS/IPS technology worldwide.	http://www.snort.org	50/50
net	R&D	forec	Bro is an open-source, Unix-based Network Intrusion Detection System (NIDS) that passively monitors network traffic and looks for suspicious activity.	http://www-ids.org	open
net	R&D	forec	The Suricata Engine is an Open Source Next Generation Intrusion Detection and Prevention Engine. This engine is not intended to just replace or emulate the existing tools in the industry, but will bring new ideas and technologies to the field.	http://www.openinfosecfoundation.org/	open
app	oper	forec	PHPIDS (PHP-Intrusion Detection System) is a simple to use, well structured, fast and state-of-the-art security layer for your PHP based web application.	http://php-ids.org/	open
net	oper	forec	New -> suricata		
org	oper	prev	BASE is the Basic Analysis and Security Engine. It is based on the code from the Analysis Console for Intrusion Databases (ACID) project. This application provides a web front-end to query and analyze the alerts coming from a SNORT IDS system.	http://base.secureideas.net/	open

org	oper	prev	Sguil (pronounced sgweel) is built by network security analysts for network security analysts. Sguil's main component is an intuitive GUI that provides access to realtime events, session data, and raw packet captures. Sguil facilitates the practice of Network Security Monitoring and event driven analysis.	http://sguil.sourceforge.net/	open
org	oper	forec	OSSIM stands for Open Source Security Information Management. Its goal is to provide a comprehensive compilation of tools which, when working together, grant network/security administrators with a detailed view over each and every aspect of his or her networks, hosts, physical access devices, server, etc.	http://www.alienvault.com/community.php?section=Home	open
loc	foren	prev	John the Ripper is a fast password cracker. Its primary purpose is to detect weak Unix passwords.	http://www.openwall.com/john/	open
loc, net	R&D	prev	Metasploit provides useful information and tools for penetration testers, security researchers, and IDS signature developers.	http://www.metasploit.com/	open
net	foren	forec	Ettercap is a suite for man in the middle attacks on LAN. It features sniffing of live connections, content filtering on the fly and many other interesting tricks.It supports active and passive dissection of many protocols (even ciphered ones) and includes many feature for network and host analysis.	http://ettercap.sourceforge.net/	open
net, app	oper	forec	The Nessus® vulnerability scanner is the world-leader in active scanners, featuring high-speed discovery, configuration auditing, asset profiling, sensitive data discovery and vulnerability analysis of your security posture.	http://www.nessus.org/nessus/	50/50
net, app	oper	forec	The Open Vulnerability Assessment System (OpenVAS) is a framework of several services and tools offering a comprehensive and powerful vulnerability scanning and vulnerability management solution.	http://www.openvas.org/	open
net	foren	forec	EtherApe is a graphical network monitor for Unix modeled after etherman. Featuring link layer, ip and TCP modes, it displays network activity graphically. Hosts and links change in size with traffic.	http://etherape.sourceforge.net/	open
loc	R&D	foren	Hydra is a parallized login cracker which supports numerous protocols to attack. New modules.Number one of the biggest security holes are passwords, as every password security study shows.	http://freeworld.thc.org/thc-hydra/	open
net	oper	forec	Kismet is an 802.11 layer2 wireless network detector, sniffer, and intrusion detection system.	http://www.kismetwireless.net/	open
loc	R&D	forec	strace is a useful diagnostic, instructional, and debugging tool. System administrators, diagnosticians and trouble-shooters will find it invaluable for solving problems with programs for which the source is not readily available since they do not need to be recompiled in order to trace them	distributions Linux (Ubuntu)	open
loc	R&D	forec	ftrace is a small utility that uses the frysk engine to trace systemcalls in a similar manner to strace.	distributions Linux (Ubuntu)	open
loc	R&D	forec	GDB, the GNU Project debugger, allows you to see what is going on `inside' another program while it executes -- or what another program was doing at the moment it crashed.	http://www.gnu.org/software/gdb/	open

loc	R&D	forec	Kprobes enables you to dynamically break into any kernel routine and collect debugging and performance information non-disruptively. You can trap at almost any kernel code address, specifying a handler routine to be invoked when the breakpoint is hit.	distributions Linux (Ubuntu)	open
loc	R&D	forec	ltrace is a program that simply runs the specified command until it exits. It intercepts and records the dynamic library calls which are called by the executed process and the signals which are received by that process. It can also intercept and print the system calls executed by the program.	distributions Linux (Ubuntu)	open
loc	R&D	forec	The LTTng project aims at providing highly efficient tracing tools for Linux. Its tracers help tracking down performance issues and debugging problems involving multiple concurrent processes and threads. Tracing across multiple systems is also possible.	http://lttng.org/	open
loc	R&D	forec	The Linux Performance Counter subsystem provides rich abstractions over these hardware capabilities. It provides per task, per CPU and per-workload counters, counter groups, and it provides sampling capabilities on top of those - and more.	https://perf.wiki.kernel.org/index.php/Main_Page	open
loc	oper	forec	auditd is the userspace component to the Linux Auditing System. It's responsible for writing audit records to the disk, it is used to see who changed a file in Linux (example).	http://people.redhat.com/sgrubb/audit/	open
loc	oper	prev	The Samhain host-based intrusion detection system (HIDS) provides file integrity checking and log file monitoring/analysis, as well as rootkit detection, port monitoring, detection of rogue SUID executables, and hidden processes.	http://la-samhna.de/samhain/	open
loc	oper	prev	Security Analyst Network Connection Profiler	http://www.metre.net/sancp.html	open
org	oper	prev	Prelude is a Universal "Security Information Management" (SIM) system. Prelude collects, normalizes, sorts, aggregates, correlates and reports all security-related events independently of the product brand or license giving rise to such events; Prelude is "agentless". As well as being capable of recovering any type of log (system logs, syslog, flat files, etc.), Prelude benefits from a native support with a number of systems dedicated to enriching information even further (snort, samhain, ossec, auditd, etc.).	http://www.prelude-technologies.com/fr/bienvenue/index.html	50/50
loc	oper	detec	psad is a collection of three lightweight system daemons (two main daemons and one helper daemon) that run on Linux machines and analyze iptables log messages to detect port scans and other suspicious traffic. A typical deployment is to run psad on the iptables firewall where it has the fastest access to log data.	http://cipherdyne.org/psad/	open
loc	oper	forec	a collection of performance monitoring tools for Linux. These include sar, sadf, mpstat, iostat, pidstat and sa tools	http://sebastien.godard.pagesperso-orange.fr/	open
org	oper	prev	Logcheck is a simple utility which is designed to allow a system administrator to view the logfiles which are produced upon hosts under their control.	http://logcheck.org/	open
org	oper	prev	Logwatch is a customizable log analysis system. Logwatch parses through your system's logs and creates a report analyzing areas that you specify. Logwatch is easy to use and will work right out of the package on most systems.	http://sourceforge.net/projects/logwatch/files/	open
org	oper	prev	Swatch is designed to watch system logs for particular strings and can react on them. Due to this swatch is able to protect SSHD from a brute force attack.	distributions Linux (Ubuntu)	open

loc	R&D	harden	(LIDS) is a patch to the Linux kernel and associated administrative tools that enhances the kernel's security by implementing Mandatory Access Control (MAC). When LIDS is in effect, chosen file access, all system network administration operations, any capability use, raw device, memory, and I/O access can be made impossible even for root.	http://www.lids.org/	open
loc	R&D	harden	Systrace is a computer security utility which limits an application's access to the system by enforcing access policies for system calls. This can mitigate the effects of buffer overflows and other security vulnerabilities.	http://www.citi.umich.edu/u/provos/systrace/	open
org	oper	prev	EJBCA is an enterprise class PKI Certificate Authority built on J2EE technology. It is a robust, high performance, platform independent, flexible, and component based CA to be used stand-alone or integrated in other J2EE applications.	http://ejbca.sourceforge.net/	open
loc	R&D	forec	DTrace is a comprehensive dynamic tracing framework created by Sun Microsystems for troubleshooting kernel and application problems on production systems in real time.		open
loc	R&D	forec	SystemTap provides free software (GPL) infrastructure to simplify the gathering of information about the running Linux system. This assists diagnosis of a performance or functional problem. SystemTap eliminates the need for the developer to go through the tedious and disruptive instrument, recompile, install, and reboot sequence that may be otherwise required to collect data. SystemTap provides a simple command line interface and scripting language for writing instrumentation for a live running kernel. We are publishing samples, as well as enlarging the internal "tapset" script library to aid reuse and abstraction.	http://sourceware.org/systemtap/	open
loc	R&D	forec	Ronin is a Ruby platform for exploit development and security research. Ronin allows for the rapid development and distribution of code, exploits or payloads over many common Source-Code-Management (SCM) systems.	http://ronin-ruby.github.com/	open
loc	R&D	forec	"Utilizes tools such as OpenVAS, Metasploit, nmap, nikto, smbclient, nbtscan, traceroute, Microsoft Baseline Security Analyzer, and other open source tools to gather as much information about a single host or an entire network (limited to a subnet) as possible. PDF reports are generated and scan archives can be sent in an email at the end of the scan to an IT manager or whomever.	http://code.google.com/p/od-autoassess/	open
loc	oper	prev	As any good system administrator knows, there's a lot more to keep track of in an active network than just webservers. Lire is hands down the most versatile log analysis software available today. Lire not only keeps you informed about your HTTP, FTP, and mail traffic, it also reports on your firewalls, your print servers, and your DNS activity. The ever growing list of Lire-supported services clearly outstrips any other software, in large part thanks to the numerous volunteers who have pioneered many new services and features. Lire is a total solution for your log analysis needs.	http://www.logreport.org/	open
org	oper	prev	Logsurfer is a program for monitoring system logs in real-time, and reporting on the occurrence of events. It is similar to the well-known swatch program on which it is based, but offers a number of advanced features which swatch does not support.	http://www.crypt.gen.nz/logsurfer/	open

org	R&D	forec	LogHound is a tool that was designed for finding frequent patterns from event log data sets with the help of a breadth-first frequent itemset mining algorithm. LogHound can be employed for mining frequent line patterns from raw event logs	http://ristov.users.sourceforge.net/loghound/	open
org	R&D	forec	SLCT is a tool that was designed to find clusters in logfile(s), so that each cluster corresponds to a certain line pattern that occurs frequently enough.	http://ristov.users.sourceforge.net/slct/	open
org	R&D	forec	Logpp is a tool for preprocessing event logs and feeding relevant information to other programs for storing or in-depth analysis. During its work, logpp reads lines appended to input files (like tail(1) in -f mode), matches the lines with patterns (e.g., regular expressions), converts matching lines according to given templates, and writes the results to given destinations. Logpp supports multi-line matching and several types of output destinations like regular files, FIFOs, external programs, and the system logger. Therefore, logpp can act as a filter in front of the more complex event log analysis system and increase the system's performance by weeding out irrelevant log data;	http://logpp.sourceforge.net/	open
org	foren	forec	a log file parser that produces a body file used to create timelines (for forensic investigations). Log2timeline takes a log file (or a directory) and parses it to produce a body file that can be imported into other tools for timeline analysis.	http://log2timeline.net	open
loc	oper	forec	LogZilla is a frontend for viewing syslog-ng messages logged to MySQL in realtime. It features customized searches based on device, priority, date, time, and message as well as reporting/graphing of event data.	http://nms.gdd.net/index.php/LogZilla	open
org	oper	forec	LogAnalyzer is a web interface to syslog and other network event data. It provides easy browsing, analysis of realtime network events and reporting services.	http://loganalyzer.adiscon.com/	open
loc	oper	prev	GnuPG allows to encrypt and sign your data and communication, features a versatile key managment system as well as access modules for all kind of public key directories.	http://www.gnupg.org/	open
loc	R&D	forec	Tcpreplay gives you the ability to use previously captured traffic in libpcap format to test a variety of network devices. It allows you to classify traffic as client or server, rewrite Layer 2, 3 and 4 headers and finally replay the traffic back onto the network and through other devices such as switches, routers, firewalls, NIDS and IPS's. Tcpreplay supports both single and dual NIC modes for testing both sniffing and inline devices.	http://tcpreplay.synfin.net/	open
org	oper	forec	Cacti is a complete network graphing solution designed to harness the power of RRDTool's data storage and graphing functionality. Cacti provides a fast poller, advanced graph templating, multiple data acquisition methods	http://www.cacti.net/	open
org	oper	forec	FlowScan analyzes and reports on Internet Protocol (IP) flow data exported by routers. Consisting of Perl scripts and modules, FlowScan binds together (1) a flow collection engine (a patched version of cflowd), (2) a high performance database (Round Robin Database - RRD), and (3) a visualization tool (RRDtool). FlowScan produces graph images that provide a continuous, near real-time view of the network border traffic.	http://www.caida.org/tools/utilities/flowscan/pub/	open
loc	R&D	forec	Ourmon is a statistically oriented open-source network monitoring and anomaly detection system.	http://ourmon.sourceforge.net/	open
loc	R&D	forec	IPAudit monitors network activity on a network by host, protocol and port.	http://ipaudit.sourceforge.net/	open

loc	R&D	forec	IPAudit listens to a network device in promiscuous mode, and records every connection between two ip addresses. A unique connection is determined by the ip addresses of the two machines, the protocol used between them, and the port numbers (if they are communicating via udp or tcp).	http://thnetos.wordpress.com/nsm-console/	open
loc	R&D	forec	Tstat is a passive sniffer able to provide several insight on the traffic patterns at both the the network and transport levels.	http://tstat.tlc.polito.it/index.shtml	open
org	oper	mon	RANCID monitors a router's (or more generally a device's) configuration, including software and hardware (cards, serial numbers, etc) and uses CVS (Concurrent Version System) or Subversion to maintain history of changes. It sens an email when difference is noted	http://www.shrubbery.net/rancid/	open
loc	oper	prev	FreeIPA is an integrated security information management solution combining Linux (Fedora), Fedora Directory Server, MIT Kerberos, NTP, DNS, Dogtag (Certificate System). It consists of a web interface and command-line administration tools.	http://freeipa.org/page/About	open
loc	oper	forec	Webfwlog is a flexible web-based firewall log analyzer and reporting tool. It supports standard system logs for linux, FreeBSD, OpenBSD, NetBSD, Solaris, Irix, OS X, etc. as well as Windows XP®. Supported log file formats are netfilter, ipfilter, ipfw, ipchains and Windows XP®. Webfwlog also supports logs saved in a database using the ULOGD target of the linux netfilter project.	http://devel.webfwlog.net/index.php	
loc	R&D	forec	Kojoney is a low level interaction honeypot that emulates an SSH server	http://kojoney.sourceforge.net/	
org	R&D	forec	SCAP is a line of standards managed by NIST (http://scap.nist.gov/). It was created to provide a standardized approach to maintaining the security of enterprise systems, such as automatically verifying the presence of patches, checking system security configuration settings, and examining systems for signs of compromise.	http://www.open-scap.org/page/Main_Page	
loc	oper	prev	The Shibboleth System is a standards based, open source software package for web single sign-on across or within organizational boundaries. It allows sites to make informed authorization decisions for individual access of protected online resources in a privacy-preserving manner.	http://shibboleth.internet2.edu/	
loc	oper	prev	Squid is a caching proxy for the Web supporting HTTP, HTTPS, FTP, and more. It reduces bandwidth and improves response times by caching and reusing frequently-requested web pages	http://www.squid-cache.org/	
loc	oper	prev	DeleGate is a multi-purpose proxy server for multiple application protocols running on multiple platforms	http://www.delegate.org	
loc	R&D	forec	Aanval is the industry's most comprehensive Snort & Syslog intrusion detection and correlation console designed specifically to scale from small single sensor installations to global enterprise deployments.	http://www.aanval.com/flex/#op=main	

loc	oper	forec	TrueCrypt is a software system for establishing and maintaining an on-the-fly-encrypted volume (data storage device). On-the-fly encryption means that data is automatically encrypted or decrypted right before it is loaded or saved, without any user intervention. No data stored on an encrypted volume can be read (decrypted) without using the correct password/keyfile(s) or correct encryption keys. Entire file system is encrypted (e.g., file names, folder names, contents of every file, free space, meta data, etc).	http://www.truecrypt.org/	
loc	oper	forec	EncFS provides an encrypted filesystem in user-space. It runs without any special permissions and uses the FUSE library and Linux kernel module to provide the filesystem interface. You can find links to source and binary releases below. EncFS is open source software, licensed under the GPL.	http://www.arg0.net/encfs	
org	oper	forec	As with most encrypted filesystems, Encfs is meant to provide security against off-line attacks; ie your notebook or backups fall into the wrong hands, etc. The way Encfs works is different from the “loopback” encrypted filesystem support built into the Linux kernel because it works on files at a time, not an entire block device. This is a big advantage in some ways, but does not come without a cost.	http://snorby.org/	
loc	oper	forec	grsecurity is an innovative approach to security utilizing a multi-layered detection, prevention, and containment model.	http://grsecurity.net/	
loc	oper	forec	RSBAC is a flexible, powerful and fast (low overhead) open source access control framework for current Linux kernels, which has been in stable production use since January 2000 (version 1.0.9a). All development is independent of governments and big companies, and no existing access control code has been reused.	http://www.rsbac.org/	
loc	oper	forec	Practically, it allows full fine grained control over objects (files, processes, users, devices, etc.), memory execution prevention (PaX, NX), real time integrated virus detection, and much more.	http://fossology.org/	
org	oper	forec	Vsam (Vulnerability, Scanning, Analysis and Management) is a open source virtual appliance developed and distributed in an effort to bring to the security community a tool to provide full management capabilities to Nessus scan data. Based on the great work of the Inprotect project, Vsam extends the ability of Inprotect by bringing the power of virtualization to this highly functional project. Distributed as a "black box" appliance, no knowledge of Lunix or any underlying software is required. Once downloaded and setup in a VMware environement, Vsam can be ready to use in minutes.	http://vsam.sourceforge.net/	
net	oper	prev	L7-filter is a packet classifier for Linux. Unlike most other classifiers, it doesn't just look at simple values such as port numbers. Instead, it does regular expression matching on the application layer data to determine what protocols are being used.	http://l7-filter.sourceforge.net/	
loc	oper	forec	Afick is a security tool, very close from the well known tripwire. It allows to monitor the changes on your files systems, and so can detect intrusions.	http://afick.sourceforge.net/	
org	oper	prev	The Snare Server, from InterSect Alliance, is a proprietary Log Monitoring solution that builds on the open source Snare agents to provide a central audit event collection, analysis, reporting and archival system.	http://www.intersectalliance.com/snareserver/index.html	

net	oper	prev	knockd is a port-knock server. It listens to all traffic on an ethernet (or PPP) interface, looking for special "knock" sequences of port-hits. A client makes these port-hits by sending a TCP (or UDP) packet to a port on the server. This port need not be open -- since knockd listens at the link-layer level, it sees all traffic even if it's destined for a closed port. When the server detects a specific sequence of port-hits, it runs a command defined in its configuration file. This can be used to open up holes in a firewall for quick access.	http://www.zeroflux.org/projects/knock	
org	oper	prev	SecVisor, a tiny hypervisor designed to ensure that only approved kernel code is executable. SecVisor provides lifetime kernel protection regardless of the scale of an attack and the extent to which system control is compromised.	http://www.cylab.cmu.edu/partners/success-stories/SecVisor.html	
org	oper	prev	sHype is a hypervisor security architecture – it provides a strong isolation, mediated sharing and communication between Virtual Machines. It controls resource control and accurate accounting guarantees.	http://www.research.ibm.com/secure_systems_department/projects/hypervisor/	
org	oper	forec	Linux-VServer provides virtualization for GNU/Linux systems. This is accomplished by kernel level isolation. It allows to run multiple virtual units at once. Those units are sufficiently isolated to guarantee the required security, but utilize available resources efficiently, as they run on the same kernel.	http://linux-vserver.org/Welcome_to_Linux-VServer.org	
org	oper	forec	OpenVZ is container-based virtualization for Linux. OpenVZ creates multiple secure, isolated containers (otherwise known as VEs or VPSs) on a single physical server enabling better server utilization and ensuring that applications do not conflict. Each container performs and executes exactly like a stand-alone server; a container can be rebooted independently and have root access, users, IP addresses, memory, processes, files, applications, system libraries and configuration files	http://wiki.openvz.org/Main_Page	
org	oper	forec	Sourcefire’s Razorback Framework features an open source, distributed detection system, robust API set and a fully extensible database and data management system. It has been specifically designed with the needs of high-level incident response and detection teams. Razorback enables you to perform advanced processing of data and detection of events by fetching data as it traverses the network and even fetch data from a server. It is able to perform advanced event correlation since this framework works in a distributed fashion. You can consider it as the open source Snort NIDS front-end.	http://nuggetfarm.sourceforge.net/	
org	oper	prev	OPIE is a free implementation of the S/KEY (one time password) specifications (RFC 1760 and RFC 2289). The idea is that each password is only usable once so it doesn’t matter if anyone grabs it as it’ll be useless when they try to use it.		

Category	WWW site	Event Type	Data Type
Network Filtering			
Ebtables	http://ebtables.sourceforge.net/	informationnel	Syslog + user defined software (nflog + ulog targets)
Netfilter	http://www.netfilter.org/	informationnel	Syslog + user defined software (nflog + ulog targets)
NuFw	http://www.nufw.org/	informationnel	Syslog + user defined software (nflog + ulog targets)
L7-filter	http://l7-filter.sourceforge.net/	informationnel	Syslog + user defined software (nflog + ulog targets)
knockd	http://www.zeroflux.org/projects/knock	informationnel	Syslog + user defined text logfile
Network detection			
Bro	http://bro-ids.org	alertes	text + database
Snort	http://www.snort.org	alertes	text + database
Suricata	http://www.openinfosecfoundation.org/	alertes	Syslog + text
Network Monitoring			
Arpwatch	http://ee.lbl.gov/	alertes	text
Mon	http://mon.wiki.kernel.org/index.php/Main_Page	informationnel	Email, Alphanumeric paging/messaging Windows pop-up messages, Instant messaging
Munin	http://munin-monitoring.org/	informationnel	Email, syslog, external scripts
Nagios	http://www.nagios.org/	informationnel	Email, , external scripts
Opennms	http://www.opennms.org/	informationnel	
Network Reporting			
Lire	http://www.logreport.org/	informationnel	text
Webfwlog	http://devel.webfwlog.net/index.php	informationnel	text
Network Analysis			
Ngrep	http://ngrep.sourceforge.net/	informationnel	network data
Tcpdump	http://www.tcpdump.org/	informationnel	network data
Wireshark	http://www.wireshark.org/	informationnel	network data
Network Audit			
Argus	http://www.qosient.com/argus/	Informationnel + alertes	network informations
Flow-tools	http://www.splintered.net/sw/flow-tools &	Informationnel + alertes	network informations

Nfdump	http://nfdump.sourceforge.net/	informationnel	network informations
Nmap	http://www.insecure.org	informationnel	network informations
Flowviewer - FlowTracker FlowGrapher	http://ensight.eos.nasa.gov/FlowViewer	Informationnel + alertes	network informations
Application Firewall			
Squid	http://www.squid-cache.org	Informationnel + alertes	text logfile
Delegate	http://www.delegate.org	Informationnel + alertes	text logfile
mod_security	http://www.modsecurity.org/	alertes	text logfile
GreenSQL	http://www.greensql.net	alertes	text logfile
Intrusion Detection: HIDS			
Ossec	http://www.ossec.net/	alertes	text logfile
AIDE	http://sourceforge.net/projects/aide/	alertes	text logfile
Tripwire	http://www.tripwire.com/	alertes	text logfile
Sec	http://simple-evcorr.sourceforge.net/	alertes	text logfile
Prelude-ids	http://www.prelude-technologies.com/fr/bienvenue/index.html	alertes	text logfile
Application Monitoring			
Nagios	http://www.nagios.org/	Informationnel + alertes	text logfile + visual alerts
Splunk	http://www.splunk.com/	Informationnel	Text
Ossim	http://www.alienvault.com/	Informationnel + alertes	Text + visual alerts
Application Reporting			
Lire:	http://www.logreport.org/	Informationnel	text logfile
Application Analysis			
Nessus	http://www.nessus.org/	Informationnel + alertes	text logfile + visual alerts
Metasploit	http://www.metasploit.com/	Informationnel + alertes	text logfile + visual alerts
Openvas	http://www.openvas.org/	Informationnel + alertes	text logfile + visual alerts
Spike	http://www.immunitysec.com/resources-freesoftware.shtml	Informationnel + alertes	text
Webscarab	http://www.owasp.org/index.php/OWASP_WebScarab_NG_Project	Informationnel + alertes	text
Fusil	http://bitbucket.org/haypo/fusil/wiki/Home	Informationnel	text
Application Audit			
Lint	http://docs.sun.com/source/806-3567/lint.html	Informationnel + alertes	text

Splint	http://splint.org/	Informationnel + alertes	text
Valgrind	http://valgrind.org/	Informationnel + alertes	text
Hardening			
AppArmor	http://www.novell.com/linux/security/apparmor/	alertes	text
Grsecurity	http://grsecurity.net/	alertes	text
RSBAC	http://www.rsbac.org/	alertes	text
SELinux	http://www.nsa.gov/research/selinux/index.shtml	alertes	text
MAC	http://en.wikipedia.org/wiki/Mandatory_access_control	alertes	text
DTOS	http://www.cs.utah.edu/flux/fluke/html/dtos /HTML/dtos.html	alertes	text
FLASK	http://www.cs.utah.edu/flux/fluke/html/flask.html	alertes	text
Virtualization			
secvisor	http://www.cylab.cmu.edu/partners/success-stories /SecVisor.html	Informationnel	text
sHype	http://www.research.ibm.com /secure_systems_department/projects/hypervisor/	Informationnel	text
sVirt	http://vidéoprojecteur/page/SVirt	Informationnel	text
KvmSec	note : A Security Extension for Linux Kernel Virtual Machines	Informationnel	text
Tracking Data			
strace	http://en.wikipedia.org/wiki/Strace	Informationnel	text
gdb	http://www.gnu.org/software/gdb/	Informationnel	text
ftrace, kprobe, ltrace, strace	Note : included in Linux distributions	Informationnel	text
LTTng	http://lttng.org/	Informationnel	text
perf	http://perf.wiki.kernel.org/index.php/Main_Page	Informationnel	text
systemtap	http://sourceware.org/systemtap/	Informationnel	text