Intrusion detection systems

Joaquin Garcia-Alfaro

Recommended minimum reading time: 4 hours

© Fundació Universitat Oberta de Catalunya (FUOC) Av. Tibidabo,
39-43, 08035 Barcelona
Authorship: Joaquin Garcia-Alfaro
Production: FUOC

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. The terms of the license can be consulted in http://www.gnu.org/licenses/fdl-1.3.html.

Contents

Introduction

The main consequence of the use of any digital technology is the exploitation of its weaknesses and, therefore, the execution of cyber attacks. Computer networks are no exception. Any network connected to the Internet is exposed to possible attacks. This is why it is very important to impose new security requirements beyond protection mechanisms (such as cryptography and firewall systems). Although deficiencies in these systems can be checked using conventional tools, they are not always corrected. In general, these weaknesses can put at risk network security and facilitate illegal entries into the system.

The area of cyber defence is precisely trying to solve this new problem with the use of mechanisms for monitoring and detecting attacks. Most current organisations have data protection mechanisms integrated into their networks. But, although these mechanisms must be considered essential, the security assumed by the organisation must carry on increasing.

Thus, a purely perimeter level of protection (based only on the integration of firewall systems and encryption systems into the network) is not enough. We must think that not all access to the network goes through the firewall or that all threats originate in the external area of the firewall. In addition, firewall systems, like other elements of the network, can also be attacked.

It is essential to install detection mechanisms, capable of alerting the network administrator at the time that these attacks take place, not only for the detection part, but also to remediate the attacks. An analogy that helps to understand the need to incorporate these elements could be the comparison between the security of a computer network and the security of a building: the entrance doors carry out a first level of access control, but we usually do not leave it at that; we will install motion detectors or surveillance cameras at key points in the building to detect the existence of unauthorised people, or people who may misuse resources and endanger security. In addition, there will be security guards, registration books in which it will be necessary to write down all the staff who access a certain department that we consider critical. All this information is processed from a security control office where the recording of the cameras is supervised and the registration books are kept. All these elements, projected into the digital world, shape what is known in the field of cybersecurity as *cyber defence elements*.

Objectives

The objectives you must achieve once you have worked through the materials of this module are the following:

1. To understand the origins of the concept of *intrusion detection* and see some of the classic techniques that can be used.

2. To learn to classify these systems according to various criteria, such as the place where the data analysis process is carried out or the specific detection mechanism.

3. To understand the limitations linked to the traditional processes of an intrusion detection system, including the problem of false positives and false negatives.

4. To know about the existence of additional management tools capable of normalising, merging and matching events and alerts collected in a distributed manner. These tools can consolidate the treatment of incidents detected through different event monitoring architectures.

5. To see a specific example of an intrusion detection system at the network level, using the Snort free open source tool.

6. To see complementary technologies to classical detection systems, with the specific example of deception systems and technologies.

1. Cyber defence and tools for intrusion detection

Cyber defence assumes that an attacker is able to violate the security policy associated with an information system, in order to misuse its resources or disrupt its operation.

The mechanisms associated with the **cyber defence** technologies are responsible for finding and reporting all kinds of types of malicious activity in the system or network, with the aim of reacting and stopping the attack in the most appropriate way.

In general, it is desirable to be able to identify the exact attack that is taking place in order to stop it and to recover from its effects.

In some particular situations, it will only be possible to detect and to report the suspicious activity that has been found, given the impossibility of really knowing what has happened and reporting how to stop it.

Generally, cyber defence mechanisms work under the premise of the *worst-case scenario*, assuming that an attacker has gained access to the system and is able to use or modify its resources.

The most notable elements within the category of cyber defence mechanisms are the Intrusion Detection Systems (**IDS**).

Origins of the term IDS

One of the first cyber defence communities comes from the world of security in database systems.

In these communities, rather than talking about attacks, they talk about intruders inside a database. Most of the references to the term *intruder* or *intrusion* are usually changed for *attacks* by other communities in the world of security (e.g., communities associated with the security of industrial systems).

A second community associated with the origins of the term IDS comes from the world of error detection and system failures. This community is interested in how to automate the detection of situations in which the different components of a system behave abnormally. In the case of an IDS, the aim is not to detect behaviours associated with random errors. On the contrary, the aim is to detect situations with marked intentionality. While errors have a probabilistic behaviour, attacks leave a trail associated with human behaviour, with malicious intent. For this reason, the term *intruder* is also used in these communities, to differentiate these two kinds of situations.

We will now introduce the main definitions that are used in the field of intrusion detection, with the aim of clarifying common terms that we will use later.

An **intrusion** is a sequence of actions taken by a malicious adversary with the ultimate goal of causing unauthorised access to a system or network of systems.

The intrusion is the sequence of steps taken by the intruder.

Each step taken represents a violation of the security policy of the system or the network. Therefore, the existence of a security policy indicating mandatory compliance actions, or with the malicious actions to be prevented, is a key requirement to detect an intrusion. In other words, a violation can only be detected when the actions observed can be compared to the set of rules defined in the security policy.

Intrusion detection is the process of identifying and responding to activities observed against the security policy of a system or network of systems.

This last definition introduces the notion of the intrusion detection process, which involves a whole series of technologies, users and tools needed to achieve success. Next, we will see these technologies in more detail, from their early origins to the current systems.

1.1. Background and current systems

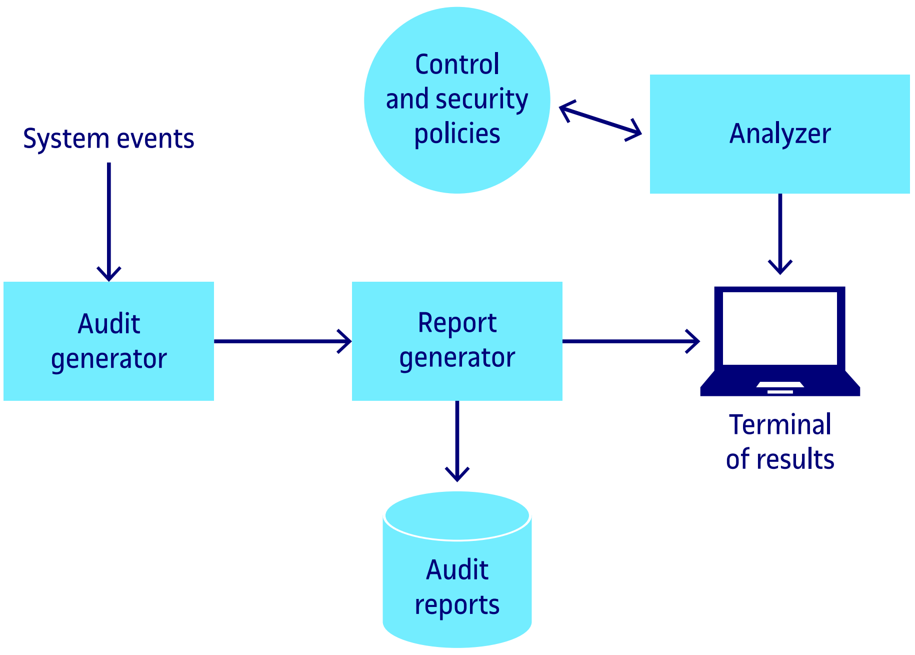
Current intrusion detection systems are a direct evolution of auditing systems. These systems were intended to measure the time that operators spent using the systems they monitored, with millisecond accuracy, and served, among other things, to bill for the service.

The first systems appeared in the 1950s, when the American company Bell Telephone System created a development group with the aim of analysing the use of computers in telephone companies. This team established the need to use audits through electronic data processing, unlike the previous system based on the preparation of paper reports.

This led the Bell System companies to embark on the first large-scale computer-controlled telephone billing system in the late 1950s.

Figure 1 shows a simple diagram of the operation of an audit system, in which the system events are captured by audit generators that will take the data to the element in charge of storing them in a report file.

Figure 1. Operation of a classic audit system



Trusted Computer System Evaluation Criteria (TCSEC)

Series of documents of the NSA (National Security Agency of the United States Department of Defense) on trust systems, also known as the *Rainbowseries*, due to the colours of their covers. The main book in this series is known as *the Orange Book*). For more information, see the website: http://www.fas.org/irp/nsa/rainbow.htm.

Since the 1970s, the U.S. Department of Defense began to invest considerable resources for research into security policies, directives, and control guidelines. These efforts culminated in a security initiative in 1977, in which the concept of *trust systems* was defined.

Trust systems are systems that employ enough software and hardware resources to enable the simultaneous processing of a variety of confidential or classified information. These systems include different types of information distributed into levels, which correspond to their degree of confidentiality.

In the late 1970s, a section on audit mechanisms as a requirement for any trust system with a high level of security was included in the *Trusted Computer System Evaluation Criteria* (TSCSEC). This document, known as *the Tan Book,* lists the main objectives of an audit mechanism that we can summarise very briefly in the following points:

- To allow the review of access patterns (by an object or by a user) and the use of system protection mechanisms.

- To allow the discovery of both internal and external attempts to circumvent protective mechanisms.

- To allow the discovery of user transition when moving from regular to high level privileges (privilege elevation).

- To allow blocking attempts by users to bypass system protection mechanisms.

- To serve, in addition, as a guarantee to users that all the information collected about attacks and intrusions will be sufficient to control the possible damage caused to the system.

1.1.1. Early solutions

The *Intrusion Detection Expert System* (IDES) project, developed between 1984 and 1986 by Dorothy Denning and Peter Neumann, was one of the first real-time intrusion detection systems. This project, funded by the US Navy, proposed a correspondence between anomalous activity and abuse, or misuse (understanding strange or unusual activity in a statistical context as anomalous).

IDES used profiles to describe the subjects of the system (mainly users), and activity rules to define the actions that took place there (system events or CPU cycles). These elements allowed establishing, by statistical methods, the patterns of behaviour needed to detect possible anomalies.

A second real-time intrusion detection system to highlight was Discovery, which was able to detect and prevent security problems in databases. The novelty of the system was in the monitoring of applications rather than analysing an operating system in its entirety. By using statistical methods developed in COBOL, Discovery was able to identify possible abuses.

Other systems were also developed to help American officials find trademarks of internal attacks on the main computers of their airbases. These computers were mainly corporate servers that worked with unclassified but very confidential information.

One of the last important systems of this period was the Multics Intrusion Detection and Alerting System (MIDAS), created by the National Computer Security Center (NCSC). This detection system was implemented to monitor the NCSC Dockmaster system, where one of the most secure operating systems of the time[1] was executed. Like the IDES system, MIDAS used a hybrid system that combined both statistical detection of anomalies and the safety rules of

(1)It is the Multics operating system, precursor of today's Unix systems.

an expert system. MIDAS used a progressive analysis process consisting of four levels of rules. In addition to these rules, it also had a database that it used for determining signs of abnormal behaviour.

MIDAS was one of the first detection systems for Internet-connected intrusions. It was published online in 1989, and it monitored the mainframe Dockmaster in 1990, to strengthen user authentication mechanisms.

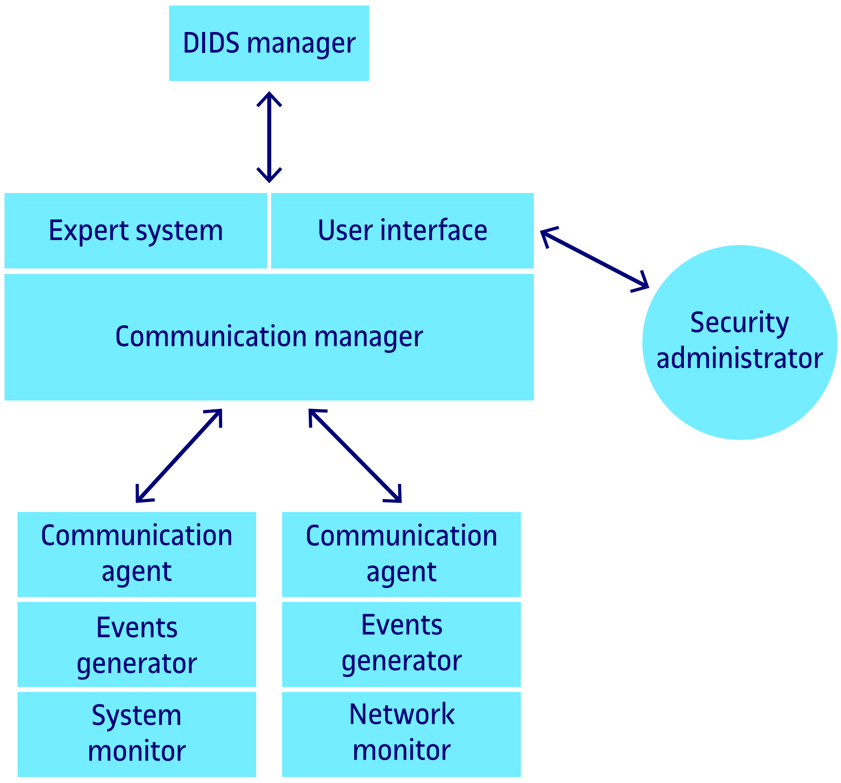
1.1.2. Current intrusion detection systems

Since 1990, the fast growth of computer networks led to the emergence of new intrusion detection models. As well as that, the damage caused by the famous Robert Morris worm (in 1988) helped to unite commercial and academic efforts in the search for security solutions in the field of cyber defence.

The first step was to merge the monitoring processes of the operating system with the monitoring systems of network traffic. This was the case of the Distributed Intrusion Detection System (DIDS), capable of enabling a security group to monitor security breaches through networks connected to the Internet.

The initial goal of DIDS was to provide means to centralise control and publication of results in a central controller. Figure 2 shows the operating diagram of the DIDS system.

Figure 2. Diagram of the operation of the Distributed Intrusion Detection System (DIDS)



At the same time, the first commercial intrusion detection programs began to appear. Some companies developed them to occupy a prominent position in the field of cyber security, although others did them to improve the levels of protection required by national security agencies, such as the NCSC.[2]

(2)It stands for National Computer Security Centre

Currently, a large number of intrusion detection systems to protect operating systems and complete networks are available. Many of these systems are commercial or reserved for military and research environments. There are also a large number of free solutions that can be used without any restrictions.

1.2. General architecture of a detection system

Since the beginning of the 1980s, many studies regarding the construction of intrusion detection systems have been carried out. In all these studies, different proposals and designs have been made in order to meet the following requirements (More information available at: http://dx.doi.org/10.1007/BF02994844):

- **Accuracy.** An intrusion detection system should not confuse legitimate actions with malicious actions at the time of detecting them. When legitimate actions are detected as malicious actions, the detection system may end up causing a denial of service against a legitimate user or system. These types of detections are known as ***false positives***. The smaller the number of false positives, the more precision the intruder detection system will have.

- **Efficiency**. The detector must minimise the undetected malicious activity rate (known as ***false negatives***). The lower the rate of false negatives, the greater the efficiency of the intrusion detection system. This is a complicated requirement, as it can sometimes become impossible to obtain all the necessary knowledge about past, current and future attacks.

- **Performance.** The performance offered by an intrusion detection system must be sufficient to manage real-time detection. Real-time detection must respond to intrusion detection before the attack causes damage to the system. According to studies, this time should be less than one minute.

- **Scalability.** As the network grows (both in size and speed), the number of events to be processed by the detection system will also increase. The detector must be able to withstand this increase in the number of events, without any loss of information. This requirement is of great relevance in distributed attack detection systems, in which events are launched on different equipment of the system and must be matched by the intruder detection system.

- **Fault tolerance.** The intrusion detection system must be able to continue to offer its service even if different elements of the system are attacked (including the situation where the system itself receives an attack or intrusion).

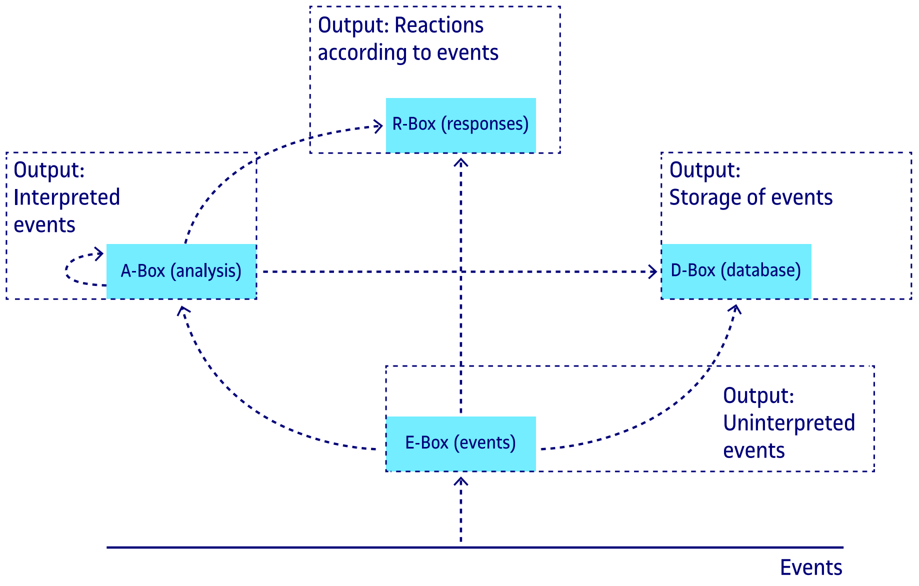
In addition, more recent criteria also include concepts such as knowledge *completeness*, implementation *facility*, maintenance *facility* and *explanation* of algorithms used for detection. These new criteria are considered essential for the future expansion of commercial IDS tools.

In order to normalise the heterogeneity of IDS architectures and tools, and achieve the aforementioned criteria, standardisation communities such as IETF[3] have been working in recent decades on the conception of general-purpose IDS architectures. One of these, the architecture known as CIDF,[4] is shown in Figure 3.

(3)Stands for Internet Engineering Task Force

(4)Stands for Common Intrusion Detection Framework

Figure 3. CIDF architecture (The Common Intrusion Detection Framework)


Output: Reactions according to events
R-Box (responses)
Output: Interpreted events
A-Box (analysis)
Output: Storage of events
D-Box (database)
E-Box (events)
Output: Uninterpreted events
Events

Early IETF working groups, like the well-known IDWG,[5] complement architectures such as the CIDF with the definition of development requirements, like the following:

(5)Stands for Intrusion Detection Working Group

1) The need to formalise the interaction between IDS tools with other security elements, such as prevention mechanisms (firewall systems, access control lists, etc.).

In this area, proposals for cooperation between the different security elements have been worked on through the exchange of messages. These mechanisms must ensure correct communication between the different elements of an IDS architecture and provide additional criteria such as the security of communications, authenticity and integrity of the information exchanged (alerts and events), etc.

2) The need to define the exchanged messages (events, alerts, etc.) among the system's elements. Therefore, the IETF has proposed formats for the exchange of messages between security devices, such as IDMEF[6] and IODEF.[7]

(6)Stands for Intrusion Detection Message Exchange Format, RFC 4765

(7)Stands for Incident Object Description Exchange Format

Recommended link

The current version of the IDMEF and IODEF specifications is currently under review. We recommend seeing the www.secef.net website for more information.

By observing the above standardisation efforts, we can identify the following elements for the construction of an IDS-type general architecture:

1) event collectors

2) event processors

3) response units and remediation plans

We will now see some of these elements in more detail.

1.2.1. Event collectors

The event collectors, also known as **sensors**, are responsible for collecting information from equipment monitored by an IDS. The information collected is transformed into a sequence of events. These sequences will be analysed later by event processors.

The information stored in the events collected by the sensors is the decision base for the IDS detection. It is very important to be able to guarantee its integrity when facing potential modification attacks, when transmitting these events between the sensor that created them and the processing component that will treat them.

There are different ways to classify the possible implementations of these event collectors. The most common proposals are detailed next.

The first type of collectors, also known as ***host-based sensors*** **or system-based sensors**, are in charge of analysing and collecting information about events that have happened at operating system level (such as connection attempts and system calls).

These sensors are also known as ***application-based sensors***, in the case of completing the information collected from the operating system, with events originated from applications.

In the second category we find sensors that collect information from events that have happened at the level of network traffic (for example, analysing the IP headers of datagrams that pass through the network interface). These types of components are known as ***network-based sensors***.

The specific choice of the type of sensor will depend on the purpose of the desired detection. In fact, communities concerned about intrusion detection systems have thoroughly debated which sensors can offer better performance from a detection and effectiveness point of view. An ideal IDS should unify all possible options and offer a hybrid solution with the best of each option. Here are some additional details of each type:

1) Host-based sensors. These sensors are easy to configure, apart from being able to offer accurate information about events and potential attacks.

Snort, Suricata and Zeek

A widely used detection tool as a network-based sensor is Snort.

This tool is a network intrusion detector developed under the free software paradigm, capable of carrying out real-time traffic analysis, as well as logging packets in TCP/IP networks.

Two complementary tools in addition to Snort are Suricata and Zeek (formerly called Bro).

Check the websites www.snort.org, www.suricata.io and www.zeek.org for more information.

The data generated by these sensors can have a high density of information, such as the information reported by the file servers of an operating system log. They may also include a large amount of preprocessing information that later facilitates the work of other data processing components.

A disadvantage is that these sensors can have a significant impact on the efficiency of the system in which they are executed.

2) Network-based sensors. The main advantage of network-based sensors, compared to other solutions, is the possibility of being able to work in a non-intrusive way (i.e., in a passive manner). The collection of information does not affect the way the equipment works or the infrastructure itself. Since they do not necessarily reside in the equipment to be analysed, they are more resistant to attacks.

Moreover, most network-based sensors are also independent of the operating system and can obtain network-level information (such as the existence of fragmentation in IP datagrams) that could not be provided by host-based sensors.

Some network-based sensors are actually switches with an analysis capability that is transparent to the rest of the system.

As the main disadvantage of network-based sensors, it is worth noting the low scalability that this approach offers. In cases of networks with very high traffic loads, these sensors are likely to start losing packets, which means a loss in their ability to collect information.

These sensors would hardly be able to continue working normally on high-speed networks, such as Ethernet networks at speeds higher than a gigabit.

Another problem with network-based sensors is the increase in encrypted communications. The use of cryptography to protect communications makes the information to be collected incomprehensible to the sensors and thus reduces their detection capabilities. Some current solutions to deal with this problem include combining classical cryptography with homomorphic cryptography, to process part of the traffic even though it cannot decipher it.

Installation of event collectors

In addition to the sensor type, it is also important to determine the exact place where these components should be placed (from where to collect the information). The simplest to place are application-based sensors, generally installed in the parts of the program in which debugging and generation of log files services are offered. But the situation is more difficult for the other variants.

When considering the installation of system-based sensors, the wide variety of operating systems there are, and the different facilities offered by each, poses a serious problem. In addition, it is not easy to determine what part of the large amount of information created by the kernel of an operating system should be relevant when analysing.

In the case of Unix-like operating systems, there is the *Orange Book* proposal (which we have already mentioned in this module), which shows some points of interest in which information should be analysed.

In the case of network-level sensors, segmentation through switches is a major drawback in choosing the right location to place these sensors. For example, a star topology causes packets to be directed only between the two parts of a communication, so the sensor should be placed at a point where it could analyse any exchange of information.

A first option would be to place the sensor on the link where all the equipment on the network joins. This option could mean the need to analyse such a high amount of data that the sensor would end up losing information.

The other option would be the placement of the sensor between the network link that separates the interior and the exterior, as if it were an additional perimeter protection system.

A variant of these two options would be the use of the tap port that many switches offer. It is a special port that reflects all the traffic that passes through the equipment. But this can easily overload the subsequent analysis capacity if the amount of traffic is very high. In addition, the internal bandwidth of the device is not always enough to deal with all active ports at once. If the traffic analysed begins to grow, the capacity of the tap port may be exceeded, with the corresponding loss of packets that this would entail.

1.2.2. Event processors

Event processors, also known as **analysers**, form the core of the detection system. They have the responsibility to operate on the information collected by the sensors in order to infer possible intrusions.

In order to infer intrusions, the analysers implement a specific detection scheme. Two of the most commonly used methods for detection are the misuse detection model and the anomaly detection model. We will now briefly comment on these two detection schemes.

Misuse-based detection

The detection of intrusions based on the misuse model has prior knowledge of malicious sequences and activities. Event processors that implement this scheme analyse events for known attack patterns or activity that attacks typical vulnerabilities.

The sequences or patterns described are known as *attack signatures* and could be compared to the virus signatures used by current antiviruses.

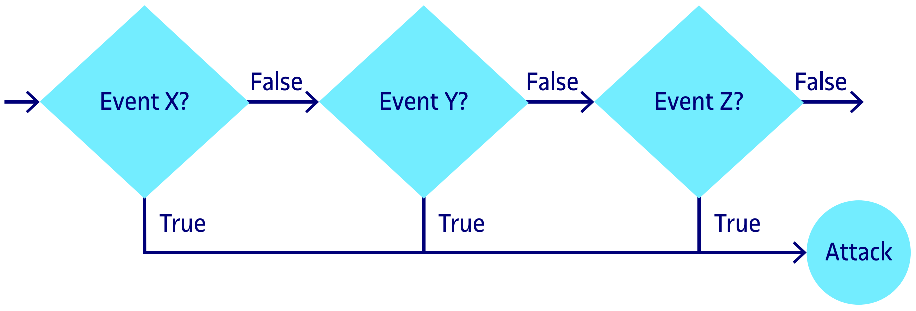
Thus, the detection components based on the misuse model will compare the events sent by the sensors with the attack signatures that they keep stored in their knowledge bases.

When an event or sequence of events matches an attack signature, the analyser will issue an alert.

When implementing a detection scheme based on misuse, two of the models most commonly used are analysers based on pattern recognition and analysers based on state transitions:

1) Pattern recognition analysers. By using *if-then-else* rules to examine data, these analysers process the information through internal functions in the system, in a completely transparent way to the user. Figure 4 shows the diagram of an *if-then-else* rule.

Figure 4. Example of an *if-then-else* rule

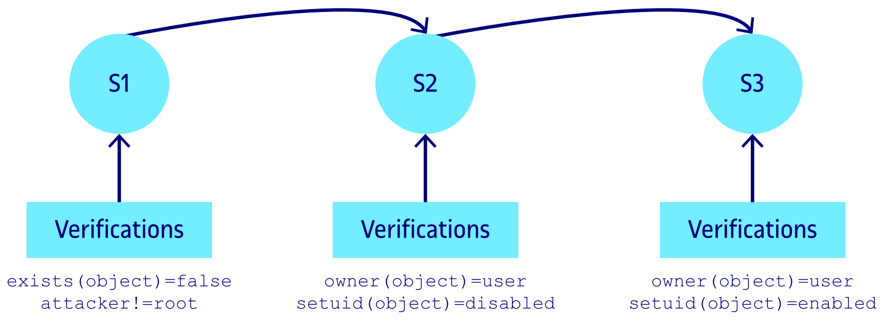


Although this model allows us to detect an intrusion from already known patterns, the main disadvantage is that these patterns do not define a sequential order of actions.

Detecting attacks composed of a sequence of events using this method can lead to great difficulties. The maintenance and updating of the pattern database are other critical points of this model.

2) State transitions. This model makes use of finite automata to represent the attacks, in which the nodes represent the states, and the arrows (arcs), the transitions. The use of transition diagrams (Figure 5) facilitates the association between states and the different steps taken by an intruder from the time it enters a system, with limited privileges, until it gains control.

Figure 5. Example of a state transition diagram



As the main advantages of this model, we can highlight that transition diagrams allow a high-level representation of penetration scenarios, and offer a way to identify a series of sequences that form an attack.

As well as this, these diagrams define very simply the attacks to be detected. The analysis engine could use different variants of the same diagram to identify similar attacks.

By contrast, transition diagrams, and therefore different steps in the sequence, must be created using specific languages that are often very limited and insufficient to recreate complex situations.

This limitation means that this model cannot detect some of the most common attacks, and that the use of additional analysis engines is necessary as a complement to this model. For example, engines with Bayesian inference or first-order logic, also used by deductive databases.

Anomaly-based detection

Event processors that base their detection on an anomaly scheme will try to identify suspicious activities by comparing the behaviour of a user, process or service, with profile behaviour classified as normal. A profile serves as a metric (measurement of a set of variables) of normal behaviours. Any deviation that exceeds a certain threshold with respect to the stored profile will be treated as evidence of an intrusion.

Disadvantages of anomaly-based detection

The drawbacks of the anomaly-based detection model make most of the commercial detection systems available today generally implement schemes based on the misuse model.

One of the requirements of this model is the need to initialise a default profile that gradually adapts to the behaviour of a normal (non suspicious) user, process or service. It is therefore necessary, to use heuristics and statistical descriptors that help to correctly model changes in behaviour as soon as they happen. Other proposals seek to incorporate artificial intelligence techniques to carry out these tasks (for example, use of machine learning with neural networks).

Anomaly-based detection has clear advantages over misuse-based detection. A first advantage is the possibility of detecting unknown situations (for example, *zero-day* attacks). This is possible since, regardless of the source of the intrusion, as soon as malicious activities start to deviate from the normal behaviour, the event processor will launch an alert.

Anomaly-based detection also has disadvantages. One of the first disadvantages is the lack of guarantee in the detection process. Stealthy intruders can slowly perpetrate malicious actions to evade detection.

As a second drawback, we can highlight the difficulty in explaining and accurately describing the attacks detected by anomaly-based analysers. Generally, an analyser not only has to trigger an alert, but also must specify where the attack comes from, what changes the system has undergone, etc.

Finally, the high rate of false positives and false negatives that can occur using this detection scheme is a serious drawback, since a deviation from the expected profile will not always coincide with an intrusion attempt. In the case of processors in which the events come from network-based sensors, it is possible that the number of triggered alerts easily get unmanageable. This often causes network administrators to ignore alerts issued by the detection system, or even deactivate the system altogether.

All of these drawbacks make most of the commercial detection systems available today implement their analysers by using the misuse detection model.

Complementary reading

The following article (available online) provides more information about possible improvements needed in the world of anomaly-based detection, to change the current commercial situation: Seng, S; Garcia-Alfaro, J; Laarouchi, Y. (2021). "Why anomaly-based intrusion detection systems have not yet conquered the industrial market?", *14th International Symposium on Foundations and Practice of Security, Springer Nature* <https://doi.org/10.1007/978-3-031-08147-7_23>.

Stored data

In most situations, the volume of events collected by the IDS, including the alerts triggered by the analysers, becomes so high that a storage process outside the detector is necessary. Let us suppose, for example, that all packets in a high-speed network must be inspected by the analysers of the detection system. In this case, it is necessary to consider an external storage hierarchy that reduces the volume of information without penalising the possibilities of analysis.

One possibility is the classification of information in terms of short and medium term analysis. In the case of a **short term** analysis, the information is stored directly in the same components of the IDS (in internal *buffers*), so that after processing the data, and transforming it into an event format, it is transmitted to other processing elements. In the case of **medium term** information, the pre-processed data is stored on secondary devices (with the appropriate format) rather than being transmitted to the internal system processors.

The storage time for medium-term information can be two or three days, with the aim of being consulted by the system processors if the analysis process requires it.

Eventually, and after a compression process (to reduce the size), some of the medium-term information may continue to be stored for long periods of time (around months or even years) waiting to be consulted by **the long-term** detection processes.

1.2.3. Response units and remediation plans

The response units are responsible for initiating or providing decision aid information, in relation to possible repair plans that respond to a detected intrusion. Very rarely will response actions be automatic (**active response**). Most of them simply provide information and require human interaction (**passive response**), in order to assist with additional information before activating the final response by a security expert (or the administrator himself).

Responses and **remediation plans** aim to act against an intrusion, trying to neutralise the tasks already performed by the intruder before reaching the final objective. An active response, right at the moment when the intruder is detected, is usually not recommended. In general, response units will attempt to simulate and provide decision aid information, while the intrusion is still ongoing. Thus, administrators will be able to try to anticipate and take actions that neutralise the objectives of the intruder. An example could be to suggest new configuration rules to the firewall system associated with the affected components, with the aim of blocking future network connections such as the one that the intruder originated. Another example of a remediation plan could be the execution of additional tools to track the next phases or attacks associated with the intruder, to be able to perform the appropriate analysis later.

In most cases, we will have rather passive responses, which are limited to launching an external alert, to inform and describe the attack detected to the system administrators.

Most response components offer different ways to send this information to administrators, such as email, SMS messages, WhatsApp, GLPI tickets, etc.

In the case of automating the remediation plan, without first going through the system administrators, there could be collateral damage. For example, an active response (without human intervention) could lead to a denial of service against legitimate users or systems. It is very likely that some of the alerts thrown by the processors may handle are incorrect (**problem of false positives**).

For this reason, if a response unit immediately cut off the connection that originated this false positive, it could imply the loss of work of a legitimate user or service.

Problem of false positives and false negatives

A false positive occurs in those situations where an IDS characterises legitimate traffic, which is not part of any attack, as malicious; it is, therefore, an event detected by mistake. Contrarily, a false negative takes place in those situations in which malicious traffic is discarded and considered legitimate traffic by mistake; it is, therefore, an event that should be detected, but which escapes the detection process.

In most systems (e.g., e-commerce servers) such errors can lead to customer loss, which is inadmissible. Hence, most companies in the electronic commerce sector opt for hiring specialists who manually analyse the reports generated by the detection system in order to determine whether an active response to this guidance is necessary or not.

Finally, it should be noted that, like sensors, response units can also be classified into different categories according to the point of action. Both categories are again **host-based response units** (in charge of acting at the operating system level, for example, by blocking users, process completion, etc.) and **network-based response units** (in charge of acting at the network level, for example, dropping out connection attempts, applying network address filtering, etc.).

2. Management of events, alerts and incidents

In the previous section we introduced intrusion detection systems as isolated solutions, with components implemented in general architectures that allow the detection of elementary actions. These actions usually correspond to previous stages of an intrusion or the final objectives of an intrusion, that is, actions coordinated with complex objectives, beyond mere data theft. We are talking about coordinated actions that can seek long-term attacks, such as disrupting industrial or financial processes, with periods that go beyond months and even years. In order to anticipate these situations and prepare remediation plans accordingly, additional elements must be used to complete and conclude the detection process made by the components of the traditional IDS that we have seen in the previous section.

In this section, we will deal with these additional elements, which we summarise with the concept of *NMS and SIEM*[8] type of platforms. Although there are many differences between both concepts (the NMS are more focused on error diagnosis, and SIEM are more focused on the detection and activation of remediation plans), in this section, we will refer only to SIEM platforms, understanding it as a combination of both concepts.

(8)NMS stands for *Network Management System* and SIEM stands for *Security Information & Event Management*

Recommended links

Some products (with commercial and free software versions), related to the NMS and SIEM concept, can be consulted at the following links: www.nagios.org (NMS), www.vigilo-nms.org (NMS), www.prelude-siem.org (SIEM), https://cybersecurity.att.com/products/ossim (SIEM).

Coordinated actions and distributed attacks

Situations that cannot be identified by looking for patterns in isolation, but must be detected from the combination of multiple indications found at different points in a system.

SIEM platforms are absolutely essential nowadays. Although they have existed for more than three decades, the expansion of the Internet in corporations and institutions has led to the recent developments and improvements that we summarise below:

- **Collection and normalisation of events**. A SIEM must be able to manage the collection of events from extremely heterogeneous sources. Therefore, a process of data normalisation must be carried out, not only at the syntactic level, but also at the semantic level. At the syntactic level, the existence of formats and standards, such as the IDMEF[9] and IODEF[10] formats can help in the tasks of syntactic normalisation of alerts from heterogeneous collectors or detection sensors. As for semantic normalisation, the use of ontologies, modelling languages such as the UML (*Unified Modeling Language*), or any other mechanism for the formal representation of alerts, can help in this matter.

(9)Stands for *The Intrusion Detection Message Exchange Format.*

(10)Stands for *Incident Object Description and Exchange Format.*

- **Consolidation of the monitoring functions of the detection tools**. The second vital function attributed to a SIEM is the consolidation of low-level alerts produced by the security components of a network, among which firewall systems, IDS, antivirus and vulnerability detection systems can be highlighted. As we have seen in the case of IDS, network security tools are vulnerable to problems related to false positives and false negatives. For this reason, what is expected from the use of a SIEM is the management of the processes of fusion, aggregation and correlation of alerts from previous hosts, and to reduce the rate of false positives. In addition, they will improve the diagnosis of errors to also reduce false negatives.

- **Preparation of remediation plans**. In general, most IDSs are usually configured as purely passive mechanisms. However, solutions available nowadays provide the technology needed to turn them into active or semi-active solutions (waiting for confirmation from an operator before activating the reaction process), with the aim of being able to react and neutralise the detected activities or actions. As we have seen in the previous section, this functionality must be analysed and activated with caution. The normalisation and correlation tasks of a SIEM should improve this point and reduce the potential collateral damage of remediation plans.

We will now detail some of the required tasks in order to carry out the functionalities that are expected of a SIEM.

2.1. Configuration of event collectors

In addition to information from an IDS, the data collected by a SIEM may come from any other component with the capability of creating log files, such as:

- **Firewall systems** (or any other type of component for packet filtering through network-level access control lists). Although we have often seen these components as prevention mechanisms, which aim to block traffic considered dangerous to the system, these components can be configured to report the information observed through their network interfaces. Their audit files are therefore of great interest to complement the process of aggregation and correlation of events of the SIEM.

- **Email servers** (based on protocols such as SMTP, POP or IMAP). Again, the audit files generated by these servers will offer a wealth of information to characterise and discover malicious activities, such as the spread of worms, *zero-day* attacks, or to control traffic associated with other illegal activities.

- **Traffic management servers** (based, for example, on the SNMP protocol). The information reported by the components will also contain information associated with actions related to the violation of the internal security policies of the organisation where they are installed.

Another thing to keep in mind during the configuration of the SIEM detection sensors will be the data, events and alerts provided by an IPS (Intrusion Prevention System). An IPS is primarily based on research into the vulnerabilities of equipment and systems. Most IPS-type solutions actually combine intrusion detection techniques with traditional access control mechanisms. The border between IDS and IPS is currently difficult to define and can be complemented with many other preventive solutions, such as vulnerability detection systems and antivirus systems.

Vulnerability detection systems try to analyse the configuration of systems deployed in network, with the aim of discovering poorly configured parts that potentially present vulnerabilities that could be the target of attacks. These systems also include detection of software defects (i.e., programming errors or *bugs*), conception errors in the topological configuration of a network, hardware errors, etc. However, antivirus systems are designed to protect workstations and servers against known malware. Most of these systems will use a database of antivirus signatures that identifies the known malicious software. From the prevention point of view, it is expected that both a vulnerability detection system and an antivirus will be able to correct vulnerabilities and disinfect the equipment that are malware victims. Thus, the installation and combination of an IDS with other prevention mechanisms has the ultimate aim of detecting and reacting to the general concept of *intrusion*.

2.2. Information and collection policies

Since the purpose of a SIEM is to be able to provide system operators with centralised management capabilities, it is common for the configuration of the collection components associated with the SIEM (IDS and the components of previous examples) to be done through the SIEM user interface. That is why a global information collection policy is usually used, administered by the SIEM, and which can then be refined for the local configuration of each of the equipment associated with the SIEM (IDS, firewall, mail servers, etc.).

This policy, or its associated sub-policies, is usually defined by rules, based on the use of regular expressions, the search for traffic patterns, signalling protocols, etc. The components may also incorporate the possibility of dealing with a collection of events through configurations and based on the recognition of anomalous activities. Consequently, the policy must offer the required syntax and semantics to be able to deal with the use of statistical models, collected through data mining, expert systems, Bayesian networks, etc.

SIEM's event collection and log information policy may also involve executing a set of actions and preparing the data before passing them to the next stage. For example, it may involve data preprocessing from the execution of regular expressions, with the aim of conducting a search for specific patterns (search for signs of a particular virus, or for a specific port scan action with specific system components), as well as replacing specific parts of the events to improve the identification of a particular component or to enrich the data (for example, adding computer names as a complement to their IP addresses).

Some other technical criteria that will need to be considered in a SIEM's event collection policy will be to determine precisely the location of the collection components that must be controlled in the system. This aspect will determine the coverage and the global vision modules that the system will incorporate to ensure that the subsequent process of aggregation and correlation of information guarantees quality management. Therefore, it will also be necessary to specify, within the collection policies, the physical topology and the logical description of the system that the SIEM must monitor. It is important to be able to treat and structure this system in terms of subnetworks, so that the parts of the system that have a greater need for surveillance can be identified, and thus the correlation functions can recognise the tasks associated with specific incidents for each subnetwork of the system. For example, if the system contains DMZ-type areas (demilitarized zones), it should be possible to create specific rules to closely monitor the actions being taken on this part of the network. The use of subnetwork masks (which is specified in terms of CIDR[11] or interdomain routing without classes, for example) is common in the log file collection policies of most SIEMs.

(11)Stands for *Classless Inter-Domain Routing.*

2.3. Normalisation of the collected information

Once the information is collected, and before moving on to subsequent processes for detecting attack scenarios, the SIEM must ensure that it is possible to put into correspondence all the data coming from the detection of the same event (such as malicious, suspicious or anomalous events). Thus, there must be a set of normalisation processes to preprocess the collected data and anticipate problems that could hinder the correspondence of data related to events derived from the same network traffic flow (such as the origin of the traffic, the destination, the ports, etc.) and that correspond to the supervision tasks of different monitoring sensors installed in different places of the system (either deployed in the same domain or in different domains).

The normalisation process must ensure, for example, that data associated with network traffic have the same format with regard to the classification of the origin of traffic and also metadata associated with the time at which the event was detected, associated protocols and services, origin and destination addresses, content of the packets associated with traffic, etc. Since the nature of the sensors is potentially heterogeneous, the normalisation process must guarantee, either with SIEM proprietary formats, or with efforts from existing standards, that there are no interoperability problems that limit the expressiveness of the information that the SIEM correlation module will have to add.

Examples of normalisation efforts

Some examples of efforts regarding the problem of normalisation are the following:

- CIDF (Common Intrusion Detection Framework)
- IDWG (Intrusion Detection Working Group)
- IDMEF (Intrusion Detection Message Exchange Format)
- INCH (Extended Incident Handling)
- FINE (Format for Incident Report Exchange)
- IDIP (Intrusion Detection and Isolation Protocol)
- OSEC (Open Security Evaluation Criteria)
- IODEF (Incident Object Description and Exchange Format)

Each of the above examples has tried to define common languages to specify the description of events and the activities associated with the events that security components must exchange. Apart from languages (both syntactic and semantic) to guarantee homogeneity in data exchange, it is also a priority to allow the description of a common process that specifies the precise protocol for the exchange of data between the different sensors configured in a SIEM. Of the examples listed above, two of the formats and procedures that have the support of working groups of the Internet Engineering Task Force (IETF) are the *IDMEF*[12], RFC 4765 and the *IODEF*[13] (summarised in Figure 6).

(12)Stands for *Intrusion Detection Message Exchange Format.*

(13)Stands for *Incident Object Description Exchange Format.*

Link of interest

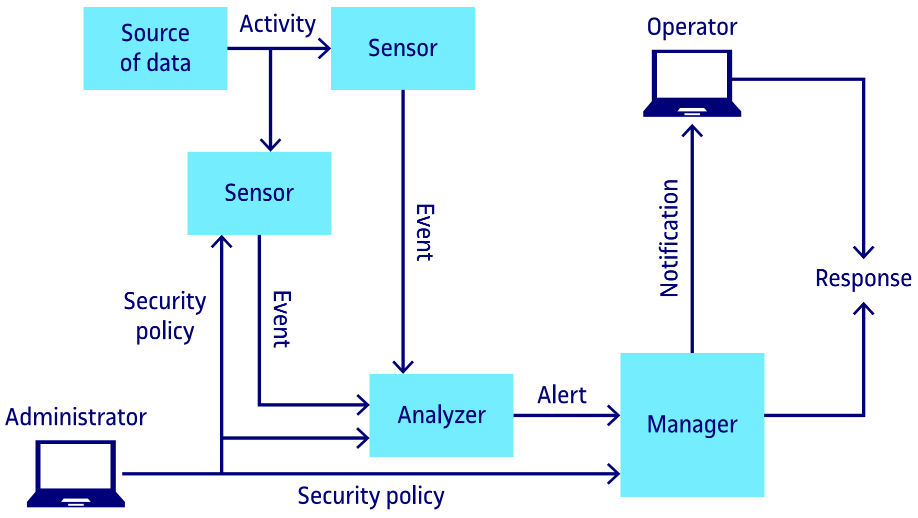
The last version of the IDMEF and IODEF specifications is now being revised. For more information, we recommend seeing the http://www.secef.net/ website.

Complementary reading

You will find more information about the model associated to IDMEF in the following book:

Stallings (2017). *Cryptography and Network Security - Principles and Practice* (7th edition), Pearson Education, Inc., Hoboken, NJ. All rights reserved. See chapter 22 for more information.

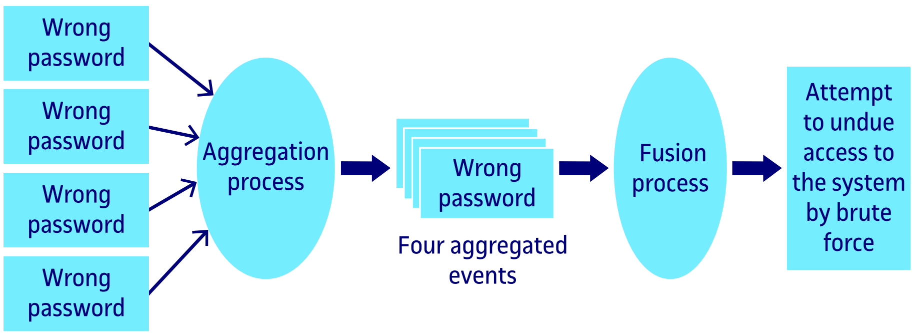
Figure 6. Components associated with efforts such as IDMEF and IODEF



2.4. Aggregation and fusion of information

Data aggregation and fusion functions are used to intelligently reduce large volumes of data that are likely to contain redundant events (repetitions and congruent information). Both functions must be applied before matching events detected by different system components. First, the aggregation process will have to be responsible for grouping the data resulting from the detection of the same event, reported by one same sensor or by different ones. Once the grouping of this information is done, the process of merging the information will take place, with the aim of summarising and offering a single datum that characterises the detected event. Figure 7 shows the difference between the aggregation and fusion functions with a simple example.

Figure 7. The difference between aggregation and fusion



During the aggregation process, the information corresponding to detected events will be grouped into sessions and will attempt to unify the data of the same event so that they can be used later during the fusion process, such as the source address, the destination address, the ports, the protocols, etc. In this way, the different data associated with an attack on a specific element of the system will be grouped into a single session and a single identifier. The rest of the data reported by other system sensors will be linked to the same reference, so that during the final fusion process it will be possible to generate an identifier alert. This alert will contain all the information observed by the different sensors configured by the SIEM. The alerts generated from the fusion process will therefore contain a synthesis of all SIEM knowledge about each one of the basic attacks observed by the monitoring system. As a result, the amount of data needed to be stored in the system is reduced without any loss of information. Once all the information reported by the system has been merged, the alerts will be communicated to the last SIEM component, responsible for managing and matching (correlating) the flow of alerts.

2.5. Correlation of alerts and generation of reports

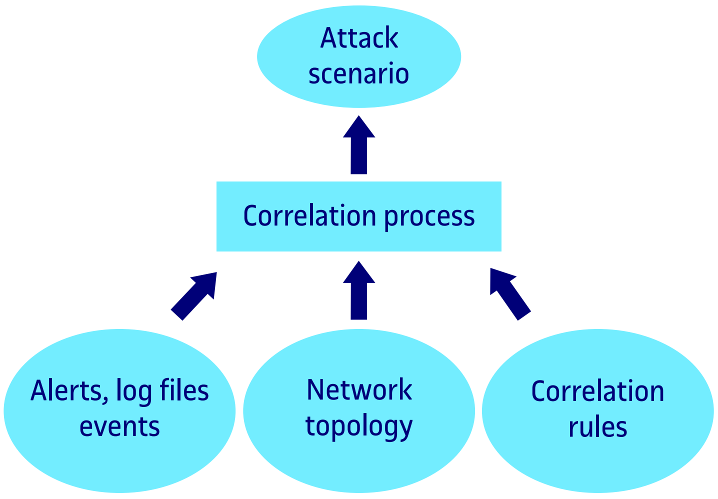
In general, we can define the *alert correlation process* as the conceptual interpretation of multiple alerts with the aim of providing a semantic improvement and reducing the overall amount of alerts in an intrusion detection system.

The correlation of alerts is therefore considered one of the keys in the evolution of intrusion detection systems, as it tries to solve the most relevant drawbacks of these systems (i.e., excessive alerts), improve them semantically and reduce false positives and negatives. The studies related to the correlation of alerts in the field of intrusion detection are relatively recent. Most of them deal with observations and experiments made in current systems. The theoretical part in this field is still in the process of evolution.

Most commercial SIEMs still do not have real functionalities for conducting a full alert correlation. Although most of the solutions that exist talk about correlation services, the vast majority are limited to storing logical links between alerts stored in the same relational database that the responsible administrator can then consult with a control console. In contrast, within the field of academic research on intrusion detection systems, there are a large number of proposals for the inclusion of alert correlation techniques in new generation systems.

Most of these proposals essentially exploit the information contained within the alerts and, in addition, try to make explicit references to annexed knowledge, necessary for the final process of matching all the events observed in the same system. Therefore, the correlation in these systems is not limited only to the information contained within the alerts generated by SIEM probes, but they also make use of a prior knowledge of the surveillance state of the systems, on the attacks that can be made, and even on the topology of the system and the correlation rules generated by the operators and interpreted by expert systems. Figure 8 summarises these types of correlation systems.

Figure 8. Correlation function



As we can see in Figure 8, the first entry in the correlation process corresponds to the events already aggregated and fusioned by the lower SIEM modules. The objective of the correlation process is precisely to match these groupings of events and reconstruct attack scenarios to which the observed actions could belong. To do this, it will be necessary to combine the received alerts with the physical and logical information about the system (for example, the topology of the system and the structuring of the IP addressing of the equipment) and the predefined knowledge of malicious activities. The latter is represented in Figure 8 in the form of correlation rules.

The physical and logical properties of the network must be specified in a topological database, either manually (by system administrators), or through the use of automatic discovery and assisted creation services of network topologies. The latter must be responsible for the creation of topological maps with the configurations of the devices and the security policies associated with the protected information system.

The awareness of malicious activities must be configured in the system by means of correlation rules. Each correlation rule will be defined by characterising sets of actions corresponding to the same intrusion scenario, and will specify the required conditions to carry out an action and the consequences on the system after the execution of each action. As with the detection rules of an IDS based on misuse, these correlation rules will be the basic element to ensure the detection of attack scenarios formed by the actions detected in the system. Therefore, to ensure the effectiveness of the correlation process, it will be of paramount importance to ensure a correct configuration of the relevant correlation rules for each system, as well as its correct updating and commissioning. The elimination or corruption of a single rule containing information about multiple incidents may result in the loss of detection of a large number of scenarios. The configuration of this part of the SIEM will therefore be very prone to errors and requires a deep knowledge by the operators in charge of configuring the system.

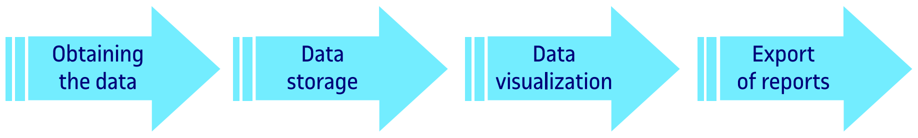
The detection of an incident or intrusion scenario is derived, during the correlation process, from the series of events indicated by the set of correlation rules preconfigured in the system. From a small set of correlation rules, it is possible to define many intrusion scenarios. Thus, the aim of the correlation process is to reduce the excess of information that the system operator will have to manage. Instead of asking the administrator to analyse thousands of events, the correlation process provides the generation of incident reports. Each report will contain the representation of scenarios that, with a high probability, could have been developed in the system from the events (i.e., primitive actions) detected by the SIEM probes.

With the same aim, the correlation process can also be configured to reduce the number of false alerts that are analysed (false positives). To do this, the system can be configured to execute internal verification actions triggered after the generation of each intrusion scenario and verify the certainty that this incident has occurred in the system or not. In this way, incidents that can be discarded with a high probability will be eliminated from the final report that the operator must prepare. This internal verification may be limited, for example, in the execution of an analysis of vulnerabilities of the system that serves to decide if a specific incident can be discarded, if the associated vulnerabilities are not present in the system represented in the system's topological database; that is, an incident involving the exploitation of vulnerabilities or services not deployed in the system may be discarded with a high probability and thus, avoid overloading the system operator's analytical capabilities.

Additionally, these incidents may be reported as false positives rather than alerts or intrusion scenarios. All this will facilitate the ordering of the response mechanisms in the system and also the optimisation of the remediation actions that will have to be deployed in the system on the real incidents that have been detected.

Finally, as a graphic complement to the alert correlation, most SIEMs offer graphical interfaces (of the *dashboard* type) to manage the generation of reports and provide command control to the end user. Figure 9 summarises this process, concluding the presentation of the typical components of a SIEM.

Figure 9. Stages associated with the generation of SIEM reports



3. Detecting network intruders with Snort

Snort is a very complete open source security tool for creating intrusion detection systems in network environments. It is very popular among the community of network and service administrators. Thanks to its ability to capture and record packets in TCP/IP networks, Snort can be used to implement from a simple packet sniffer for traffic monitoring of a small network to a complete intrusion detection system in real time.

As a network monitor, Snort behaves like a true vacuum cleaner (hence its name) of IP datagrams, which offers different possibilities in terms of their treatment: from acting as a simple passive network monitor that is responsible for detecting malicious traffic that circulates through the network to the possibility of sending all the captured traffic to log file servers or database servers.

But, apart from excellent features such as packet detector and alert generator, Snort has many other features that have allowed it to become one of the most complete software solutions for the construction of detection systems in network environments based on pattern recognition. Snort is defined as a *Lightweight* Network Intrusion Detection System sensor. This *lightweight* qualification means that, as an IDS, its design and implementation allow it to operate under different operating systems and that its functions as a detection mechanism may be part of different (even commercial) security products. The latest versions of Snort also define themselves as an IPS, that is, an Intrusion Prevention System, since they offer the possibility of blocking traffic, in active response unit mode, as a complement to the detection of intrusion attempts. However, in this section we will discuss Snort only as a collector of network events, that is, as a network sensor for the detection of intruders, since this is its initial, and best known, function.

In fact, Snort's popularity has increased in recent years in parallel with the increasing popularity of open source operating systems, such as the Linux and BSD family of systems. However, its nature as an open source product does not limit it to being available only for such operating systems. Snort can run on commercial solutions, such as Microsoft Windows.

From the point of view of its detection engine, Snort would be part of the detection category based on misuse. Using signature recognition, Snort will match all captured traffic against its detection rules.

A Snort detection rule is nothing more than a set of requirements that will allow it to activate an alert, if they are met. An example would be a Snort rule that would verify the use of P2P applications for file sharing over the Internet, while verifying the use of the GET chain in services other than the traditional HTTP protocol port. If a packet captured by Snort matches this simple rule, its notification system will issue an alert to indicate the facts. Once the alert is released, it can be stored in different ways and with different formats, such as a single system log file, an entry into an alert database, a SNMP event, etc.

Commercial version of Snort

Although Snort is available under the GPLv2+ (GNU Public Licence, version 2) licence, it also offers alternative licences and commercial products. Most alternative products are based directly on Snort and distributed by the Cisco company.

Cisco acquired the rights to Snort in 2013, along with the Sourcefire company, founded by the creator of Snort, Marty Roesch. The analysis of commercial versions of Snort is beyond the scope of this didactic module. For more information, you can visit http://cisco.com and search for information about these products.

We will now see the origins of Snort and an analysis of its architecture and some of its most remarkable features.

3.1. Origin and architecture of Snort

More than just a detector

The author of Snort was trying to indicate with the name Snort that it was more than just a detector. The term *snort* means inhaling obsessively and violently. In addition, Marty said at the time that he already had too many applications called *a.out* and that all popular names for detectors, called TCP-*something*, were already occupied.

Very briefly, we can define Snort as a packet sniffer with additional functionalities for packet logging, alert generation and a misuse-based detection engine.

Snort was developed in 1998 under the name APE. Its developer, Marty Roesch, was trying to implement a multiplatform packet detector (although the initial development was made for the GNU/Linux operating system) that had different options for classifying and displaying the captured packets. Marty implemented Snort as an application based on the libcap library (for the development of packet capture), which ensured great portability in both the capture and in the collected traffic format.

Snort began to be distributed through the *Packet Storm* website (http://www.packetstormsecurity.com) on the 22nd of December, 1998, with only one thousand six hundred lines of code and a total of two source files. At that time, the main use given to it by its author was an analyser of his network connections with a cable modem and as a debugger of the network applications he was implementing.

The first signature analyser developed for Snort (also known as a rule analyser by the Snort development community) was added as a new application functionality in January 1999. This new functionality allowed Snort to start being used as an intrusion detector.

In December 1999, Snort version 1.5 was released. In this version, its author decided on a new architecture based on plugins, which is still preserved in current versions. After this version, Marty Roesch left the company where he worked and began to dedicate himself full-time to the task of adding new functionalities to improve the configuration capabilities and facilitate the use of Snort in more professional environments. Thanks to Snort's great acceptance among the community of administrators, Marty thought it was a good time to offer his product with a support for companies and obtained the necessary funding to establish Sourcefire. Cisco acquired the Sourcefire rights in July 2013.

However, Snort remains free code and promises to do so forever. Version 3 of Snort features more than 100,000 lines of code and involves a complete restructuring of the original design of its initial architecture. The free version of Snort, under the GNU licence, can be freely downloaded from Snort's website (https://snort.org) and allows any user to have support for the latest available versions and the latest updates to the rule files for those versions.

Recommended links

Version 3 of Snort is available at https://github.com/snort3/snort3.

The main differences between version 2 and version 3 of Snort can be quickly consulted on the following website: https://blog.snort.org/2020/08/snort-3-2-differences.html.

Currently, Snort has a large repertoire of accessories that allow reporting notifications to different database managers and a large number of traffic preprocessors that allow analysing remote calls and port scanning before these are contrasted with the set of rules associated with the search for new incidents.

Snort's rule sets have also evolved as the application has grown. The size of the rules sets for the latest Snort version available for download is increased in a similar way to the speed of new releases of exploits. These rule files are currently classified into different categories such as P2P, denial-of-service attacks, attacks against web services, viruses, pornographic traffic, etc.

Each of these rules is associated with a unique identifier (Sensor ID, SID) that allows recognising and finding information about the detected attack or misuse. For example, the SID for the SSH *banner attack* is 1838. In addition, thanks to Snort's extensive use among the network administrator community, other intruder detection tools have adopted the Snort rules format and also the coding used for the dumping of the captured packets (based on *libcap*).

Support for these rule files is increasing every day. In this way, any user of Snort, or any other network detection tool with a compatible rule format, could create its own rules as new attacks appear and collaborate with the Snort development community to keep its signature base perfectly updated.

3.1.1. Snort's basic architecture

Snort provides a set of features that make it a very powerful security tool, among which are the capture of network traffic, the analysis and registration of captured packets and the detection of malicious or dishonest traffic. Before seeing Snort's outstanding features in more detail, it is important to know and understand its architecture.

Snort consists of a set of components, most of which are developed as plugins that allow the customisation of Snort. Among these components, the preprocessors stand out, which allow Snort to manipulate the content of the packets more efficiently before moving it to the detection element; its notification system is also outstanding, which allow the reported information to be sent and stored in different formats and following different methods.

Snort plugins

The term *plugin* refers to the software modules of an application, developed independently of the general kernel of the application. The aim is to add additional features without affecting the core source code or other components. To do this, the application must provide an API (*Application Programming Interface*), that allows the development and compilation of such complements. Thus, a Snort plugin is a component developed according to the Snort plugin API, which will be used alongside the core of the Snort code, but separated, so that a change in the component code does not affect the kernel or other components.

Snort's central architecture is based on the following four components:

- Packet detector
- Preprocessor
- Detection engine
- System of notifications

According to this structure, Snort will allow the capture and preprocessing of network traffic through the first two components (packet detector and preprocessor) and will then check them through the detection engine (according to the set of activated rules) and will generate, through the last of the components, the relevant notifications.

Figure 10 shows Snort's basic architecture that we have just commented on. Looking at the figure, we can draw an analogy between Snort and a mechanical machine for automatic coin sorting:

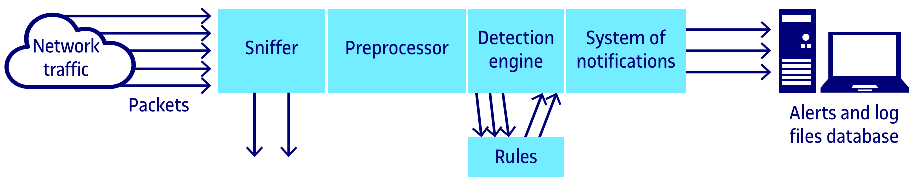
1) It takes all the coins (network packets collected by the packet detector).

2) Each coin will be dropped down a ramp to determine which group of coins it belongs to (packet preprocessor).

3) It sorts the coins according to each coin type and packs them into cannons according to the category (detection engine).

4) Finally, the administrator will decide what to do with each one of the ordered coin cannons (notification system).

Figure 10. Snort's basic architecture



Snort's preprocessor, detection engine and notification system are also implemented in the form of independent components. We will now examine in more detail each of the basic components of Snort that we have just seen.

3.2. Packet sniffer and preprocessor

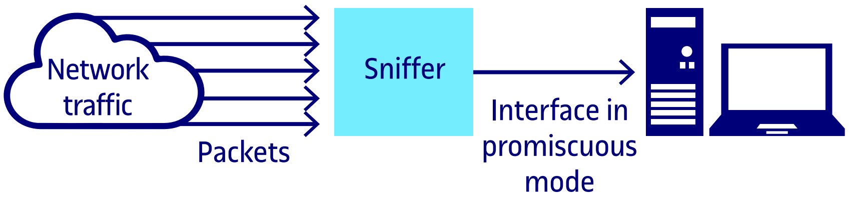
A sniffer is a device (software or hardware) that is used to capture packets travelling through the network to which it is associated.

In the case of TCP/IP networks, this traffic is usually IP datagram traffic, although it is also possible to have traffic of different types, such as UPnP traffic or (Apple) Bonjour traffic. In addition, since IP traffic also consists of different types of protocols, such as TCP, UDP, ICMP, routing protocols, IPSec, etc., many sniffers will need to know beforehand the type of traffic to be able to later interpret the packets that are being collected, and display them in a language that a network administrator can understand.

Like many other tools related to network security, sniffers can be used for more or less dishonest purposes. Among the different uses that can be given to a sniffer, we can think of traffic analysis for the solution of congestion and network problems, improvement and study of the performance of resources, passive capture of sensitive information (passwords, user names, etc.).

Thus, like the rest of traditional detectors, the Snort packet decoder will be the element in charge of picking up the packets, which the other components will examine and classify later. To do this, the packet sniffer must be able to capture all the traffic that it can, to then pass it to the next component (the preprocessor), which will be responsible for detecting what type of traffic has been collected. Figure 11 shows a diagram of how the Snort packet sniffer works.

Figure 11. How the Snort packet sniffer works



Network interface in promiscuous mode

The promiscuous mode is the way in which the network card of a computer connected to a network (whether a wired network or a wireless network) will allow the capture of all the traffic that flows through this network interface.

As the Snort packet sniffer collects the traffic passing through the network, it will deliver it to the preprocessing element. This element will adapt the captured packets and deliver them to the detection engine.

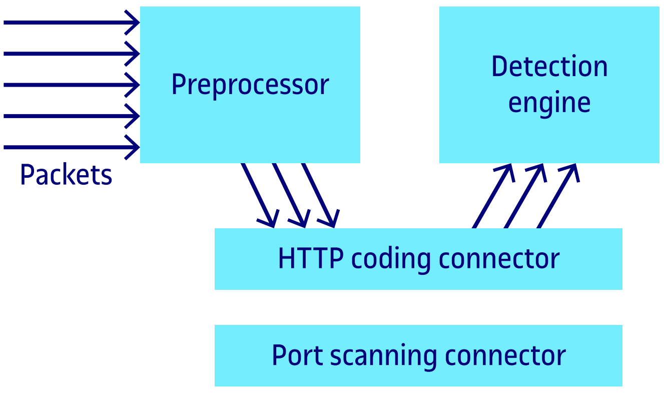
Thus, the preprocessor will get raw packets and will verify them through a set of plugins. For example, it will use a plugin for the treatment of packets related to RPC[14] type traffic, or a plugin for the treatment of packets related to port scans. These plugins will verify packets in search of certain behaviours that allow Snort to determine their type. Once the type is determined, the packet will be sent to the detection engine.

(14) Stands for *Remote Process Calls.*

This preprocessing feature is really important for a detection tool, as it is possible to use third-party applications that can be activated and disabled according to the needs of the preprocessing level. For example, if a network administrator is not concerned about the RPC traffic that enters and leaves their network (and therefore does not need to analyse it), for whatever reason, it will be sufficient to disable the RPC plugin and continue using the rest.

Figure 12 shows a diagram in which the Snort preprocessor uses two of its plugins to verify the type of packets it receives and decide whether to move them to the detection engine or not.

Figure 12. Snort preprocessor that combines two data processing plugins



Snort preprocessors offer great flexibility for the implementation of different traffic processing algorithms.

3.3. Rules and detection engine

As we have already mentioned, Snort bases its detection on the misuse model. For this reason, Snort must be configured using a set of rules that will use the detection module to perform the recognition of attacks and intrusion signatures. Snort rules are usually grouped into signature sets that categorise incidents. Thus, we will find sets of rules associated with the detection of Trojans, the detection of intermediate buffer overflow attacks, and so on.

Each rule can be divided into two parts. First, we have the header of the rule, in which we indicate the action associated with this rule in case it is fulfilled (generation of a log file or generation of an alert); the type of packet (TCP, UDP, ICMP, etc.); the address of origin and destination of the packet, etc. Second, we have the option field of the rule, in which we will find the information that must be contained in the packet (in the data part, for example) so that the action associated with the rule is activated.

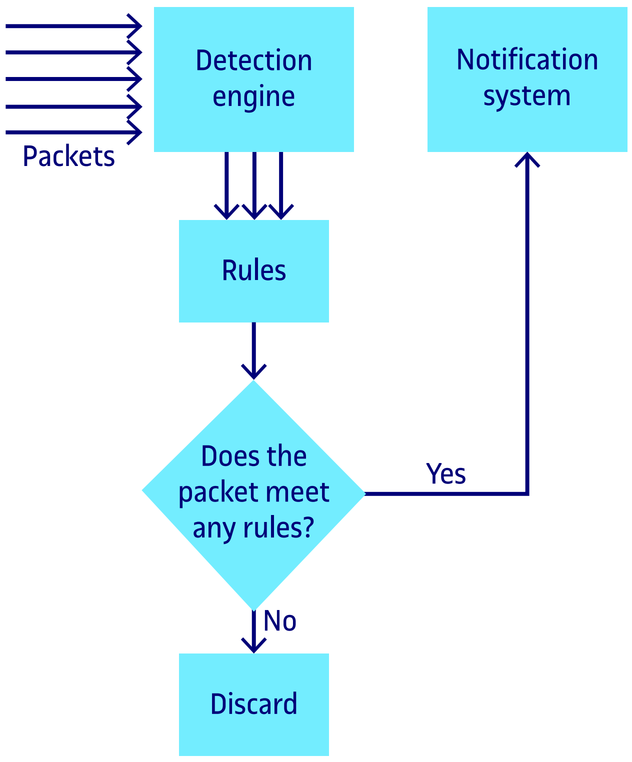
The action defined in a Snort rule can be chosen from five basic actions:

- ***Alert***. Generation of an alert that also contains the log information corresponding to the packet that activates the rule.
- ***Log***. Generation of only the log information associated with the content of the packet associated with the activation of the rule.
- ***Pass***. Pass the packet associated with the rule without having to log or alert about the event.
- ***Activate***. Generation of an alert together with the activation of a dynamic rule (see Dynamic action).
- ***Dynamic***. Rule that remains inactive and is activated through the activate action, for example, to trigger the generation of logs associated with the packets associated with the initial rule to obtain additional information regarding traffic subsequent to the packet that triggered the rule (and with a given time duration).

Using Snort in IPS mode can provide additional actions, such as reactive actions to block detected traffic. In this case, the use of additional actions (of *drop*, *sdrop* and *rejected* types, as in firewall systems), allows to directly reject the packets identified by Snort as suspicious.

In Snort's detection engine we find the required treatment algorithms to complete the detection process. Based on the information provided by the preprocessor and its associated plugins, the detection engine will contrast this data with the rules base defined by the operator. If any of the rules matches the information obtained, the detection engine will be responsible for warning the notification system, indicating the rule that it has skipped. Figure 13 shows a simple outline of the overall behaviour of Snort's detection engine.

Figure 13. Snort's detection engine



Of all the elements we have seen, the detection engine and the syntax used by the detection rules are the most complicated and difficult parts to understand when studying Snort's behaviour. However, once we start working with Snort and have minimally learned the syntax used, it is quite easy to get to customise and adjust the behaviour of Snort's detection functionality. In addition, the rule sets can be easily activated or deactivated, so that the desired detection behaviour can be defined according to the type of network in which Snort will be configured.

3.4. Notification system

Once the information captured by the Snort packet decoder is analysed by the detection engine, the results must be reported in some way. Through this component, this function can be performed and the results can be generated in different formats and to different equipment.

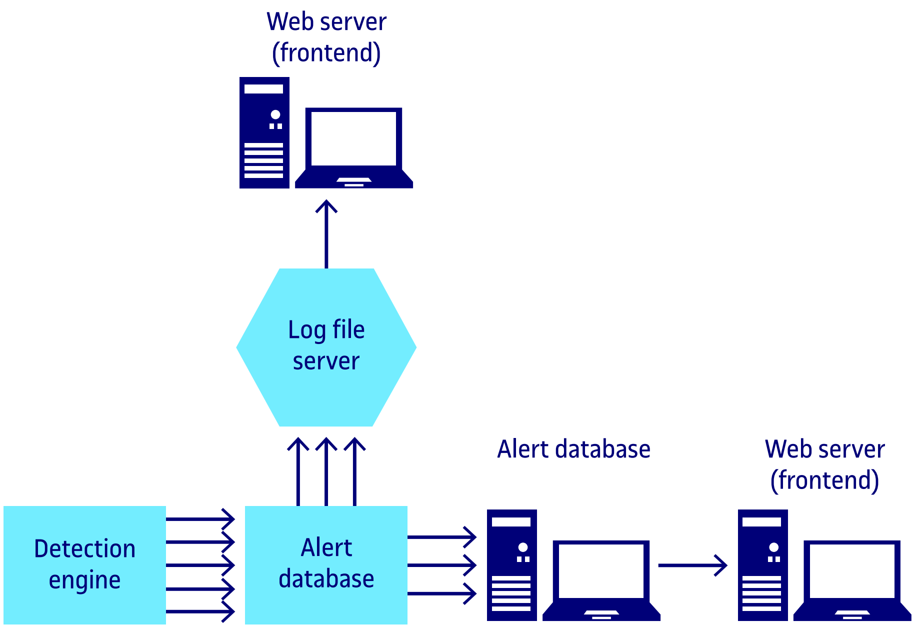
When the detection engine launches an alert, the result may involve the generation of a log file, sending the alert over the network through an SNMP message or even storing the information associated with the alert in a structured way by some database management system. Many of these tools are available and can be freely downloaded from the Internet.

As in the case of the detection engine and the preprocessor, Snort's notification system also uses a plugin system for the processing of information. Figure 14 shows a diagram for the operation of this notification system using plugins.

Snort alerts

If we try to run Snort in a congested network with a reasonable number of detection signatures activated, the amount of alerts launched by Snort can reach a hundred in a short time. This huge volume of information will not be easy to deal with using a simple browser of log files.

Figure 14. Snort notification system, through plugins



The purpose of installing Snort on a network is not only to obtain information (intrusion attempts), but to analyse this information and to be able to take the necessary actions based on the data obtained. If the number of active rules is high and the network traffic increases easily, it will not be very easy to analyse the information reported by Snort unless we use some other complementary tools, such as correlations and visualisation tools.

Furthermore, the interesting aspect of a detection system like Snort is not simply to log events, but to be able to react to intrusion attempts in a reasonably short period of time. Therefore, it will be necessary to use second applications that help consolidate and analyse the information reported by Snort in order to alert network administrators of the intrusion attempts analysed.

Currently, there are a large number of utilities to work with the information generated by Snort. Some of these tools are maintained by the same community of Snort developers, although we can also find applications developed by third parties or commercial applications.

4. Deception systems and techniques

As we have just seen in the previous sections, the detection of intruders assumes that an attacker is able to violate the security policies of a system. Thus, tools are used to inform system administrators, prepare reports and remediation plans, with the ultimate goal of reacting to attacks by an intruder in the appropriate manner. We have also seen that classic intrusion detection tools have evolved towards complete incident detection and management platforms, with the aim of facilitating the management of large volumes of information created by detection tools, not only tools for intrusion detection, but also events generated from other technologies, such as vulnerability scanners, or decoy and deception systems.

Specifically, these decoy and deception systems can be seen as complementary technologies. They can provide additional information to the management, information fusion and alert correlation platforms that we have seen before. Instead of addressing the problem of cyber defence from a preventive point of view, these systems try to change the rules of the game, offering the defender (e.g., the network administrator) the possibility of taking the initiative.

> Decoy and deception systems, rather than neutralising the actions of a possible intruder, try to use similar techniques, but with the aim of learning from the intruder’s offensive actions.

We will now summarise two strategies that can be used when building such deception systems and techniques.

Recommended reading

The following article (available online) provides more information about possible improvements in implementations of the honeypot concept from the use of digital twins: **Eckhart and Ekelhart** (2019). “Digital Twins for cyber-physical security: State of the art and Outlook”, *Security and Quality in Cyber-Physical Systems Engineering, pp. 383-412, Springer Nature*. <http://dx.doi.org/10.1007/978-3-030-25312-7_14>

First of all, we have a strategy known as the *installation of honeypot systems*. It is about installing network equipment that tries to attract traffic from an intruder, imitating as much as possible the real behaviour of the operating systems of the network in question. Thus, the network administrator will be able to see the attempts of a possible intruder before it enters the real computers. The aim can also be to see how the security elements implemented in real equipment behave. In this case, emulators, co-simulators and digital twins are usually used for creating the deception equipment.

Decoy and deception equipment, also known as ***honeypots***, is network equipment that attempts to attract traffic from one or more attackers. In this way, their administrators can see attack attempts to enter the system and how the security elements implemented in the network behave.

Another aim is to obtain information about the tools and knowledge needed to make an intrusion in network environments such as those we intend to protect. All this information will end up serving to stop future attacks on the rest of the production equipment.

The conceptual idea of a *decoy and deception system* has existed for several decades. As a first approximation, we could define it as a network resource designed so that different attackers can introduce themselves in a simple way. These computers are usually designed to mimic the behaviour of production computers in order to be of interest to a community of attackers.

They usually have prevention mechanisms so that an intrusion to the decoy and deception computer does not end up causing an intrusion to the real computers of the network. Naturally, if an intruder manages to attack the computer, he does not have to realise that he is being monitored or deceived.

Thus, these computers should be installed behind configured firewall systems to allow incoming connections to the decoy computer, but limiting output connections.

In the second place, we have a strategy known as *building an entire network segment* which consists solely of decoy and deception equipment, all prepared to deceive the intruders (allowing access without too many difficulties). This is what is known as building a ***honeynet***.

The equipment in this segment will have to offer configured services to attract the attention of a whole community of intruders with the aim of recording all their movements through the decoy equipment.

The network can be real, with physical equipment and gateways, joined to the production network protected with traditional elements of monitoring, detection and prevention. The network can also be completely or partially virtual, combining the interaction between the physical equipment and gateways, with the use of emulation and virtualisation technologies. Here we find, for example, the use of virtual machines with virtualisation products of the Oracle/virtualbox, VMware/vSphere, Microsoft Hyper-V, etc. type, as well as the use of Docker-type software containers or alternative solutions. Also, emulation and virtualisation of network elements, with the use of software-defined networking and other traditional elements of cloud computing.

Instead of routing at the network level (i.e., at level 3 of the OSI reference model), the deception system could also function as a network bridge at the link level (i.e., level 2 of the OSI reference model), so that it could do without using IP addresses and reduce the chances of detection by the attackers.

All systems installed within a deception network should be systems of decoy and deception and as realistic as possible. That is, they should be real systems and applications, like those we can find in any production equipment.

Since we have not found simulated services in these systems of decoy and deception, all conclusions drawn from an investigation can be extrapolated directly to a real production network. Thus, all the deficiencies and weaknesses that are discovered within a deception network will be the same as in most organisations today.

The operation of the decoy and deception network is based on a single principle: all traffic that enters any of its equipment must be considered suspicious.

The monitoring process will be carried out through the detection mechanisms installed on the gateway, detecting attacks based on already known trends or statistics. However, the possibilities of investigating all the activity of a network of decoy and deception should also help detect unknown attacks.

Decoy and deception systems should be seen as research tools to improve the safety of production networks. They are a valuable solution, but require high dedication (time and storage resources).

Summary

Computer networks are exposed to cyber attacks so often that a large number of security requirements must be imposed for the protection of their resources.

Although the deficiencies of these systems can be checked by using conventional tools, they are not always corrected. In general, these weaknesses can cause a hole in network security and facilitate unauthorized or intrusive activities into the system.

In the same way that to ensure the physical safety of a building, motion detectors, surveillance cameras, log books, etc. are usually installed, a computer network needs equivalent components in the digital world to collect and create intrusion scenarios, process alerts and prevent intrusive activities.

In this didactic module, we have presented intrusion detection systems (IDS), the aim of which is precisely to provide these complementary elements to traditional security mechanisms and to be able to offer additional capabilities to warn and guide network administrators when there are attacks and computer intrusions.

We have also seen additional functionalities to facilitate normalisation, consolidation and correspondence of events collected by heterogeneous detection sensors. Some of these features have been presented in detail and we have made a first approach to Snort, an intruder detection system, based on open source and offered to the community of network administrators as a free software tool. Snort's main goal is to help network administrators continuously monitor network traffic in search of intrusion attempts or misuse. Finally, we have completed the module with a quick presentation on deception systems and techniques, as a complementary technology to learn from the offensive actions of the intruder.

Glossary

attack Aggression to the security of a system resulting from an intentional and deliberate act that violates its security policy.

bug Programming defect that can trigger a security deficiency.

CERT See ***Computer emergency response team***.

common vulnerabilities and exposures Public standard for the identification of vulnerabilities. It associates a unique identifier to each different vulnerability.
acronym **CVE**

common vulnerability scoring system Common framework for the evaluation of the criticality of vulnerabilities.
acronym **CVSS**

computer emergency response team Team of responses to computer emergencies; one of its main tasks is vulnerability management.
acronym **CERT**

computer security incident response team Computer security incident response team; one of its main tasks is vulnerability management.
acronym **CSIRT**

CSIRT See ***Computer security incident response team***.

CVE See ***Common vulnerabilities and exposures***.

CVSS See ***Common vulnerability scoring system***.

DDoS See **distributed denial of service**.

denial of service Attack that attempts to saturate the victim's resources, such as the memory or computing and processing capacity.
acronym **DoS**

distributed denial of service Denial of service that takes place from various connection points.
acronym **DDoS**

DoS See **denial of service**.

exploit Program or script that allows one or more vulnerabilities to be exploited; that is, a program that allows an attack to be made by taking advantage of the vulnerability.

exploration of ports Technique used to identify the services offered by a particular system or piece of equipment.

malware Program with malicious purposes.

risk A loss expectation expressed as the probability that a specific threat will exploit a particular vulnerability with especially harmful results.

rootkit Program that allows privileged access to a computer and manages to hide its presence from the administrator. It usually uses several vulnerabilities to install itself and achieve its purpose.

security policy Set of rules and practices that define and regulate the security services of an organisation or system with the purpose of protecting its critical and sensitive resources. In other words, it is the declaration of what is allowed and what is not.

security vulnerability A failure or weakness in the design, implementation, operation or management of a system, which can be exploited to violate its security policy.

sniffer Application that intercepts all the information that passes through the network interface to which it is associated.

threat Potential breach of security that exists on the basis of circumstances, capabilities, actions or events that may cause a security breach or some damage to the system.

Trojan horse A program, apparently harmless, that contains an attack on an uncorrected vulnerability.
syn . **trojan**

vulnerability scanner An application that allows checking whether a system is vulnerable to a set of security deficiencies.

zero-day vulnerability Vulnerability that, at the time of being exploited, is not previously known.

Bibliography

Beale, J.; Foster, J. C.; Posluns J.; Caswell, B. (2003). *Snort 2.0 Intrusion Detection*. Oxford: Syngress Publishing.

Garcia-Alfaro, J. (2004). "Mecanismos para la detección de ataques e intrusiones". In: Herrera, J.; Garcia-Alfaro, J.; Perramon, X. *Security in computer networks*. Barcelona: Fundació Universitat Oberta de Catalunya.

Garcia-Alfaro, J. (2007). "Detección de ataques en red con Snort". In: Herrera, J.; Garcia-Alfaro, J.; Perramon, X. *Advanced Security Aspects in Networks*. Barcelona: Fundació Universitat Oberta de Catalunya.

Koziol, J. (2003). *Intrusion Detection with Snort*. Indianapolis: Sams Publishing.

Kruegel, C.; Valeur, F.; Vigna, G. (2004). *Intrusion Detection and Correlation: Challenges and Solutions*. Springer-Verlag.

Northcutt, S.; Novak, J. (2002). *The Network Intrusion Detection* (3rd ed.). New Riders.

Miller, D. R; Harris, S.; Harper, A. A.; Vandyke, S.; Blask C. (2011). *Security Information and Event Management (SIEM)*. Mc Graw Hill.

Proctor, P. E. (2001). *The practical intrusion detection handbook*. New Jersey: Prentice-Hall.

Rehman, R. (2003). *Intrusion Detection Systems with Snort. Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID*. New Jersey: Prentice Hall PTR.