Misuse detection systems

Contents

- Overview of signature based IDS
- SNORT
- SNORT rules

Overview of signature based IDS

- Signature-based detection is based on the premise that abnormal or malicious network traffic is different from normal traffic.
- Because of that, it is possible to create an attack signature that matches the particular abnormal traffic pattern.

Overview of signature based IDS

- The majority of commercial IDS are signature based, or their main components are signature based.
- Modern commercial IDS are getting more and more based on distributed architecture with network nodes
  - the sensors reside on mission critical hosts instead of being dedicated to intrusion detection only – resolves for encrypted traffic monitoring, for example.

Overview of signature based IDS

- Examples of commercial IDS:
  - Network Flight Recorder (NFR)
  - Dragon
  - RealSecure (ISS)
  - NetRanger (Cisco)
  - Net Prowler
  - BlackICE
  - Centrax
  - Etc.

SNORT

- SNORT is an open-source network IDS.
- It can also be used as a “packet sniffer” and packet logger.
- When Snort is run in the sniffer mode, it displays the contents of every packet traversing the line directly to the console.
- It can display packet headers, as well as packet payloads.

SNORT

- NIDS mode is similar to the sniffer mode, in that it collects every packet it encounters.
- Instead of simply copying the data to a file or displaying it to a monitor, Snort inspects each packet and determines whether it is benign or malicious in nature.
- Snort then sends alerts when it finds suspicious-looking traffic.

SNORT

- If tuned properly, Snort is capable of monitoring every packet in a fully saturated 100Mb/s network.
- Snort begins to experience packet loss around the 200-300Mb level, and cannot be run at traffic levels higher than 500Mb.
- Thus it is not suitable for use in modern gigabit networks.

SNORT

- With Snort, a malicious traffic signature is used to create a rule that is loaded into the Snort detection engine.
- The detection engine is the primary component of Snort and is responsible for signature matching.
- SNORT can also use heuristics to detect malicious traffic that has no signature.

SNORT

- Three groups of Snort rules:
  - Suspicious packet headers detection
  - Suspicious packet payload detection
  - Specific protocol elements control
- Initial SNORT installation includes a number of predefined rules, intended to be generic for all networks.
- Then, a set of rules specific for the network in question is written.

SNORT

- Suspicious packet headers detection
  - Example:
    - The ICMP (Internet Control Message Protocol) ping used by the NMAP tool (to determine whether a host at a particular IP address is running) has a specific signature.
    - It sets the ICMP type field to 8 and has an empty data payload.

SNORT

- Suspicious packet headers detection (cont.)
  - Example (cont.):
    - This NMAP ping signature is different than a ping issued directly from a Windows or Unix operating system.
    - Since NMAP has a unique-looking ping, it is possible to create a rule that triggers whenever traffic matching this signature hits the protected network.

SNORT

- Suspicious packet headers detection (cont.)
  - Example (cont.)
    - The Snort rule for this is the following:
      - alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg: “ICMP PING NMAP"; dsize: 0; itype: 8;)
    - Generate an alert for ANY ICMP traffic that originates outside the protected network AND has an empty data payload AND has the ICMP type field set to 8.

SNORT

- Suspicious packet payload detection
  - Example:
    - alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg: "DOS SMBdie attack"; flags: A+; content: “|57724c6568004231334257 7a|”;)
    - This rule states that an alert should be generated if any TCP traffic that contains “|57724c6568004231334257 7a|” in the payload is found to be headed from outside the protected network to a computer running the Server Message Block (SMB) service.

SNORT

- Suspicious packet payload detection (cont.)
  - Example (cont.)
    - This payload means a buffer overflow in the Windows protocol, which would bring down the target host.
    - This rule triggers only when this payload is aimed at a computer running Netbios (TCP port 139) from the outside.
    - The rule is made specific to Netbios sessions to reduce the possibility of a false positive alert.

SNORT

- Specific protocol elements control
  - For accuracy and performance reasons, Snort signatures can be specific to one element of a particular protocol.
  - Example:
    - alert tcp $EXTERNAL_NET any -> $HTTP-SERVERS $HTTP-PORTS

      (msg:"WEB-IIS ISAPI .ida attempt"; uricontent: ”.ida?” ; nocase; dsize:>239; flags:A+;)
  - The .ida extension is a rarely used component of Microsoft's IIS Indexing Service.

SNORT

- Specific protocol elements control (cont.)
  - Example (cont.)
    - This rule states that any network traffic coming from the external network that is intended for the Web servers of the protected network that has .ida? in the URL creates an alert.
    - The .ida extension was found to have a serious remote buffer overflow that could result in remote control of the Web server.
    - This type of rule is more efficient because it searches URL content only, instead of the entire payload.

SNORT

- Custom rules
  - Although generic rules provide some intrusion monitoring coverage, a higher degree of rule granularity is the most effective means of increasing coverage.
  - The capability of writing custom rules is a very desirable feature of Snort.
  - The rule syntax is fairly simple and thus custom rule writing is relatively easy for those who possess solid knowledge of their own network.

SNORT

- Custom rules (cont.)
  - Example: writing a rule that alerts whenever a suspicious traffic directed towards the SSH server arrives from a particular URL.
    - alert tcp 192.168.1.1 any -> $HOME_NET 22

      (msg: "suspicious host SSH traffic";)
- Rules can be written to match any traffic signature or payload.

SNORT

- Detecting suspicious traffic via heuristics
  - Signature matching is a highly effective means for detecting known suspicious traffic.
  - However, signature matching is not 100% accurate.
  - There are situations where traffic is harmful but has no distinguishable signature.
  - Statistical Packet Anomaly Detection Engine (SPADE) module detects suspicious traffic that matches no signature.

SNORT

- Detecting suspicious traffic via heuristics (cont.)
  - SPADE works by detecting bad traffic through heuristic pattern matching.
  - SPADE observes network traffic and constructs a table that describes the normal traffic on the protected network.
  - The table contains data about the types of packets and the source and destination addresses.

SNORT

- Detecting suspicious traffic via heuristics (cont.)
  - After the table has reached a significant size, each packet that SPADE picks up is assigned a number based on the frequency in which it occurs in the table.
  - Packets that are rare for the protected network are assigned a higher number.

SNORT

- Detecting suspicious traffic via heuristics (cont.)
  - When a configured threshold is reached, an alert is generated.
  - A typical anomaly detection system with learning.

SNORT

- Detecting suspicious traffic via heuristics (cont.)
  - Example: protecting a web server by means of SPADE
    - We deploy Snort with SPADE enabled on a network segment that leads out of the Internet.
    - SPADE builds a table for incoming traffic-mostly TCP connections into ports 80 and 443.
    - After the table is built, TCP requests on ports 80 and 443 are considered "normal traffic" and assigned low numbers.

SNORT

- Detecting suspicious traffic via heuristics (cont.)
  - Example: protecting a web server by means of SPADE (cont.)
    - If an attacker were to probe the Web server looking for services on ports other than 80 and 443, SPADE would assign a high number to this traffic because it would be rare and unusual for this particular server.
    - If enough attempts to unusual ports are made in a predefined threshold, SPADE generates an alert.

SNORT

- Detecting suspicious traffic via heuristics (cont.)
  - This mode of operation is effective in detecting reconnaissance measures by attackers, who often probe ports slowly in an attempt to get lost in the background noise.
  - SPADE is capable of recognising the situation when an attacker is using multiple source addresses in an attempt to evade an IDS.

SNORT

- Detecting suspicious traffic via heuristics (cont.)
  - Distributed Denial of Service (DDoS) attacks, where many compromised hosts flood a host with so many requests that legitimate users cannot reach the server, are detected by SPADE as well.

SNORT

- Gathering intrusion data
  - A powerful feature of Snort is related to its capability of gathering data.
  - Many commercial IDS require the operator to specify in advance for which rules data should be kept.
  - It is very difficult to know in advance the attacks at which the protected network will be exposed to.

SNORT

- Gathering intrusion data (cont.)
  - The only solution is to save every payload that corresponds to suspicious traffic.
  - Snort does exactly this – it logs all payloads when possible.
  - Consequence: a lot of memory is needed.

SNORT

- Assessing threats
  - Payload data can help the operator to determine whether an attack is being perpetrated by a human or not.
  - If it ends up that a human is behind the attack, one might be able to use payload data to determine the attacker's skill level.

SNORT

- Preprocessing the data to be inspected
  - Snort has a class of plug-ins, known as preprocessors, that interact with data before the detection engine processes them.
  - Three functional groups of preprocessors:
    - Data Normalization
    - Protocol Analysis
    - Non-Signature-Matching Detection.

SNORT

- Preprocessing the data to be inspected (cont.)
  - Data normalization
    - New methods of attack and IDS evasion are constantly evolving that Snort detection engine either does not detect or does not detect efficiently.
    - Preprocessors are added to the Snort architecture to normalize data so that the detection engine can properly interpret them.

SNORT

- Preprocessing the data to be inspected (cont.)
  - Data normalization (cont.)
    - Example:
      - The Fnord preprocessor detects an IDS evasion technique borrowed from virus creators - polymorphism.
      - In an effort to defeat a signature-matching engine of an Antivirus, a virus's code randomly changes and mutates.
      - This is known as a polymorphic virus.

SNORT

- Preprocessing the data to be inspected (cont.)
  - Data normalization (cont.)
    - Example (cont.)
      - The same technique has been applied to exploits.
      - The shell code has been rendered polymorphic.
      - The Fnord preprocessor can detect mutated NOP sequences, which are a series of no-operation instructions in machine code that are used to exploit a buffer overflow.

SNORT

- Preprocessing the data to be inspected (cont.)
  - Protocol analysis
    - The SNORT detection engine has a short list of protocols that it can interpret.
    - Others, including some protocols that are heavily used over public networks, cannot be interpreted.

SNORT

- Preprocessing the data to be inspected (cont.)
  - Protocol analysis (cont.)
    - This has lead to a class of protocol preprocessors that aid in detecting protocol abuses.
    - Example: ASN1_decode, which detects inconsistencies in the Abstract Syntax Notation number one protocol.

SNORT

- Preprocessing the data to be inspected (cont.)
  - Protocol analysis (cont.)
    - Higher level protocols, such as SNMP, LDAP, and SSL, rely on ASN.1.
    - The capability to detect misuse of the ASN.1 protocol is necessary to monitor for these types of attacks.

SNORT

- Preprocessing the data to be inspected (cont.)
  - Non-Signature-Matching Detection
    - Some types of malicious traffic do not have a discernable signature.
    - This class of preprocessor uses methods other than signature matching to detect suspicious traffic.
    - Examples:
      - Reconnaissance attacks (portscanning, etc.)
      - Some DoS attacks (where traffic is normal, but consumes bandwidth and CPU time in such a way that the service in question becomes unavailable.

SNORT

- Preprocessing the data to be inspected (cont.)
  - Non-Signature-Matching Detection (cont.)
    - Corresponding preprocessors:
      - Portscan2
      - Stream4
    - Preprocessors such as Portscan2 and Stream4 can detect this class of traffic and some of the evasive techniques that attackers employ to keep IDS from discovering it.

SNORT

- Alerting
  - Snort's output plug-ins are the means Snort has to get intrusion data from the detection engine to the operator.
  - Like its preprocessors, Snort's outputting functionality is modular and pluggable.
  - Different skill levels, network configurations, and personal preferences will dictate which outputting mechanism is appropriate for the particular environment.

SNORT

- Alerting (cont.)
  - Snort supports everything from a raw binary tcpdump output to various relational database outputs.
  - Snort's outputs are not intended to be human-readable.
  - They are logged in various formats that make intrusion data readily accessible to other applications or tools.

SNORT

- Alerting (cont.)
  - Outputting can be done in these formats:
    - syslog
    - tcpdump
    - Text log file
    - XML
    - Relational database
    - SMTP
    - Snort Unified

SNORT

- Alerting (cont.)
  - The Snort designers have left the presentation of data to previously written applications.
  - Snort supports every major relational database platform (Microsoft SQL Server, Oracle, MySQL, Postgre SQL, etc.)

SNORT

- Aggregating data
  - Outputting to an industry standard format such as syslog lets the operator aggregate data from many disparate security devices.
  - Most routers and firewalls support functionality to log to a syslog server.

SNORT

- Aggregating data (cont.)
  - It is possible to import logs from other devices into the Snort database structure with logsnorter.
  - It is convenient to have all logging and intrusion-related information in one easily secured location.
  - Aggregation via syslog is a simplified means of performing *event correlation*.

SNORT

- Aggregating data (cont.)
  - Event correlation
    - Event correlation is the act of associating occurrences of events as they happened at different devices on a system or across a range of systems.
    - Event correlation is critically important to intrusion detection.

SNORT

- Aggregating data (cont.)
  - Event correlation (cont.)
    - Attackers rarely compromise a single host and walk away.
    - Most attacks involve an attacker compromising a single host and then leveraging legitimate privileges to penetrate deeper within a protected network.

SNORT

- Logging with the Unified Format and Barnyard
  - Historically, the relational database output plug-in has been the limiting factor in how much bandwidth Snort could process.
  - With a database plug-in enabled, Snort was capable of processing about 40% of the bandwidth compared to logging with the fastest method, a tcpdump file.
  - Logging via a network rather than a local disk further exacerbated the problem.

SNORT

- Logging with the Unified Format and Barnyard (cont.)
  - Snort's developers decided to outsource the database logging to a new application – Barnyard, specifically designed for the task.
  - With Barnyard, Snort spools output data in the Snort Unified Format at the maximum speed it can write to disk.
  - After an alert is written to disk, the Snort daemon is finished handling the alert and can concentrate on processing new packets.

SNORT

- Logging with the Unified Format and Barnyard (cont.)
  - This frees up the resources that would have been used outputting to a database.
  - The binary data is parsed by Barnyard into the various formats that are fed into database plug-ins attached to Barnyard.

SNORT

- Logging with the Unified Format and Barnyard (cont.)
  - Barnyard runs as an entirely separate process that is independent of Snort.
  - Alerts can now be posted to a database without affecting Snort's capability of capturing and analyzing traffic.

SNORT

- Displaying alerts
  - Intrusion detection is not an automated process.
  - It requires a human to receive the alerts and react to them in a timely fashion.
  - Getting real-time alerts out of Snort and to the operator can be configured in many ways.

SNORT

- Displaying alerts (cont.)
  - The two primary means for alerting are
    - Real-time alerting with syslog and swatch
    - Analysis Console for Intrusion Databases (ACID).

SNORT

- Displaying alerts (cont.)
  - Swatch
    - Actively monitors a syslog file for preconfigured events and generates an alert when conditions are met.
    - Alerts can be sent via a number of means (e-mail, audible alarm, etc.)

SNORT

- Displaying alerts (cont.)
  - ACID
    - A Web application that reads intrusion data stored in a database and presents the data in a browser.
    - ACID presents Snort data in a human-readable format and includes functionality to perform complex searches.
    - Complex searches can be created with more than 30 different criteria to pinpoint events occurring in intrusion data.
    - This level of accuracy is necessary to quickly identify and eliminate false positives.

SNORT

- Displaying alerts (cont.)
  - ACID (cont.)
    - Can group alerts into logically functional categories.
    - Matches links to various common vulnerabilities and exposures (CVE) on the Internet.
    - CVE is a standardized classification of vulnerabilities and exposures, and a significant resource for identifying and understanding attacks.

SNORT

- Displaying alerts (cont.)
  - ACID (cont.)
    - ACID can distinguish multiple installations of Snort from each other, and process data from other security devices.
    - This makes ACID another option for event correlation through aggregation.

SNORT

- Displaying alerts (cont.)
  - ACID (cont.)
    - Includes a charting component that is used to create statistics and graphs.
    - It can be useful to chart how the threat to the protected network changes over time.
    - It can also be useful for accomplishing network management tasks of course regard IDS.

SNORT

- Prioritizing Alerts
  - An IDS needs to be able to categorize and prioritize alerts in an organized manner.
  - Not all alerts deserve the same attention and scrutiny.
  - Example: A simple ping is no cause for immediate alarm, but a remote exploit attempt against an unpatched server is.

SNORT

- Prioritizing Alerts (cont.)
  - Types of alerting in an IDS:
    - No prioritization
    - Hard-coded prioritization
    - Customizable prioritization

SNORT

- Prioritizing Alerts (cont.)
  - No Prioritization
    - In this system, all alerts have the same priority.
    - This makes sorting by severity impossible.
    - Any automatic emergency notification mechanism is rendered useless.
    - A very bad idea - not prioritizing alerts leaves the IDS analyst frustrated and ultimately disinterested in intrusion monitoring.

SNORT

- Prioritizing Alerts (cont.)
  - Hard-coded Prioritization
    - This is only marginally better than no prioritization.
    - Here the IDS designer has decided for the customer which alerts are important and which are not.
    - Usually, they are categorized as High, Medium, and Low.
    - Although this does allow the operator to sort and filter out less important alerts, such an approach to alerting is inadequate.

SNORT

- Prioritizing Alerts (cont.)
  - Hard-coded Prioritization (cont.)
    - In this system, it is possible to generate alerts regarding irrelevant events for the particular environment.
    - Example: an Apache Web server alert, categorized as a high risk by the IDS designer, is generated even if Apache is not installed at all.

SNORT

- Prioritizing Alerts (cont.)
  - Customizable Prioritization
    - The preferred way to group alerting data is by user-defined priorities.
    - Modern networks are modular and therefore unique, so customizable prioritization of alerts is necessary.
    - Alerts can be sorted based on the priorities of the customer, so alerts are generated according to the organization’s needs.
    - Having more than three severity levels for alerts is desirable.

SNORT

- Prioritizing Alerts (cont.)
  - Customizable Prioritization (cont.)
    - Snort supports customizable prioritization of alerts.
    - It has 32 predefined alert categories.
    - The severity level of each of the categories can be modified .
    - The operator can also add as many custom alert categories as required.
    - Like signatures, alert classification is performed via simple rules.

SNORT

- Prioritizing Alerts (cont.)
  - Customizable Prioritization (cont.)
    - A sample alert classification for Trojan traffic:
      - config classification: trojan-activity, A Network Trojan was detected, 1
      - This classification is for any type of detected trojan activity (Netbus, Back Orifice, SubSeven, etc.)
      - This rule grants the highest severity level when logging the alert, 1.

SNORT

- Prioritizing Alerts (cont.)
  - Customizable Prioritization (cont.)
    - A sample alert classification for Trojan traffic (cont.):
      - Any signature rule classified as trojan-activity would correspond to this classification rule.
      - This is done with the classtype identifier in the signature rule.
      - Example: The SubSeven signature rule has trojan-activity specified for the classtype:
        - » alert tcp $EXTERNAL_NET 27374 -> $HOME_NET any (msg:”BACKDOOR subseven 22”; flags: A+; content: “|0d0a5b52504c5d3030320d0a|” ; classtype : trojan-activity;)

SNORT

- Prioritizing Alerts (cont.)
  - Customizable Prioritization (cont.)
    - A sample alert classification for Trojan traffic (cont.):
      - If we were not concerned with SubSeven activity but we want to keep all other Trojans at alert priority 1, we can override the classification rule with a priority identifier.
      - The new rule is:
        - » alert tcp $EXTERNAL_NET 27374 -> $HOME_NET any (msg:"BACKDOOR subseven 22"; flags: A+; content: "|0d0a5b52504c5d3030320d0a|" ; classtype : trojan-activity; priority: 2;)

SNORT

- Prioritizing Alerts (cont.)
  - Customizable Prioritization (cont.)
    - Most reasonable alert prioritizing requirements can be handled by these means.
    - Alert prioritization would be of little use if alerts could not be delivered in a timely, organized manner.
    - Snort gives the operator the choice of 72 different alert outputting modules.

SNORT

- Prioritizing Alerts (cont.)
  - Customizable Prioritization (cont.)
    - Modules can be used to log intrusion data in any format from raw output to point-and-click GUIs.
    - The operator is not limited to using only one alerting output method.
    - It is possible to choose as many outputting methods as necessary.