Intrusion Detection Systems

Sireesha Dasaraju
CS526 – Advanced Internet Systems
University of Colorado Colorado Springs

Abstract

With the overwhelming increase of Internet enabled critical services geared towards individuals, companies and organizations, more and more of these entities are connecting to the commercialized Internet. As a result, it becomes extremely important that these entities take appropriate measures to protect their internal networks from malicious attacks through the unpoliced Internet. An Intrusion Detection System (*IDS)* plays an important role in dealing with the threats against the internal networks. This paper documents the need for and benefits of using an IDS and various techniques used for detecting an attack. And then progresses into an indepth study of a popular, open source IDS, Snort. The structure of Snort rules is examined in detail. Finally automatic signature generation and payload anomaly detection techniques are explored and a technique to integrate them into Snort is proposed.

1. Introduction

With the explosion of access to the Internet enabled services, a computer network belonging to an individual, company or an organization should be on guard to protect itself against different threats. Some of the threats the networks should watch out are worms, viruses, DDoS attacks and so on.

The first step towards protecting a single computer or a network of computers is setting up a firewall as a intemediary between the network and the internet. A properly configured firewall can help shield the network from the outside hacker attacks. But a firewall may not work in the following situations

- A new attack may compromise hosts before the firewall rules can be updated.
- Laptops may become compormised when offsite and then infect machines behind the firewall when they are connected to the network.
- Wireless access points may allow intruders into the network.

The next step in securing an internal computer network is to add an IDS. An IDS is an application that detects attacks against a computer or network and informs the administrator when the attacks occur.

2. IDS Basics

An Intrusion Detection System monitors network traffic and monitors for suspicious activity and alerts the system or network administrator. In some cases, the IDS may also respond to anomalous or malicious traffic by taking action such as blocking the user or source IP address from accessing the network.

2.1Benefits of using an IDS

A few benefits of using an IDS are listed below.

- **Detecting attacks** : An IDS can inform an administrator if a worm is attacking the network or if a computer system has been comprimised.
- **Enforcing policies** : An IDS can monitor an internal network for behavior that violates an organization's network security or acceptable user policies.
- **Providing an audit trail** : An IDS can provide an after-the-attack audit

trail for seeing how far an attacker got, and where it came from.

2.2IDS Detection Techniques

There are two different ways an IDS can detect attacks , **Signature Detection** and **Anomaly Detection.** Some IDS solutions use signature detection, some use anamoly detection and some use a combination of both.

2.2.1 Signature Detection

Signature-based threat detection scans traffic for a set of pre-defined attack patterns. The signatures are typically important bits and pieces of the attack that the IDS should look for in incoming network packets and flag as "bad" traffic.

Signatures have two main limitations. First, they are prone to false positives without extensive tuning. Second, they are not effective at detecting unknown attacks.

2.2.2 Anomaly Detection

Behavior-based anomaly detection compares a profile of all allowed application behavior to actual traffic. Any deviation from the profile is flagged as a potential attack. Behavior anomaly detection has the potential to detect attacks of all kind , including "unknown" attacks.

This technique also leads to high false positives.

2.3Types of IDS

There are two types of IDS, Network-based and Host-based.

2.3.1 Network-based IDS (NIDS)

A network-based IDS analyzes packets coming across a network connection for data that looks like its part of an attack. NIDS perform the following tasks:

- Analyze network traffic for attacks, using signature or anomaly detection (or both). Its network interface card (NIC) captures all network traffic that goes by its NIC, not just the traffic destined for the IDS system itself.
- Generate real time alerts to notify an administrator of an attack.
- Generate logs that can be used to analyze the attacks, typically after the attack has occurred.

2.3.2 Host-based IDS (HIDS)

A host based IDS differs from network based IDS, in that it only monitors for intrusions on te system it's running on. It performs the following tasks

- Analyze network traffic for attacks, using signature or anomaly detection (or both). Its network interface card (NIC) captures just the traffic destined for the IDS system itself.
- Examine system logs for unusual events, such as multiple invalid login attempts.
- Check the integrity of files on the system.

2.4IDS Performance

The performance of an IDS is measured interms of False Positive Rate, False Negativ Rate and Crossover Error Rate.

2.4.1False Positive Rate

The False Positive Rate is the frequency with which the IDS reports malicious activity in error. A false positive occurs when an IDS generates an alert on either

- Network traffic that looks like an attack to the IDS, but isn't an attack.
- A real attack that attack doesn't apply to the system being monitored.

False positives are a problem because they create alert noise that can hide a real attack. They are the nuisance reports that require investigation but lead to dead end. These are also sometimes called "Type 1 errors". Increasing the sensitivity of an IDS results in a higher false positive rate, while decreasing the sensitivity lowers the false positive rate.

2.4.2False Negative Rate

The False Negative Rate is the frequency with which the IDS fails to raise an alert when malicious activity actually occurs. These are the most dangerous types of errors, as they represent undected attacks on a system. These errors are also referred to as Type II erorrs. False negative rates change in an inverse proportion to a false positive rates, which means as the false positive rate increases, the false negative rate decreases and vice-versa.

2.4.3Crossover Error Rate

The Crossover Error Rate (CER) is often used to provide a baseline measure for comparision of Intrusion Detection Systems. As the sensitivity of systems may cause the false postive/negative rates to vary, it is critical to have some common measure that may be applied across the board. The CER for a system is determined by adjusting the system's sensitivity until the false positive rate and the false negative rate are equal. Then different IDSs may be run on the same network and measure the CER for each.

3. Snort IDS

Snort is an open sourced, network based IDS that uses signature detection.

3.1.1Reasons for using Snort

- Snort is an open source project and so is readily available and is free.
- Snort is passive, which leads it to monitor any system on the network with no configuration to the target computer.
- Snort is portable and fast.
- Snort is versatile and can be used as an IDS, IPS (intrusion prevention system), scrubber, Inline firewall.
- Snort is able to log to numerous databases include Oracle, Microsoft SQL Server, MySQL and PostGre SQL.
- Snort rule files are simple, easy to write and are effective.
- Snort is ported to every major operating system.
- Flexible and simple, Snort uses plugins for all of its functions so adding new functionality is easy.

3.1.2 Components of the Snort IDS

Snort is desinged to be composed of several components, with each of the components performing a specific task. These components are connected to each other as shown in the figure.

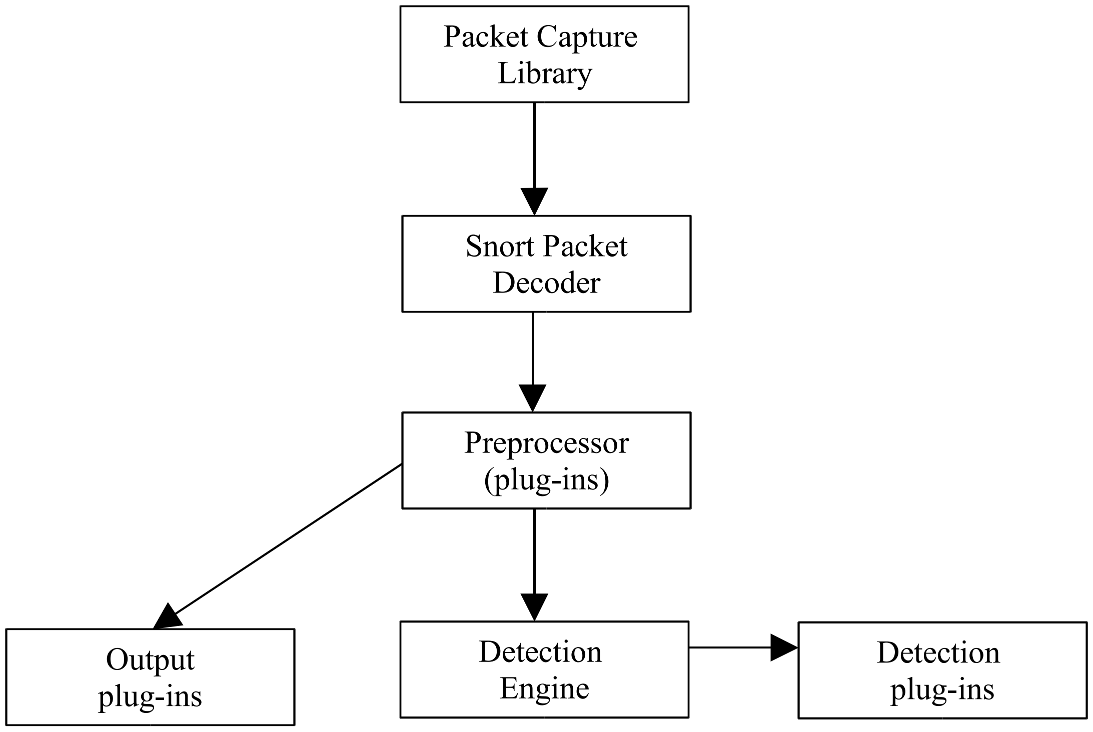


Figure 1 Components of a Snort IDS

When a network packet arrives at the network Snort it monitoring, the following sequence of events occur.

a. The packet capture library delivers the packets to Snort, via the network card. These are the unprocessed Data-Link Layer pockets.

b. The packet decoder takes the Layer 2 data sent over from the packet capture libray and takes it apart. First it decodes the Data Link frame, then the IP protocol, then the TCP or UDP packet.

c. The decoded data is then preprocessed, which performs a variety of transformations making the data easier for Snort to digest. Preprocessors can alert on, classify or drop a packet before sending it on to the more CPU-intensive detection engine.

d. The detection engine is the heart of Snort. It takes the information from the packet decoder and preprocessors and operates on it at the transport and application layers, comparing the contents of the packet to its rules-based detection plugin.

e. Finally if any of the rule conditions are met, an alert is generated and logged.

3.1.2 Snort Rules

Snort uses a simple, lightweight rules description language that is flexible and quite powerful. Each Snort rule should be completely contained on a single line.

```
alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86 a5|"; msg:
"mountd access";)
```

Snort rules are divided into 8 major categories :

- Low-level protocols (icmp, netbios, tcp udp)
- High-level protocols (http,ftp,dns,pop3,imap)
- Web server specific (web-attack, web-cgi, web-client)
- Exploit specific (shellcode, backdoor, exploit)
- Service impacting (dos, ddos)
- Policy specific (policy, info, misc,porn)
- Scanning and probing activities (scan, bad-traffic)
- Viruses, worms and other malware (virus)

There are two parts to a Snort Rule, a Rule header and a Rule Options.

3.1.2.1Snort Rule Header

The rule header contains the information that defines "who, what and where" of a packet, as well as what to do in the event that a packet with all the attributes indicated in the rule should show up. The header acts as a front-end filter that separates our traffic by using five key factors : source IP address, destination address, source port, destination port and protocol.

3.1.2.1.1Rule actions

Rule action is the first item in the rule. The rule action tells Snort what to do when it finds a packet that matches the rule criteria. There are five rule actions available for Snort.

- log : The log action logs the offending packets to the output logging that is setup during the Snort configuration.
- alert : The alert action will log the entry and post a notification when some event is associated with higher priority. This is the default action for most rules that come with Snort.
- pass : The pass action can ignore a matched packet and continue processing.
- activate/dynamic : These two actions act together. The activate action triggers an alert and executes what is specified by dynamic action. The dynamic action is associated with a rule that shouldn't run until another event is encountered.

3.1.2.1.2 Protocols

The next field in a rule is the protocol. Snort can analyze suspicious behavior of the protocols, IP, ICMP, TCP and UDP.

3.1.2.1.3Soure/Destination IPAddress/PortNumbers

The next field deals with the IP address and port information. The source and destination networks are identified in a rule that is of the form

(source network) (port) --> (destination network) (port)

The IP addresses in the rule are expressed using CIDR (Classless Inter Domain Routing) notation. Using CIDR notation, a single or a range of IP addresses/ports can be specified in the rule. Negation operator !, can be used to exclude a specific IP address or port.

log udp any any -> 192.168.1.0/24 1:1024 *log udp traffic coming from any port and destination ports ranging from 1 to 1024*
log tcp any any -> 192.168.1.0/24 :6000 *log tcp traffic from any port going to ports less than or equal to 6000*
log tcp any :1024 -> 192.168.1.0/24 500: *log tcp traffic from priveleged ports less than or equal to 1024 going to ports greater than or equal to 500*
alert tcp !192.168.1.0/24 any -> 192.168.1.0/24 111 (content: "|00 01 86 a5|"; msg: "external mountd access";)

3.1.2.1.4 The Direction Operator

The direction opeator -> indicates the orientation or direction of the traffic that the rule applies to. The IP address and port numbers on the left side of the direction operator is considered to be the traffic coming from the source host, and the address and port information on the right side of the operator is the destination host. There is also bidirectional opeator, which is indicated with a "<>" symbol. This it to indicate Snort to consider the address/port pairs in either the source or destination orientation. This is useful for recording/analyzing both sides of a conversation.

```
log !192.168.1.0/24 any <> 192.168.1.0/24 23
```

3.1.2.2Snort Rule Options (Rule Body)

The Rule option part of the Snort's rule should follow a specific structure outlined as below :

- The option section of the rule is always wrapped by one set of parenthesis.
- Body options (keywords, instructions, tests and commands) are written inside the parantheses.
- Each body option is separated by a semicolon.
- Each body option confirms to the format, item: "value";
- The entire line is terminated with a semicolon.

3.1.2.2.1 The "content" option

The content option allows the user to set rules that search for specific content in the packet payload and trigger response based on that data. The search is case sensitive. The data that is specified for search can be ASCII text or binary data or a combination of both. The binary data is generally enclosed within the pipe ("|") character and represented as bytecode.

```
alert tcp any any -> 192.168.1.0/24 143 (content: "|90C8 C0FF FFFF|/
bin/sh"; msg: "IMAP buffer overflow!";)
```

3.1.2.2.2 The "depth" option

The depth option specifies how many bytes into a packet the Snort processor should examine before moving on to the next rule. The main reason for using the depth option is to restrict the search to the most likely places where a match is found, without wasting valuable processor resources to search the entire packet.

3.1.2.2.3The "nocase" option

The nocase option indicates that the case of the characters submitted for searching should be ignored.

3.1.2.2.4The "offset" option

The offset option indicates that the search should skip the number of bytes specified.

3.1.2.2.5 The "Uniform Resource Identifier (URI)" option

The uricontent is similar to the content, but with this option the search is limited to the URI in the payload of the packet.

3.1.2.2.6 The "sid" option

The "sid" option is used to uniquely identify a snort rule.

3.1.2.2.7The "priority" option

The "priority" option is used to associate a priority with a rule. The lower the priority number, the higher the risk posed by the attack that tripped the rule.

3.1.2.2.8 The "classtype" option

The "classtype" option is used to organize rules into major groups.

3.1.2.2.9 The "rev" option

The "rev" option is used to associate revisions with the rules.

3.1.2.2.10 The "mesg" option

The "mesg" option creates a customized output message that can be included with any logs, alerts and data dumps processed by the detection engine.

3.1.2.2.11 The "reference" option

The "reference" option is used to point to an external resource from the rule, for eg., a Web-based resource.

4. Automatic Signature Generation

Identifying new intrusions and developing effective signatures that detect them is essential for protecting computer networks. This section summarizes the Nemean system that is used for automatic generation of signatures from honeynet packet traces. The building blocks of this architecture are transport and service normalization, intrusion profile clustering and automata learning that generates connection and session aware signatures.

Generation and maintenance of signatures is a difficult task because of the competing requirements. On one hand signatures should be specific so that they identify only the characteristics of specific attack profiles. The lack of specificity leads to false alarms, On the other hand the signatures should be general so that they match variants of specific attack profiles. Nemean aims to create signatures that result in lower false alarm rates by balancing specificity and generality. This balance is achieved by including the semantics awarenesss, i.e., the knowledge of session-layer and application-layer protocol semantic in the process of signature generation.

Nemean architecute consists of two main components, a data abstraction component and a signature generation component. The input Nemean is a packet trace collected from a honeynet.

The Data Abstraction Component (DAC) aggregates and transforms the packet trace into a well-defined data structure suitable for clustering by a generic clustering module without specific knowledge of the transport protocol or application-level semantics. These aggregation units are called Semi Structured Session Trees (SSTs). The aggregation step of the DAC groups packet data between two hosts into

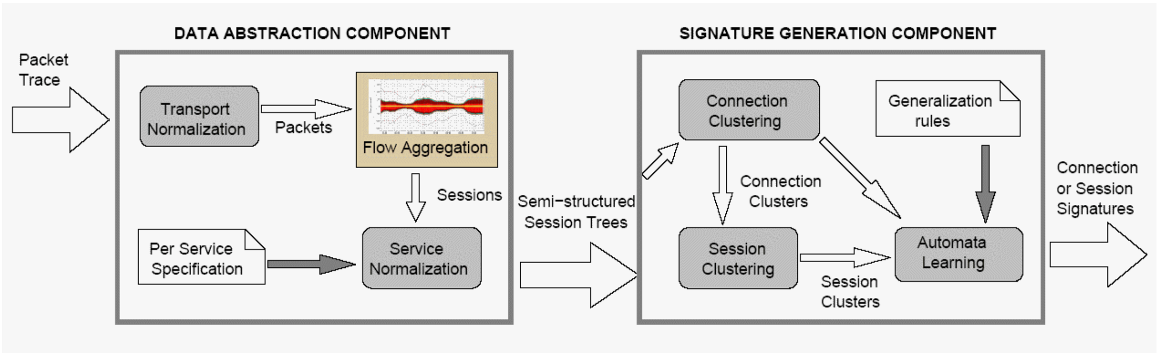


Figure 2: Components and data flow description of the Nemean architecture.

An Architecture for Generating Semantics-Aware Signatures
Proceedings of Usenix Security Symposium 2005

sessions. The normalized packet data is first composed and stored as flows. If a same pair of hosts and ports communicate again, the flow is converted into a connection. While a flow is composed of packets, connections are a collection of request-response elements. A flow might be expired if a flow has been inactive for a time period greater than a user defined timeout. A session is a sequence of connection between the same host pairs.

Before the above sessions are clustered, service specific information in the sessions is normalized to make the clustering independent of the type of service and to enable generation of a more compact signature set. These normalized session are XML encoded to create Semi Structured Session Trees (SST). Weights are assigned to the elements of the SST to highlight the most important attributes like the URL in an HTTP request and deemphasize the less important attributes, such as encrypted fileds and proxy-cache headers in HTTP packets.

The clustering module groups sessions and connections with similar attack profiles according to a similarity metric. The sessions grouped together will correspond to a single attack type or variants of a well-known attack while disparate clusters represent distinct attackts or attack variants that differ significantly from some original attack. Clusters are created based on two properties, the data that corresponds to an attack and its variants should be measurably similar and the data corresponding to different attacks must be measurably dissimilar. Properties of normal traffic vary so greatly that effective clusters cannot be created from the normal traffic.

Once the clusters are created, an automata learning module constructs an attack signature from a cluster of sessions. Clusters that contain many non-uniform sessions indicate minor changes made to an attack either to mask an existing attack or to create a variant of an existing attack. The signature generation component generalizes these minor changes to produce a signature that is resilient to evasion attempts. Generalization enable signatures to match malicious sequences that were not observed during the automata training.

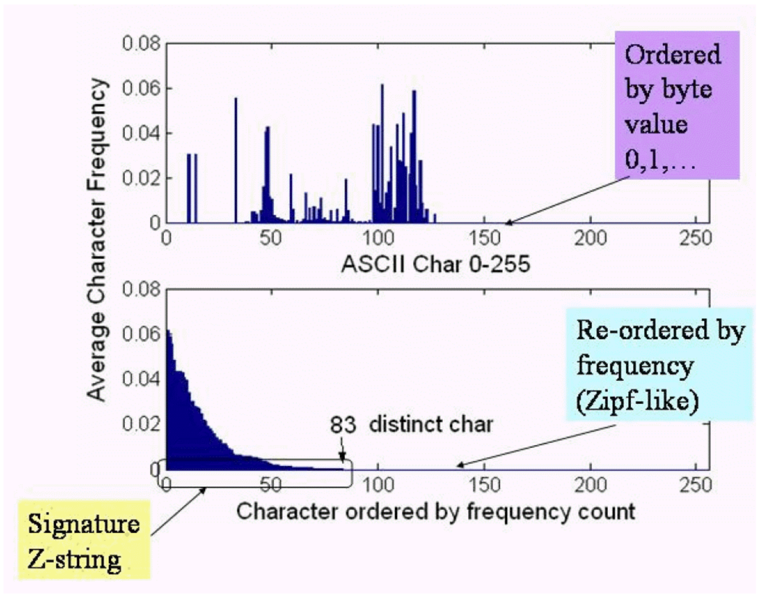
5. Anomalous Payload-based Network Intrusion Detection

There are many IDS systems available that are primarily signature-based detectors. Although these are effective at detective known intrusion attempts and exploits, they fail to recognize new attacks and carefully crafted variants of old attacks. This is the main drawback of a signatrue based IDS. An anomaly based IDS, first models normal or expected behavior in a system and detect deviations of interest that may indicate a security breach or an attempted attack. And so an anomaly based IDS is able to detect a new attack that as easily as it would an existing attack.

An anomaly based IDS operates in two different phases, a learning phase and a anomaly detection phase. During the learning phase, the IDS learns a model or profile of the expected payload delivered to a service during normal operation of a system. Each payload is analyzed to produce a byte frequency distribution of those payloads, which serves as a model for normal payloads. After this centroid model is computed during the learning phase, an anomaly detection phase begins. The anomaly dectector captures incoming payloads and tests the payload for its two statistical distributions. Any new payload found to be too distant from the normal expected payload is deemed anomalous and an alert is generated.

Mimicry attacks are possible if the attacker has access to the same information as the victim to replicate normal behavior. In case of the application payload, attackers would not know the distribution of the normal flow to their intended victim.

Consider the string of bytes corresponding to the sorted, rank ordered byte frequency of a model. The following figure shows the view of this process.



The frequency distribution of payloads of length 185 is plotted in the top graph. The lower graph represents the same information by the plot is reordered to the rank ordering of the distribution. Here, the first bar in the lower plot is the frequency of the most frequently appearing ASCII character. The second bar is likewise the second most frequent and so on. This rank order distribution follows a Zipf-like distribution (an exponential function or a power law where there are few values appearing many times, and a large number of values appearing very infrequently). The rank order distribution also defines a "Z-string". Thebyte values ordered from most frequent to least frequent serves aa a representative of the entire distribution. The rank ordered byte value distribution of the new payload deemed anomalous and can server as a simple representation of a "new worm signature". If an anamalous payload appears at a site and its rank ordered byte distribution matches another site, it is evidence that a worm has appeared.

6. Integrating Automatic Signature generation with Snort

Snort supports extensions to its functionality through pluggable processors/preprocessors. A component that can automatically generate signatures as metioned in the section #4, can be easily plugged into Snort as a "Signature Generation" processor. The incoming traffic will be first directed the detection engine. The detection engine will generate an alert if the traffic matches any of the existing signatures, there by detecting attack if it is already known. If the traffic packet generated an alert, then it will not be passed onto the new processor since the attack is already represented by an existing signature. But if no alert was generated, the traffic packet is directed to the new signature generation processor. This new processor will analyze the incoming traffic using the process outlined in Section 4 and generates new signatures. These exisitng singature store of Snort can be dynamically updated with these new signatures.

7 Integrating Anomalous Payload-based Network Intrusion Detection

Snort is primarily a signature based IDS. But can be extended to be a hybrid IDS by including a anomaly detection component. The new feature can again be plugged in as a new processor. The new processor, a "Anomaly Detector" can detect attacks using the process outlined in Section 5. The incoming traffic packet will be first directed to the singature based detection engine. This traffic packet may or may not generate an alert based on the comparision with existing signatures. If no alert is generated by signature detection, the traffic will then be directed to the Anamoly Detection processor. Thus the hybrid system will be able to capture both a reoccurance of an existing attack and first occurance of a new attack.

8 Conclusion

An Intrusion Detection System protects a single computer or a network of computers from malicious traffic by analyzing the incoming traffic and generating real time alerts. High sensitivity of an IDS can lead to high false positive rates, but lowering the senstivity can lead to high false negatives, so a system administrator should try to achieve a balance when configuring an IDS. Snort provides signature detection for a network of computers via very simple alert rules. Snort can be extended via pluggable processors to include the automatic signature component. Snort can be extended to be a hybrid IDS by including an anamoly detection component.

Bibiliography

1. Intrusion Detection. http://www.itarchitect.com/article/NMG20001130S0007

2. Types of internet threats.

   http://www.integratedsoftwaresolutions.ca/Documents/SecurityAlerts/Types_of_Threats.htm

3. Inroduction to Intrusion Detection Systems (IDS)

   http://netsecurity.about.com/cs/hackertools/a/aa030504.htm

4. What is signature detection?

   http://www.imperva.com/application_defense_center/glossary/signature_detection.html

5. Evaluating and tuning an intrusion-detection system

   http://searchsecurity.techtarget.com/tip/1,289483,sid14_gci918619,00.html?track=IDSLG

6. A look into IDS/Snort.

   http://www.antionline.com/showthread.php?s=&threadid=253920

7. Writing Snort rules http://packetstormsecurity.nl/papers/IDS/snort_rules.htm

8. Anomalous Payload-based Network Intrusion Detection by Ke Wang, Salvatore J. Stolfo.

9. An architecture for Generating Semantics-Aware Signatures by Vinod Yegneswaran, Jonathon T. Giffin, Paul Barford, Somesh Jha.