Signature-based Intrusion Detection

Boriana Ditcheva and Lisa Fowler
University of North Carolina at Chapel Hill
February 16 & 22, 2005

Detecting Attacks

- Anomaly-based Detection
- Signature-based (Misuse) Detection

- Host-based
- Network-based
  - Active/Passive

Anomaly-based detection

- Central idea: "abnormal" = "suspicious"
- Automatically learns
- Detects novel attacks (and its variations)
- Can be left to run unattended

- Requires a notion and definition of "normal"
- Susceptible to false negatives
  - Unusual is not necessarily illicit/malicious
  - Usual is not necessarily benign
    - e.g. attacks that manifest slowly
- Computation intensive

Signature-based Detection

- Looks for specific and explicit indications of attacks
  - Identified by raw byte sequences (strings), protocol type, port numbers, etc.
- Low false positives
  - "Knows for a fact" what is suspicious, what is normal
    - Detects only behavior that was previously defined to be suspicious
  - Can have *tight* signatures (high confidence)
- Simple and efficient process
- Easy to share
  - Repositories of signatures

Problems

- System must be trained
  - Requires time-consuming manual identification and specification of each new attack
  - Often requires 'expert' knowledge
  - Cannot detect novel attacks on its own
- False negatives
  - May not detect simple variations
  - Unless previously detected and identified…
- False positives
  - May detect failed attacks
  - *Loose* signatures (low confidence)
  - Poorly configured systems

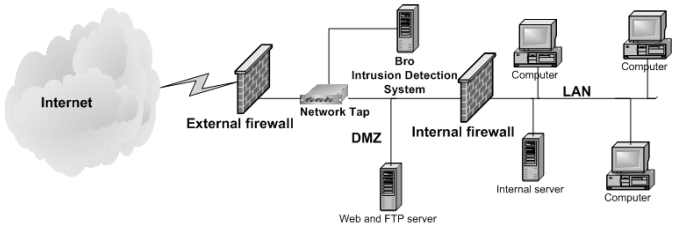
Signature-based Detection

- Goal:
  - Find a pattern or *signature* that can allow for the detection of a specific attack
    - Think about virus detection…
  - Be narrow to be more precise (reduce false negatives)
  - Be flexible to cover as many of the variants as possible while minimizing false positives

IDS Placement

- Outside firewall
  - Detects all attacks directed at your network
  - Detects more events
  - Generates more logs
- Inside firewall
  - Only detects what the firewall lets in
  - Less state information

IDS Placement



Typical placement of an IDS system (in this example, Bro
For more info, see http://www.netoptics.com/products/pdf/Taps-and-IDSs.pdf

http://bro-ids.org/Bro-quick-start/Network-Tap.html

Making a Signature

- DIY (Manual)
  - Become a “security officer”
  - Know detailed information regarding the exploit
  - Generate the signature by manual inspection
  - Can generate false positives or false negatives

Making a Signature++

- Automated
  - If we can extract or isolate suspicious network data, can conceive of a system that can aggregate the data and generate signatures

Example Signature

Trin00
http://www.snort.org/snort-db/sid.html?sid=223

GEN:SID	1:223
Message	DDOS Trin00 Daemon to Master PONG message detected
Rule	alert udp $EXTERNAL_NET any -> $HOME_NET 31335 (msg:"DDOS Trin00 Daemon to Master PONG message detected"; content:"PONG"; reference:arachnids,187; classtype:attempted-recon; sid:223; rev:3;)

All Snort signature examples from http://snort.org

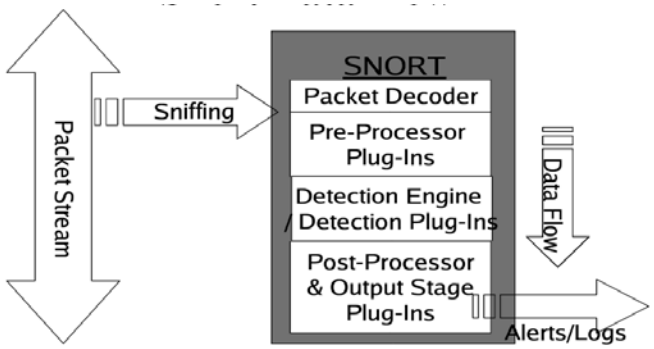
SNORT

- Lightweight signature-based intrusion detection system
- Only 100 kilobytes in compressed source distribution
- Don’t need sophisticated training to use like with other commercial NIDSs
- Configurable (Easy rules language, many logging/alerting options)
- Free

Snort's architecture

- Packet Decoder
- Detection Engine
- Logging/Alerting Subsystem
  - These subsystems ride on top of the libpcap promiscuous packet sniffing library.

Snort's architecture



Packet Decoder

- Organized around the layers of the protocol stack present in the supported data-link and TCP/IP protocol definitions.
- Sets pointers into the packet data for later access and analysis by the detection engine.

Detection Engine

- Uses comparison to predetermined rules (*to be discussed in a minute*) to decide whether a packet should be flagged or not.
- Maintains detection rules in a two dimensional linked list of Chain Headers and Chain Options.
- First rule that matches a decoded packet triggers the specified action and returns.

Rule Chain Structure

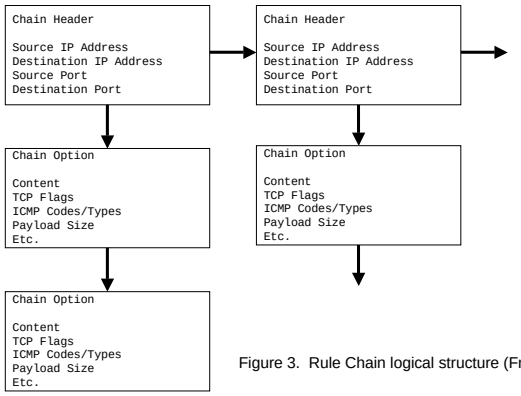


Figure 3. Rule Chain logical structure (From [4])

Logging/Alerting Subsystem

- Logging options:
  - Log packets in their decoded human readable format to an IP-based directory structure
  - Log packets in tcpdump binary format to single log file (*much faster*)
  - Do not log

Logging/Alerting Subsystem

- Alert options:
  - Alerts sent to syslog
  - Alerts logged to specified alert text file
    - Full alerting: write the alert message and packet header info through the transport layer protocol
    - Fast alerting: write condensed subset of the header info
  - Alerts sent as WinPopup messages
  - Disable alerting

Snort Rules

- Snort can take three base actions when it finds a matching packet:
  - Pass (drop the packet)
  - Log (write full packet to logging routine)
  - Alert (generates notification as specified by user)

Snort Rules

```
alert tcp !10.1.1.0/24 any -> 10.1.1.0/24 6000:6010 (msg: "X traffic";)
```

- Header Features:
- Look at uni- or bi-directional traffic
- IP addresses
  - negation, CIDR ranges
- TCP/UDP ports
  - Negation, ranges, greater than/less than

Option Fields

1. content – search packet payload for specified item
2. flags – test TCP flags
3. ttl – check IP ttl field
4. itype – match on ICMP type field
5. icode – match on ICMP code field
6. minfrag – set threshold for IP fragment size
7. id – test IP header for specified value

Option Fields

8. ack – TCP ack number
9. seq – TCP seq number
10. logto – log packets matching this rule to this specified filename
11. dsize – packet payload
12. offset – begin content search at this offset
13. depth – search content to this byte depth in file
14. msg – message to be sent when packet generates event

Example Signature

Trin00
http://www.snort.org/snort-db/sid.html?sid=223

GEN:SID	1:223
Message	DDOS Trin00 Daemon to Master PONG message detected
Rule	alert udp $EXTERNAL_NET any -> $HOME_NET 31335 (msg:"DDOS Trin00 Daemon to Master PONG message detected"; content:"PONG"; reference:arachnids,187; classtype:attempted-recon; sid:223; rev:3;)

All Snort signature examples from http://snort.org

Stacheldraht

- One attack may have many different signatures

SID	Name	CVE ID
224	DDOS Stacheldraht server spoof	
225	DDOS Stacheldraht gag server response	
226	DDOS Stacheldraht server response	
227	DDOS Stacheldraht client spoofworks	
229	DDOS Stacheldraht client check skillz	
236	DDOS Stacheldraht client check gag	
1854	DDOS Stacheldraht handler->agent (niggahbitch)	
1855	DDOS Stacheldraht agent->handler (skillz)	
1856	DDOS Stacheldraht handler->agent (ficken)	

All Snort signature examples from http://snort.org

Ping of Death

GEN:SID	1:499
Message	ICMP Large ICMP Packet
Rule	alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Large ICMP Packet"; dsize:>800; reference:arachnids,246; classtype:bad-unknown; sid:499; rev:4;)
Summary	This event is generated when a large ICMP packet is detected. Also known as the "Ping of Death".
Impact	Denial of Service (DoS) by system crash or bandwidth utilisation.
Detailed Information	Some implementations of the IP stack may result in a system crash or may hang when a large ICMP packet is sent to them. Alternatively a large number of these packets may result in link saturation, especially where bandwidth is limited. This attack was prevalent a number of years ago when the TCP/IP stack of a number of operating systems could not handle large packet payloads.
Affected Systems	Multiple older systems.
Attack Scenarios	A malicious individual may send a series of large ICMP packets to a host with the intention of either crashing or hanging the host, or to saturate the available bandwidth.
Ease of Attack	Simple.
False Positives	A number of load balancing applications use 1500 byte ICMP packets to determine the most efficent route to a host by measuring the latency of multiple paths. HP-UX systems configured with PMTU discovery will send echo requests in response to several types of network connections. PMTU Discovery is enabled in HP-UX 10.30 and 11.0x by default. Windows 2000 uses large ICMP payloads to determine the speed of a link when utilizing a Windows domain controller. If you think this rule has a false positives, please help fill it out.
False Negatives	None Known If you think this rule has a false negatives, please help fill it out.
Corrective Action	

SQL Slammer

GEN:SID	1:2003
Message	MS-SQL Worm propagation attempt
Rule	alert udp $EXTERNAL_NET any -> $HOME_NET 1434 (msg:"MS-SQL Worm propagation attempt"; content:"|04|"; depth:1; content:"|81 F1 03 01 04 9B 81 F1 01|"; content:"sock"; content:"send"; reference:bugtraq,5310; reference:bugtraq,5311; reference:cve,2002-0649; reference:nessus,11214; reference:url,vil.nai.com/vil/content/v_99992.htm; classtype:misc-attack; sid:2003; rev:8;)
Summary	This event is generated when an attempt is made by the "Slammer" worm to compromise a Microsoft SQL Server.
Impact	A worm targeting a vulnerability in the MS SQL Server 2000 Resolution Service was released on January 25th, 2003. The worm attempts to exploit a buffer overflow in the Resolution Service. Because of the nature of the vulnerability, the worm is able to attempt to compromise other machines very rapidly.
Detailed Information	The Monitor Service provided by MS SQL and MSDE uses unchecked client provided data in an SQL version check function. The worm attempts to exploit a buffer overflow in this version request. If the worm sends too many bytes in the request that triggers the version check, then a buffer overflow condition is triggered resulting in a potential compromise of the SQL Server.
Affected Systems	This vulnerability is present in unpatched MS SQL Servers. The following unpatched services containing MS SQL or Microsoft Desktop Engine (MSDE) may potentially be compromised by this worm: * SQL Server 2000 (Developer, Standard, and Enterprise Editions) * Visual Studio .NET (Architect, Developer, and Professional Editions) * ASP.NET Web Matrix Tool * Office XP Developer Edition * MSDN Universal and Enterprise subscriptions

SQL Slammer

Attack Scenarios	This is worm activity.
Ease of Attack	Exploits for this vulnerability have been publicly published. A worm has been written that automatically exploits this vulnerability.
False Positives	None known. If you think this rule has a false positives, please help fill it out.
False Negatives	None known. If you think this rule has a false negatives, please help fill it out.
Corrective Action	Block external access to the MS SQL services on port 1433 and 1434 if possible. Patches from Microsoft are available that fix this vulnerability. The patches are available from www.microsoft.com/technet/security/bulletin/MS02-039.asp
Contributors	Sourcefire Research Team Brian Caswell <bmc@sourcefire.com> Nigel Houghton <nigel.houghton@sourcefire.com>
Additional References	
Rule References	bugtraq: 5310 bugtraq: 5311 cve: 2002-0649 nessus: 11214 url: vil.nai.com/vil/content/v_99992.htm

All Snort signature examples from http://snort.org

Nimda

GEN:SID	1:1293
Message	NETBIOS nimda .eml
Rule	alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS nimda .eml"; flow:to_server,established; content:"|00|.|00|E|00|M|00|L"; reference:url,www.f-secure.com/v-descs/nimda.shtml; classtype:bad-unknown; sid:1293; rev:10;)
Summary	This event is generated when traffic indicating Nimda worm activity is detected.
Impact	Possible infection by the Nimda virus.
Detailed Information	Nimda spreads by file infection, mass emailer, file share, or IIS unicode exploit to attack unpatched systems.
Affected Systems	Windows 95 Windows 98 Windows ME Windows 2000
Attack Scenarios	An unpatched server is connected to the internet and is infected or an infected email is opened. Once infected the worm spreads itself.
Ease of Attack	Simple
False Positives	None known If you think this rule has a false positives, please help fill it out.
False Negatives	None known If you think this rule has a false negatives, please help fill it out.
Corrective Action	Check the suspect host for signs of infection. Apply patches or upgrade the operating system

All Snort signature examples from http://snort.org

Trying for high performance

- Content matching is most expensive process
  - Performed after all other rules are tested
  - Can use offset and depth keywords to limit amount of data to be searched

Trying for high performance

- Deep packet inspection attempts to solve the problem of expensive content matching
- DPI engines scrutinize each **packet** (including the data payload) as it traverses the firewall, and rejects or allows the **packet** based upon a ruleset that is implemented by the firewall administrator.

Uses of Snort

1) To fill holes in commercial vendor's network-based intrusion detection tools:
   - When new attack comes out and signature updates from vendor are slow
   - Run Snort locally on test network to determine signature
   - Write a Snort rule
   - Use BPF command line filtering to watch only service or protocol of interest
   - Snort can be used as a specialized detector for a single attack or family of attacks in this mode

Uses of Snort

2) As a Honeypot monitor
   - Problem with honeypots is that the services they run have to be started before they will record anything, thus miss events such as stealth port scans or binary data streams (unless they perform packet level monitoring)
   - The data coming out of a honeypot requires a skilled analyst to properly interpret results.
   - Snort can be of great help to the analyst/administrator with its packet classification and alerting functionality.

Uses of Snort

3) Used as a "passive trap"
   - Administrators know which services are NOT available on their networks
   - By default, packets looking for those services are malicious traffic (port scanning, backdoors,…)
   - Write Snort rules to watch for traffic headed to these non-existent services

Uses of Snort

4) In focused monitoring:
   - Watch a single critical node or service on the network.
   - Example: the Sendmail SMTP server has a well-known (and extensive) list of vulnerabilities.
   - Single Snort sensor can be deployed with a rule set covering all known Sendmail attacks.
   - This concept can be extended to any network technology that is under-represented by commercial NIDS.

Challenges Snort faces as a stand-alone NIDS

- The constant increase in network speed and throughput – processors cannot keep up
- Sensors cannot maintain information about attacks in progress (e.g., in the case of multi-step attacks)
- Novel approaches to NID necessary to manage ever-increasing data volume

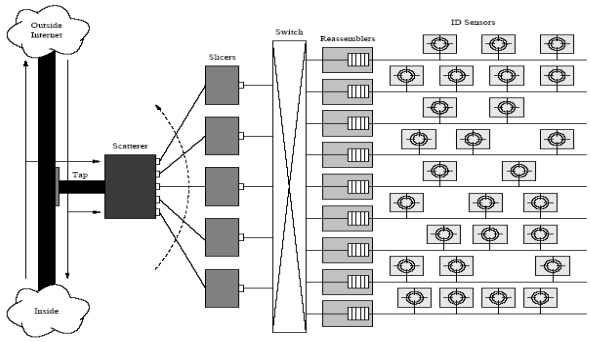
Stateful Intrusion Detection

- Divide the traffic volume into smaller portions
- Have many sensors, each processing a sub-section of the traffic volume
- Each sensor can be configured to detect a specific subset/family of attacks

Stateful Intrusion Detection

- Problems:
  - Must ensure that the traffic division is such that all parts of an attack are sent to the appropriate sensor.
  - If random division is used and different parts of the attack are assigned to different sensors, the sensors may not receive sufficient data to detect the intrusion.

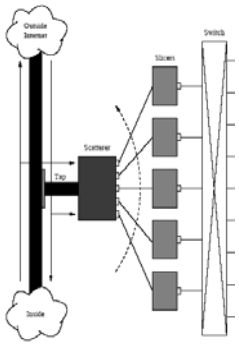
Proposed Architecture



Proposed Architecture

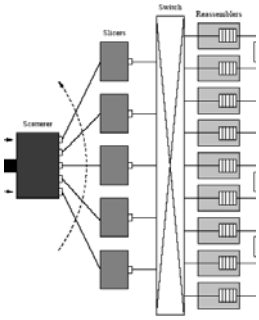
- The systems implements signature-based intrusion detection
- Detection is performed by a set of sensors, each responsible for a subset of signatures
- Each sensor is autonomous and does not interact with other sensors
- Components can be added to the system to achieve higher throughput as needed

Scatterer



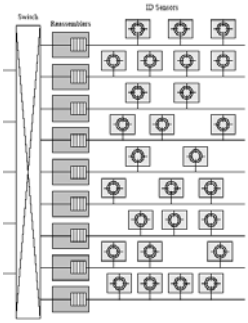
- Receives frames from network tap
- Partitions them into m subsequences to pass onto the slicers
- Can use any algorithm (assume it simply cycles over the m sub-sequences in a round-about fashion)
- Scatterer has to keep up with high traffic throughput, so it does limited processing per frame

Slicers



- Their task is to route the frames to the sensors that should be processing them
- The look at fewer packets, so can perform more complex frame routing.
- Connected to a switch which allows slicer to send frames to one or more of n outgoing channels
- Configuration of the slicers is static

Reassemblers



- The original order of two packets could be lost if they take different paths (over distinct slicers).
- Reassemblers make sure that the packets appear on the channel in the same order they appeared on the high-speed link

Sensors

- Each sensor is associated with a subset of attack scenarios
- Each attack scenario is associated with an *event space*
- The event space specifies which frames are candidates to be part of the manifestation of the attack
- The clauses of an event space can be derived automatically from attack signatures

Performance Issues

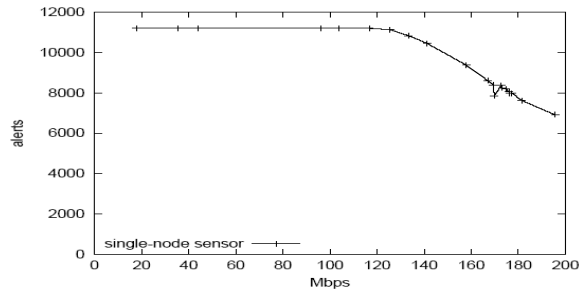


Figure 3. Single-host monitor detection rate for increasing traffic levels.

Performance Issues

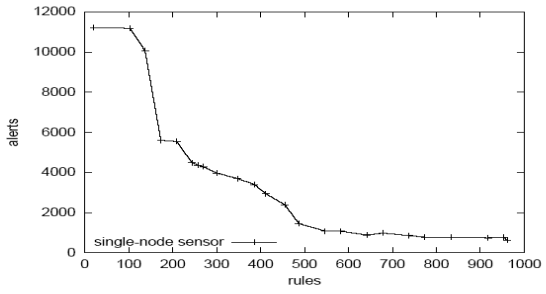


Figure 4. Single-host monitor detection rate for increasing number of signatures.

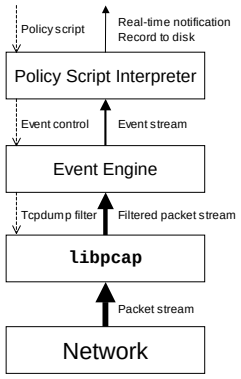
Improving the Method

- More information = More better
  - Think about the bigger picture…
    - Analysis on per-connection basis vs. per-packet
    - Aggregation
  - Regular Expression vs. String matching
  - Consider how the victim responded to the attack
  - Exact activity
  - Semantics

Bro

- Not an anomaly-based system, nor signature-based system, instead e*vent-driven*
- Two major components
  - Event engine (Protocol analysis)
  - Policy script interpreter
- Allows regular expressions
- Can analyze traffic in both directions
- Can detect multiple-stage attacks
- Fewer false positives than Snort
- FreeBSD (also Linux/Solaris)

Bro Architecture



Structure of the Bro System (from [3])

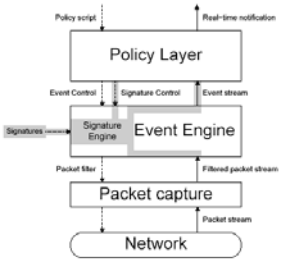
- Rules apply to:
  - Source/Destination Addresses & Ports
  - TCP/UDP
  - Payload
  - Application headers (finger, ftp, portmapper, identd, telnet, rlogin)

Paxson's Suggested Improvements

- Enhance Bro by giving it a contextual signature engine where a signature match is just the starting state
- Use vulnerability profiles
- Detect failed attack attempts
- Count alerts to detect exploit scans

Contextual Signature Engine

Figure 1: Integrating the signature engine (adapted from [25])



From [5]

What traffic is important to Bro

- Supports several internet applications, and captures all packets destined for them:
  - e.g. FTP, Finger, Portmapper, Ident, Telnet, Rlogin, …
- Captures any TCP packet with the SYN, FIN or RST flag set
  - Gets valuable information about ALL connections regardless of service with just a few packets
  - Stores start time, duration, participating hosts and ports, number of bytes in each direction, etc. for all connections

Event Engine

- Performs integrity check on packet headers
- Reassembles IP fragments to analyze complete IP datagrams
- Checks connection state, creating new state if none already exists

TCP Header Analysis

- With initial SYN packet, event engine schedules a timer (currently, 5 minutes)
  - If timer expires and connection hasn't changed state, engine generates connection_attempt event
  - If other endpoint replies with correct SYN ACK packet, engine generates connection_established event
  - If endpoint replies with RST packet, connection has been rejected. Engine generates connection_rejected.
- When connection finishes with normal FIN exchange, generate connection_finished event

UDP Header Analysis

- Simply generate udp_request and udp_reply events
  - Host A sends packets first → initiated a "request" and messages are flagged as udp_requests
  - Packets from Host B are designated as the "replies" (udp_reply)

Other Analyzers

- Conn
  - Generic connection analysis (start time, duration, sizes, hosts, etc.)
  - `new_connection` (new TCP connection)
  - `connection_established`
  - Can store info if partially connected
    - `TCP_INACTIVE, TCP_SYN_ACK_SENT, TCP_CLOSED, etc.`
- HTTP
  - Uses a capture filter targeting TCP destination port 80, 8080, & 8000
  - Can specify pattern in URL (sensitive_URIs)
    - e.g. URIs containing `/etc/passwd`
- Scan
  - Can detect port scans, address scanning, and password guessing

Final Event Engine Functions

- After processing each event, the event engine checks for new events
  - Kept in a FIFO queue
- If so, it processes the new events
- Checks for expired timers

Policy Script Interpreter

- For each event passed to the interpreter:
  - Retrieves compiled code for the corresponding handler
  - Binds values of the events to the arguments of the handler and interprets the code
- That code can in turn generate new events, log notifications, record data to disk, or modify the current state

Signature Language

- Conditions
  - Header
  - Content
    - Incl. `http-request` or `ftp` followed by regexp
  - Dependency conditions
  - Context
- Actions
  - Currently only "event"

Signatures

Fixed String

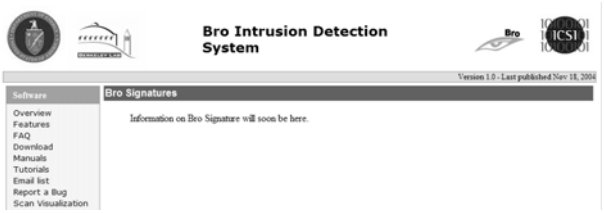
```
alert tcp any any -> a.b.0.0/16 80 (
    msg:"WEB-CGI formmail access";
    uricontent:"/formmail";
    flow:to_server,established;
    nocase; sid:884; [...]
    )
```

Regular Expression

```
signature formmail-cve-1999-0172 {
    ip-proto == tcp
    dst-ip == 1.2.0.0/16
    dst-port = 80
    http /.*formmail.*\?.*recipient=[^&]*[;|]/
    event "formmail shell command"
    }
```

Signatures

As of Feb 2005, Snort.org offers 3,124 rules



Signature Conversion

Figure 2: Example of signature conversion

```
alert tcp any any -> [a.b.0.0/16,c.d.e.0/24] 80
  ( msg:"WEB-ATTACKS conf/httpd.conf attempt";
    nocase; sid:1373; flow:to_server,established;
    content:"conf/httpd.conf"; [...] )
```

(a) Snort

```
signature sid-1373 {
  ip-proto == tcp
  dst-ip == a.b.0.0/16,c.d.e.0/24
  dst-port == 80
  # The payload below is actually generated in a
  # case-insensitive format, which we omit here
  # for clarity.
  payload /.*conf\/httpd\.conf/
  tcp-state established,originator
  event "WEB-ATTACKS conf/httpd.conf attempt"
}%
```

(b) Bro

From [5]

Reporting

- Two cron jobs are installed by default that generate and send email reports of alarms and alerts

•Reports consist of:
- Individual incidents
- Incident type (“likely successful”, “unknown”, “likely unsuccessful”)
- Local host
- Remote host
- Alarms
- Successful/Unsuccessful/Unknown connections
- Connections history

Future work

- Addition of new functionality to Bro
  - Additional protocol analyzers for the event engine
  - Additional event handlers for generated events
- Communication between Bro & routers
  - e.g. If a scan attack is detected, Bro instructs router to drop packets

Signature Based NIDS: The Present

The process:

- New attack is released in the wild
- Manual attack detection
- Manual generation of an attack signature
- Manual update of NIDS’s database with the new signature
- Automated attack detection

Signature Based NIDS: The Future

- Take the costly manual human-based process out of the loop
  - Automated signature generation?
- Killer combination of an IDS:
  - The ability to *learn* (like anomaly-based IDSs)
  - The *low false-positive* rates (like signature-based IDSs)

➔ A signature-based system that can automatically generate signatures!

Attacks on IDSs

- Overload attacks
  - IDS incapable of keeping up with incoming data stream
- Crash attacks
  - IDS crashes or runs out of resources
- Subterfuge attacks
  - IDS is mislead as to true significance of the traffic

References

1. Kreibich, C and Crowcrowft, J. 2004. *Honeycomb: creating intrusion detection signatures using honeypots. ACM SIGCOMM Computer Communication Review.* 34 (1). pp. 51-56. http://portal.acm.org/citation.cfm?id=972384
2. Kreugel, C. et al. 2002. *Stateful intrusion detection for high-speed networks. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy.* May 2002. pp. 285-294. http://www.cs.unc.edu/~jeffay/courses/nidsS05/signatures/kemerer-slicing-SP02.pdf
3. Paxson, V. 1999. *Bro: a system for detecting network intruders in real-time. Computer Networks.* 31(23-24). December 1999. pp. 2435-2463. http://www.cs.unc.edu/~jeffay/courses/nidsS05/signatures/paxson-bro-cn99.pdf
4. Roesch, M. 1999. *Snort—lightweight intrusion detection for networks. In: Proceedings of LISA '99.* 7-12 November 1999. USENIX. pp. 229-238. http://portal.acm.org/citation.cfm?id=1039864
5. Sommer, R and Paxson, V. 2003. *Enhancing byte-level network intrusion detection signatures with context. In: Proceedings of the 10th ACM conference on Computer and Communications Security*. October 2003. ACM. pp. 262-271. http://www.cs.unc.edu/~jeffay/courses/nidsS05/signatures/sommer-context-ccs03.pdf
6. Totsuka, A et al. 2000. *Network-based intrusion detection—modeling for a larger picture*. *Proceedings of LISA 2000.* 3- 8 November 2000. USENIX. pp. 227-232. http://www.usenix.org/events/lisa02/tech/full_papers/totsuka/totsuka.pdf