Network and host-based intrusion detection systems

CYBR371: System and Network Security, (2024/T1)

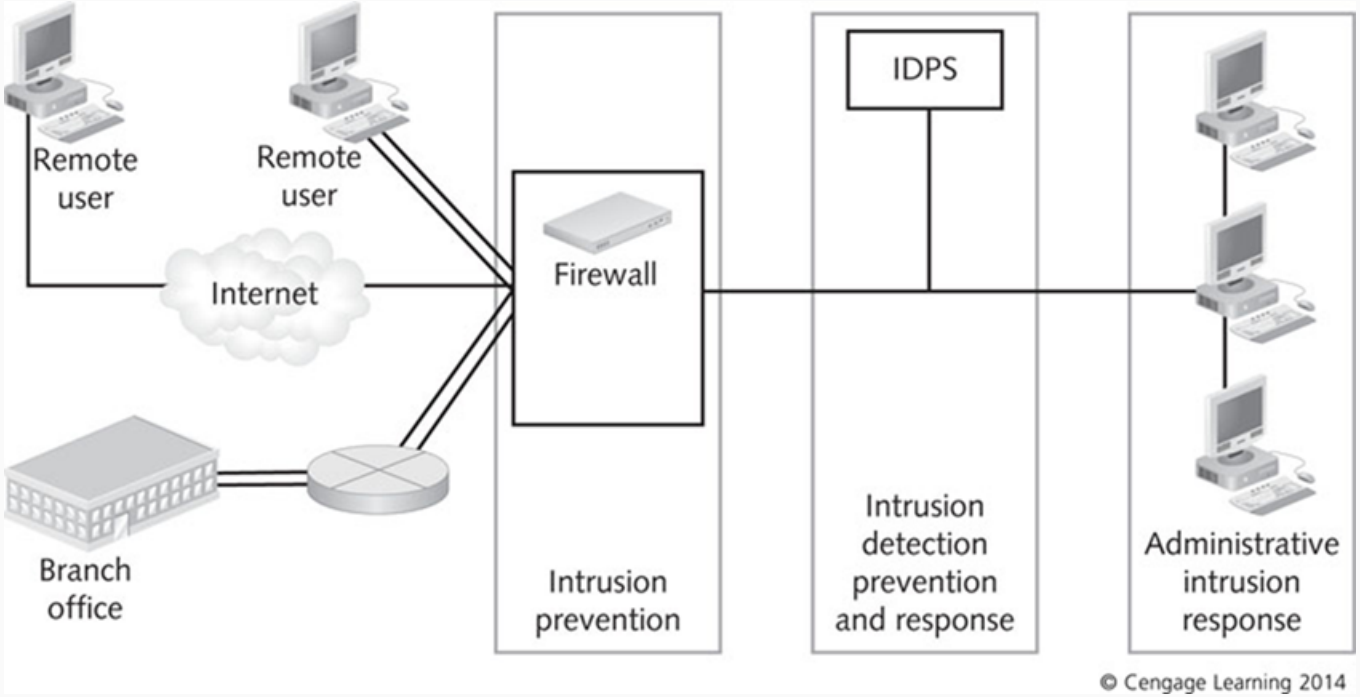
Arman Khouzani, Mohammad Nekooei
Slides modified from "Masood Mansoori"

01 May 2024

Victoria University of Wellington – School of Engineering and Computer Science

Goals of an IDPS

- Monitoring
- Logging
- Response
- Accountability

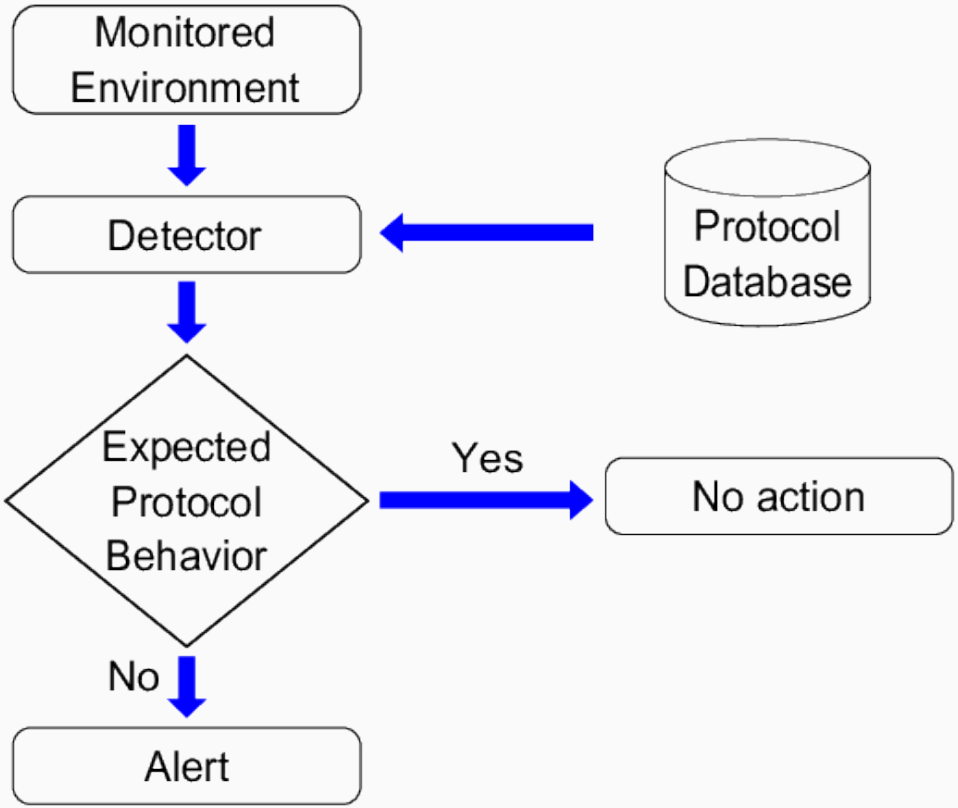


IDS measurement of Reliability

- **False positives:** legitimate traffic rather than actual attacks
- **False negatives:** genuine attacks that an IDPS does not detect could occur
- **True positive:** used to describe a genuine attack that an IDPS detects successfully
- **True negatives:** legitimate communications that do not set off an alarm

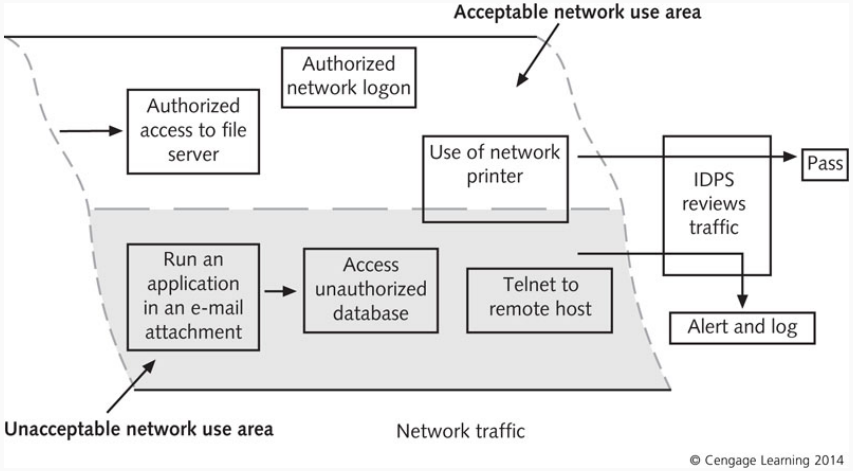
IDPS categories based on Detection Techniques

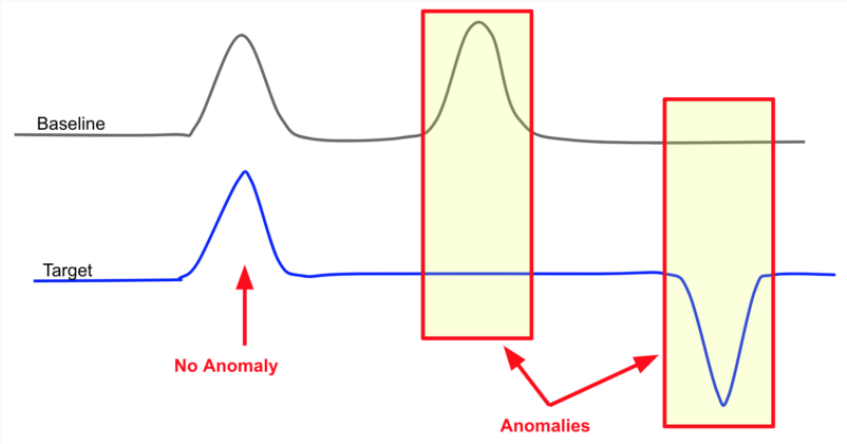
1. Anomaly Based Detection
2. Stateful Protocol Analysis and Detection
3. Signature Based Detection



IDPS Detection Engines: Anomaly detection system

1. Anomaly detection system





IDPS Detection Engines: Anomaly detection system

Simple Example:

```
module AnomalousDNS;
export {
    redef enum Notice::Type += {
    Conn_Duration,
    Conn_Packets,
    };
    ## Connection duration limit
    const conn_duration_limit = 45secs &redef;
    ## Connection packets limit, measured on origin
    const conn_pkts_limit = 12 &redef;
}
event dns_message(c: connection, is_orig: bool, msg: dns_msg,
    len: count)
    {
    if ( c$duration > conn_duration_limit )
...
```

IDPS Detection Engines: Stateful Protocol Analysis

2. Stateful Protocol Analysis: Compares predetermined profiles of generally accepted definitions of benign protocol activity for each protocol state against observed events.

- IDPS is capable of understanding and tracking the state of network, transport, and application protocols that have a notion of state.

IDPS Detection Engines: Stateful Protocol Analysis

Stateful protocol analysis approaches:

- Traffic rate monitoring
- Protocol state tracking
  - E.g. FTP unauthenticated vs. authenticated sessions and commands
  - E.g. repeated commands
- Dynamic Application layer protocol analysis
  - E.g. length of usernames arguments
- IP packet reassembly

IDPS Detection Engines: Stateful Protocol Analysis

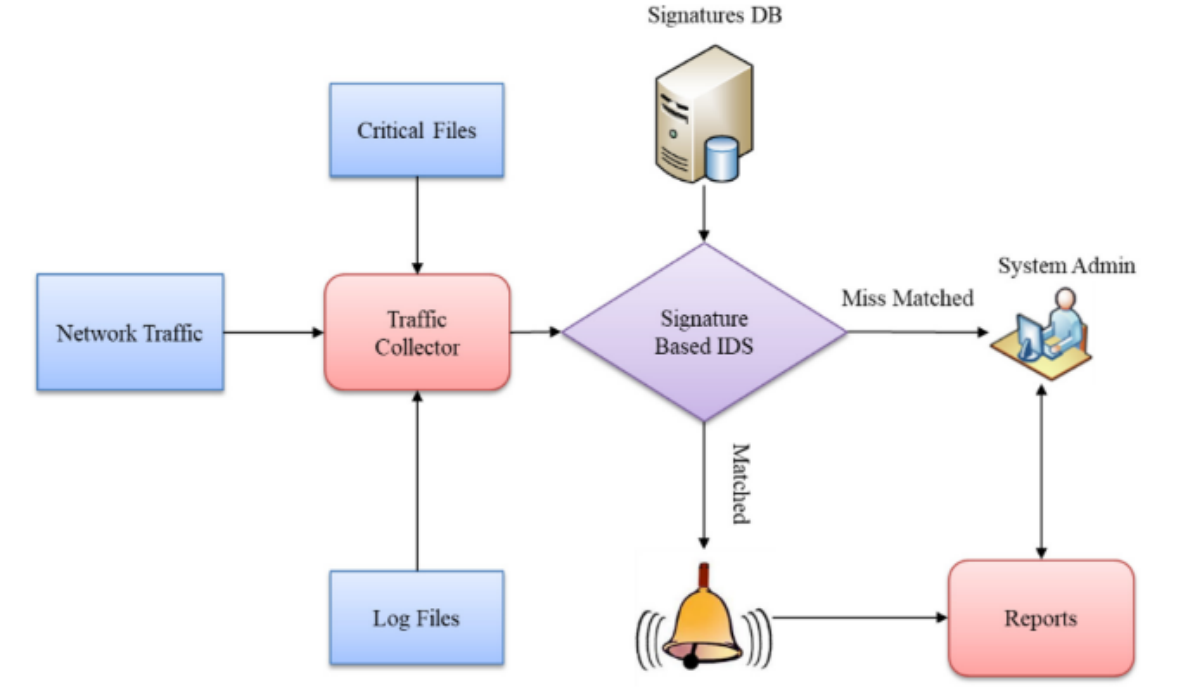
Stateful Protocol Analysis Issues:

- Protocol information might not be fully available
- Vendors deviate from the protocol guidelines
- Resource intensive
- Failure to detect DDoS attack as a result of benign protocol behavior

IDPS Detection Engines: Signature detection

3. Signature detection: triggers alarms based on characteristic signatures of known attacks.

- Signature-based IDPS best for companies that want a basic IDPS and mostly concerned with known attacks.



IDPS Detection Engines: Comparison

Detection method	Advantages	Disadvantages
Anomaly	Because an anomaly detection system is based on profiles an administrator creates, an attacker cannot test the IDPS beforehand and anticipate what will trigger an alarm.	Configuring the IDPS to use profiles of network users and groups requires considerable time.
	As new users and groups are created, IDPS profiles can be updated to keep up with these changes.	Updating IDPS profiles can be time consuming.
	Because an anomaly detection system does not rely on published signatures, it can detect new attacks.	The definition of what constitutes normal traffic changes constantly, and the IDPS must be reconfigured to keep up.
	The system can detect attacks from inside the network by employees or attackers who have stolen employee accounts.	After installation, the IDPS must be trained for days or weeks to recognize normal traffic.
Signature	This approach makes use of signatures of well-known attacks.	The database of signatures must be updated to maintain the IDPS's effectiveness.
	This IDPS can begin working immediately after installation.	New types of attacks might not be included in the database.
	This IDPS is easy to understand and less difficult to configure than an anomaly-based system.	By making minor alterations to an attack, attackers can avoid matching a signature in the database.
	Each signature in the database is assigned a number and name so that the administrator can specify which attacks should set off an alarm.	Because a misuse-based system requires a database, extensive disk storage space might be needed.

Detection and Prevention Capabilities

Prevention Capabilities

- IDPS can be configured to take preventative countermeasures.
  - Example: resetting all network connections when an intrusion is detected.
  - Need to be careful with this!
- Some IDPSs allow administrators to specify which measure should be taken for each alert type

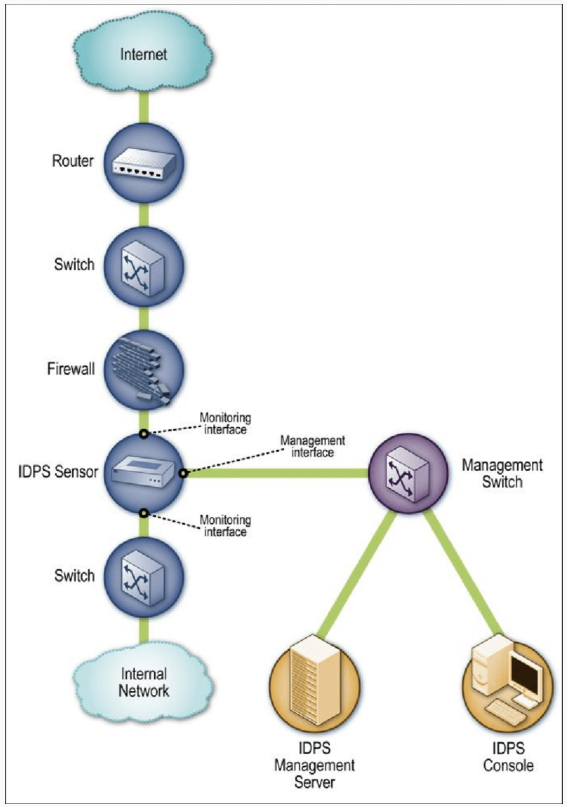
Detection and Prevention Capabilities

IDPS response actions:

- Alarm
- Drop
- Reset
- Code analysis: Prevents malicious code from running
- File system monitoring: Prevent files from being modified
- Network traffic analysis & filtering: stop incoming traffic (act as firewall)

Examining IDPS Components

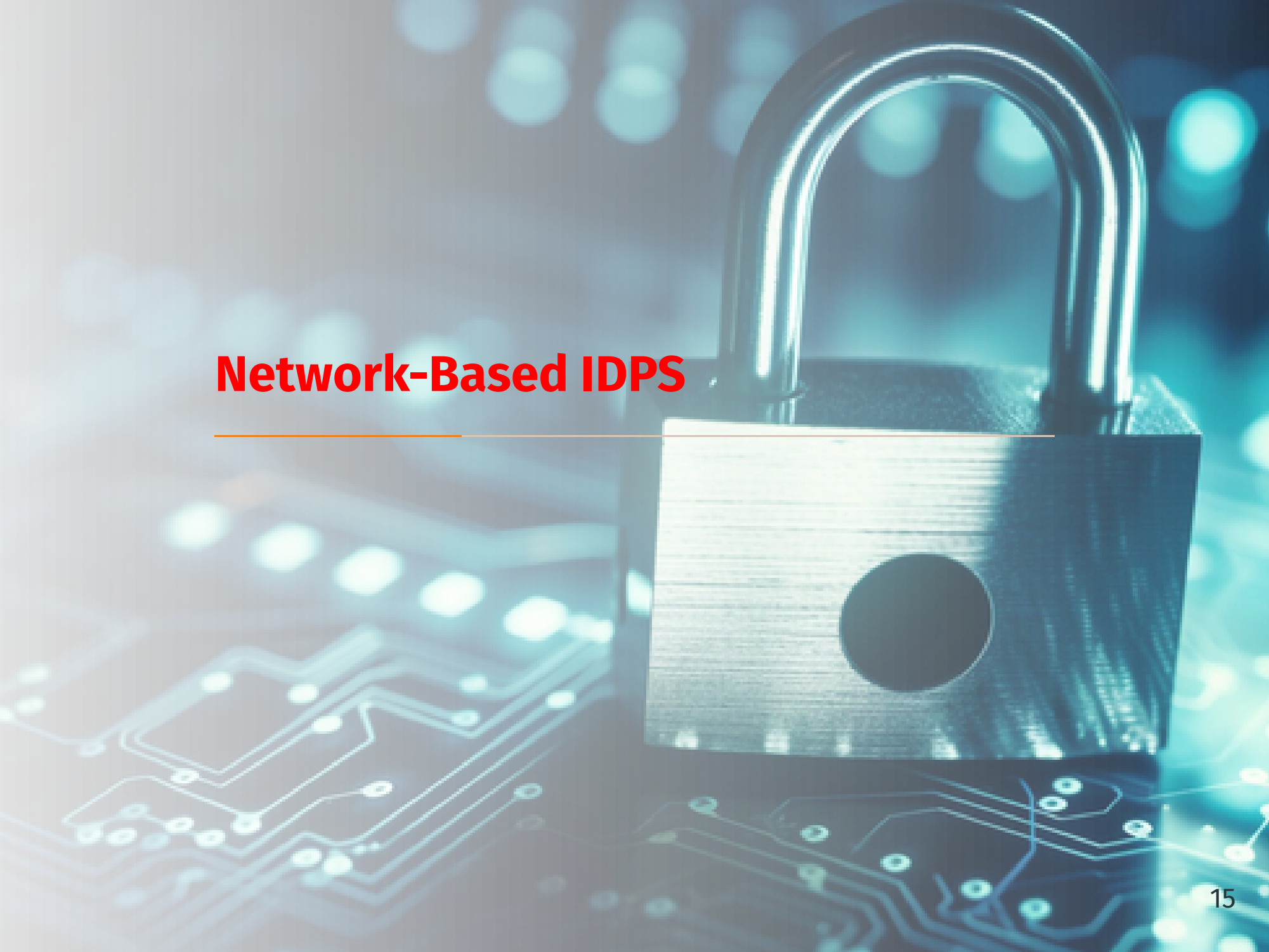
1. Network sensors or host-based agents.
2. Detection and prevention engine.
    - Database server that stores attack signatures or behaviours.
3. Command console and management interface



Options for IDPS Deployments

- Network-based IDPS
- Host-based IDPS
- Hybrid IDPS

Network-Based IDPS



Network-Based IDPSs

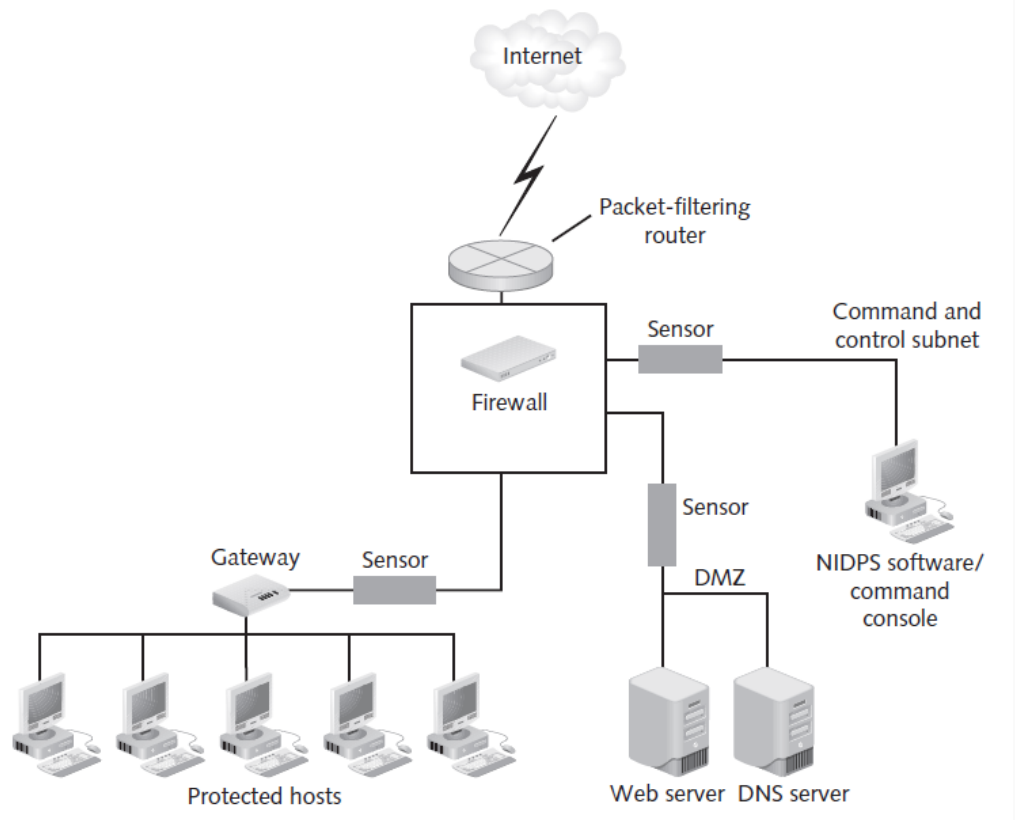
A network-based IDPS (NIDPS) resides on a segment of an organisation's network and monitors traffic.

- Cost of ownership reduced
- Packet analysis
- Real time detection and response
- Malicious intent detection
- Complement and verification
- Operating system independence
- Does not tell whether attack occurred

Network-Based IDPS Sensor Placement

Sensor is hardware or software that monitors network traffic in real time.

- Sensors should be placed at common-entry points.
- Sensors could be positioned at either side of the firewall.
  - Behind the firewall is a more reasonable location.



NIDS Sensors and Their Placement

Types of Sensors an NIDPS can use:

Inline sensors: positioned so that network traffic must pass through it.

Passive sensors: monitor copies of traffic; no actual traffic passes through them.

Question? How can NIDPS access the traffic in a switched environment?

Network-Based IDPS Capabilities

Information Gathering Capabilities:

- Identifying hosts, operating systems, applications and network characteristics

Logging Capabilities:

- Timestamp (usually date+time), Connection or session ID.
- Event or alert type and Rating (e.g., priority, severity, impact, confidence).
- Network, transport, and application layer protocols.
- Packet header and protocol information.
- Number of bytes transmitted over the connection.
- Decoded payload data, such as application requests and responses.
- State-related information (e.g., authenticated username).

Network-Based IDPS Capabilities

Detection Capabilities:

- Application layer reconnaissance and attacks
  - example?
- Transport layer reconnaissance and attacks
  - example?
- Policy violations
  - example?

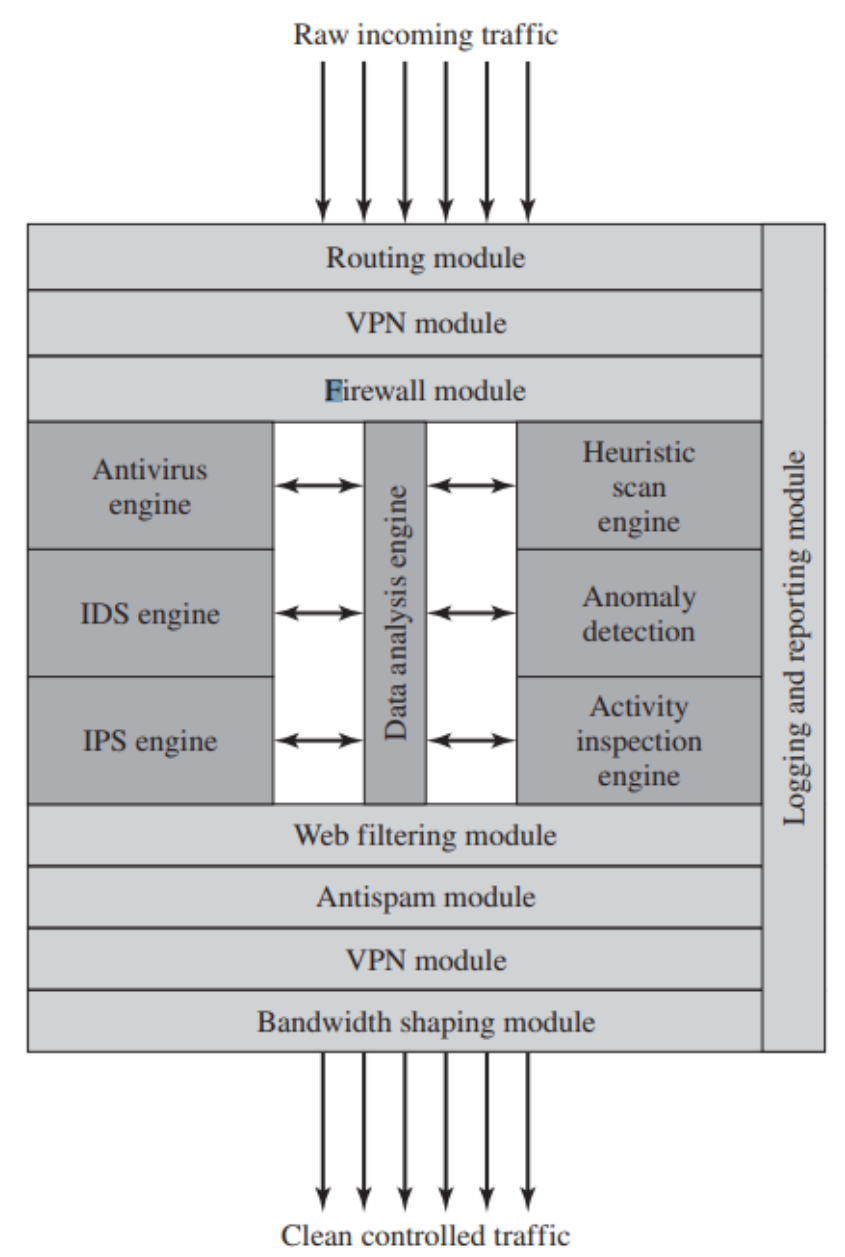
Network-Based IDPS Capabilities

NIDPS prevention capabilities vary based on sensor types:

- Passive only: Ends the current TCP session.
- Inline only: Uses inline firewalling and bandwidth throttling, and alters malicious content.
- Passive and inline – Reconfigures other network security devices.

Network-Based IDPS Capabilities

Attacks and Internet Threats		Protections	
TCP			
•Invalid port numbers •Invalid sequence •numbers •SYN floods •XMAS tree attacks •Invalid CRC values •Zero length •Random data as TCP •header	•TCP hijack attempts •TCP spoofing attacks •Small PMTU attacks •SYN attack •Script Kiddie attacks •Packet crafting: different TCP options set	•Enforce correct TCP flags •Enforce TCP header length •Ensures a proper 3-way handshake •Closes TCP session correctly •2 sessions, one on the inside and one on the outside •Enforce correct TCP flag usage •Manages TCP session timeouts •Blocks SYN attacks	•Reassembly of packets ensuring correctness •Properly handles TCP timeouts and retransmits timers •All TCP proxies are protected •Traffic Control through access lists •Drop TCP packets on ports not open •Proxies block packet crafting
UDP			
•Invalid UDP packets •Random UDP data to bypass rules	•Connection prediction •UDP port scanning	•Verify correct UDP packet •Drop UDP packets on ports not open	



Network IDPS Example (Snort)

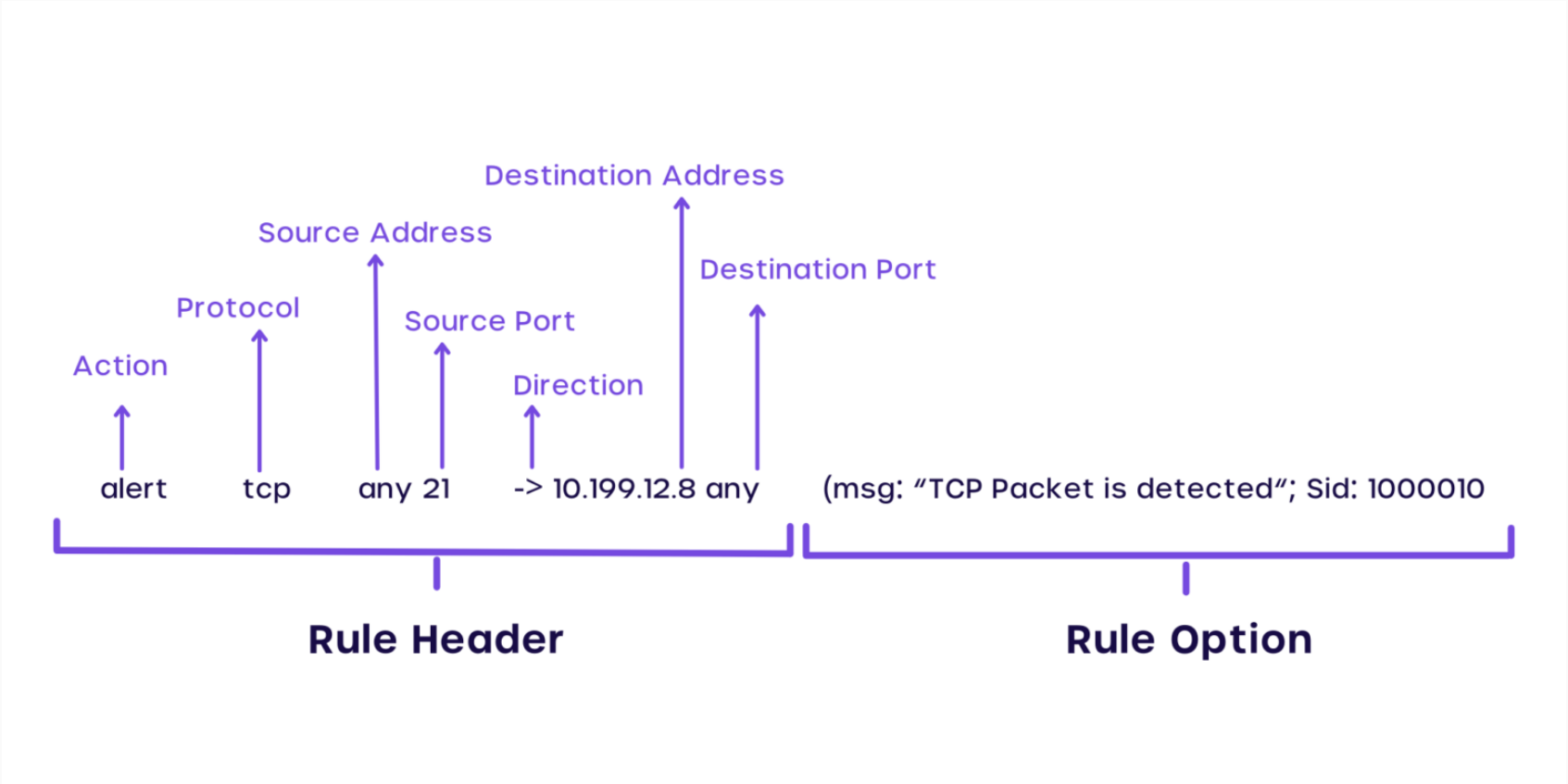
Snort is the defacto Open-Source Network Intrusion Detection System

3 Modes of Snort:

- Sniffer: print Data, Header, Header+Data
- Logging
- NIDS

Network IDPS Example (Snort)

Formatting of its signatures/rules:



Network IDPS Example (Snort)

- **Detecting Web content Attack**

```
alert tcp 192.168.1.0/24 any -> 130.195.5.1 80 (content:
    “”XYZ; msg: ”Suspecious ”packet; sid:100001;)
```

or

```
alert tcp 192.168.1.0/24 any -> any any (content: ”HTTP”
   ; offset: 5; msg: ”HTTP matched”;)
```

- **Detecting FTP Connection Attempt**

```
alert tcp 192.168.x.x any -> $HOME_NET 21 (msg”:FTP
   connection ”attempt; sid:1000002; rev:1;)
```

Snort Rules Examples

- **Detecting oversized Packets**

```
alert ip any any -> 192.168.1.0/24 any (dsize: > 6000;
    msg: "Large size IP packet detected";)
```

- **Detecting Flagged packets**

```
alert tcp any any -> 192.168.1.0/24 any (flags: SF; msg:
     "SYNC-FIN packet detected";)
```

- **Detecting TCP SYN Floods**

```
alert tcp any any -> 192.168.1.0 443 (msg: "TCP SYN
    flood"; flags:!A; flow: stateless; detection_filter:
    track by_dst, count 70, seconds 10; sid:2000003;)
```

- **Detecting Conficker Worm**

```
alert tcp any any -> any 445 (msg: "conficker.a shellcode"; content: "|e8 ff ff ff ff
     c1|^|8d|N|10 80|1|c4|Af|81|9EPu|f5 ae c6 9d a0|0|85 ea|0|84 c8|0|84 d8|0|c4|0
     |9c cc|IrX|c4 c4 c4|,|ed c4 c4 c4 94|&<08|92|\;|d3|WG|02 c3||dc c4 c4 c4 f7 16
     96 96|0|08 a2 03 c5 bc ea 95|"; sid: 1000003; rev: 1;)
```

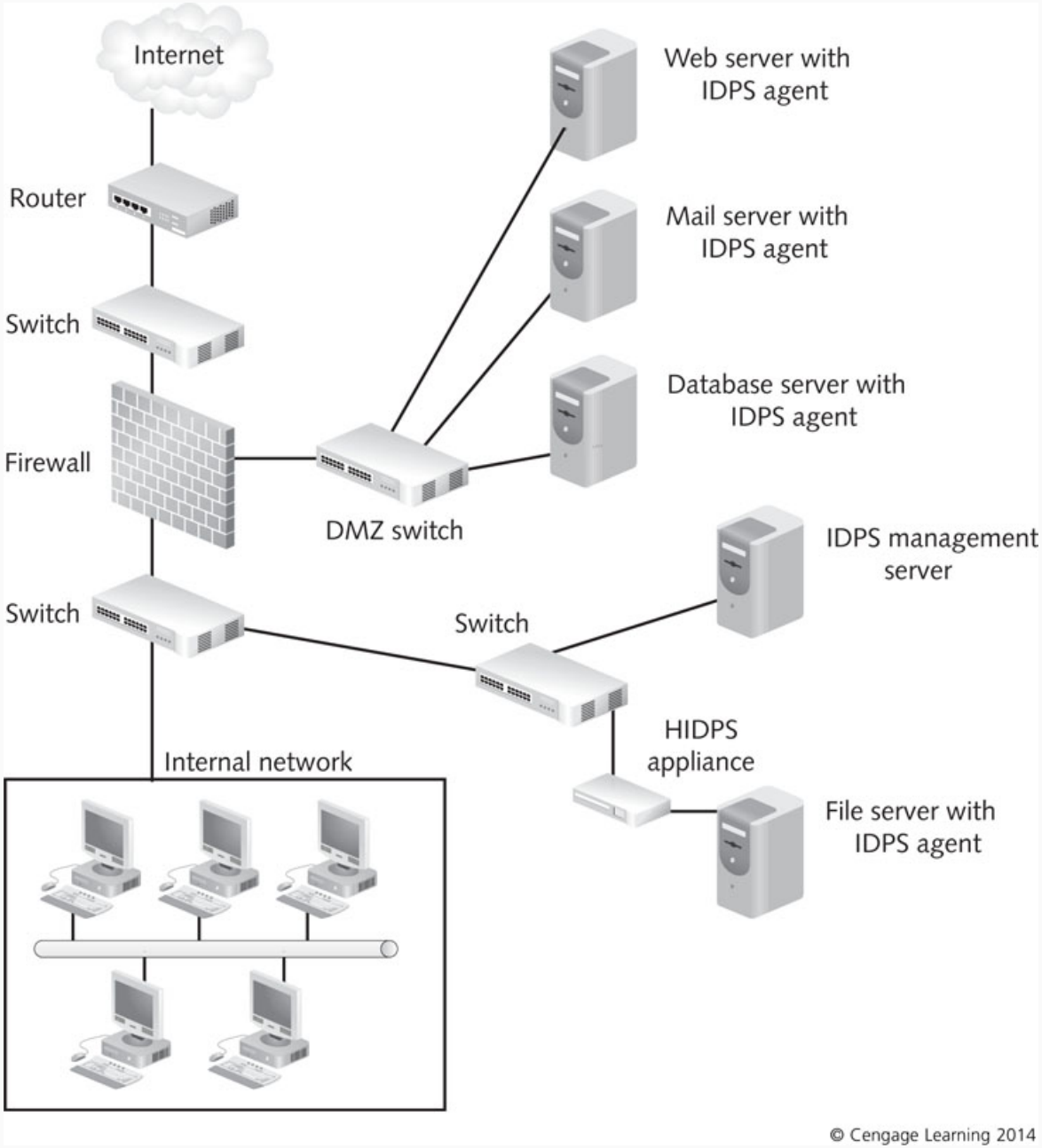
Host-Based IDPS

Host-Based IDPS

Host-based IDPS (HIDPS)

- Deployed on hosts in the network perimeter
- Commonly uses management servers
- Gathers system variables such as
  - System processes, CPU use, file accesses, system logs, and system and application configuration changes
- Does not sniff packets as they enter the LAN
  - Monitors log file entries and user activity

Host-Based IDPS: A typical HIDPS deployment



Host-Based IDPS

- Often used to protect servers (Webserver, database server).
- Can tell whether an attack attempt was successful.
- Can detect attacks that would get past NIDPS.
- Provides only data pertaining to the host, not network as a whole.
- Compares records stored in audit logs.

Host-Based IDPS: Configuraions

- Centralised configuration:
  - HIDPS sends all data to a central location.
  - Host's level of performance is unaffected by the IDPS.
  - Alert messages that are generated do not occur in real time.
  - RAM, hard disk storage, and processor speed requirements are minimal.
- Distributed configuration:
  - Processing of events is distributed between the host and the console.
  - Host generates and analyses them in real time.
  - Performance reduction in host.
  - Host should be equipped with high memory and processor speed.

Hybrid IDPSs (Hybrid Deployment)

Hybrid IDPS

- Combines the features of HIDPSs and NIDPSs.

Combining IDPS Sensor Locations:

- Put sensors on network segments and network hosts.
- Can report attacks aimed at particular segments or the entire network.

Securing IDPS Components

Both NIDS and HIDS:

- Must be able to handle the volume of traffic or activity.
- Should be tested regularly.
- Communication among components should be encrypted.
- Authentication should be required for use and administration of the IDPS.
- IDPSs should be able to work during DoS attacks.
- Their OSs should be patched & hardened (Bastion Hosts).

NIDS:

- Network downtime while deploying sensors.
- Sensors should not be addressable.

HIDS:

- Ensuring that HIDPS services cannot be disabled.

NDR and EDR

NDR: Network Detection and Response

EDR: Endpoint Detection and Response

Tripwire Intrusion Detection System

What is Tripwire

Tripwire: Host-based IDS which identifies changes made to specified files and directories.

Why use Tripwire:

- Damage assessment.
- Track system changes.
- Speed recovery from a break-in by reducing the number of files you must restore to repair the system.

Monitors Unix File System

- Permissions.
- Inode number.
- Number of links (i.e. inode reference count).
- User ID of owner.
- Group ID of owner.
- File type.
- File size.
- File is expected to grow.
- Device number of the disk on which the inode is stored.
- Device number of the device to which the inode points.
- Number of blocks allocated.
- Access timestamp.
- Modification timestamp.
- Inode creation and/or modification timestamp.
- CRC-32 hash of the data.
- MD5 hash of the data.
- SHA hash of the data.
- HAVAL hash of the data.

How does Tripwire work

Protects Itself with:

- El Gamal 1024-bit asymmetric cryptography

Message-digest algorithms used to ensure data integrity.

- MD5
- HAVAL
- SHA/SHS

Authentication and Encryption Between Manager and Server:

- All data transmission uses SSL (Secure Socket Layer).
- 168 Triple DES Encryption.

How does Tripwire work

1. Generates a baseline by taking a snapshot of specified files and directories.
2. Then compares files and directories against the baseline database.
3. And reports any modifications, additions, or deletions.

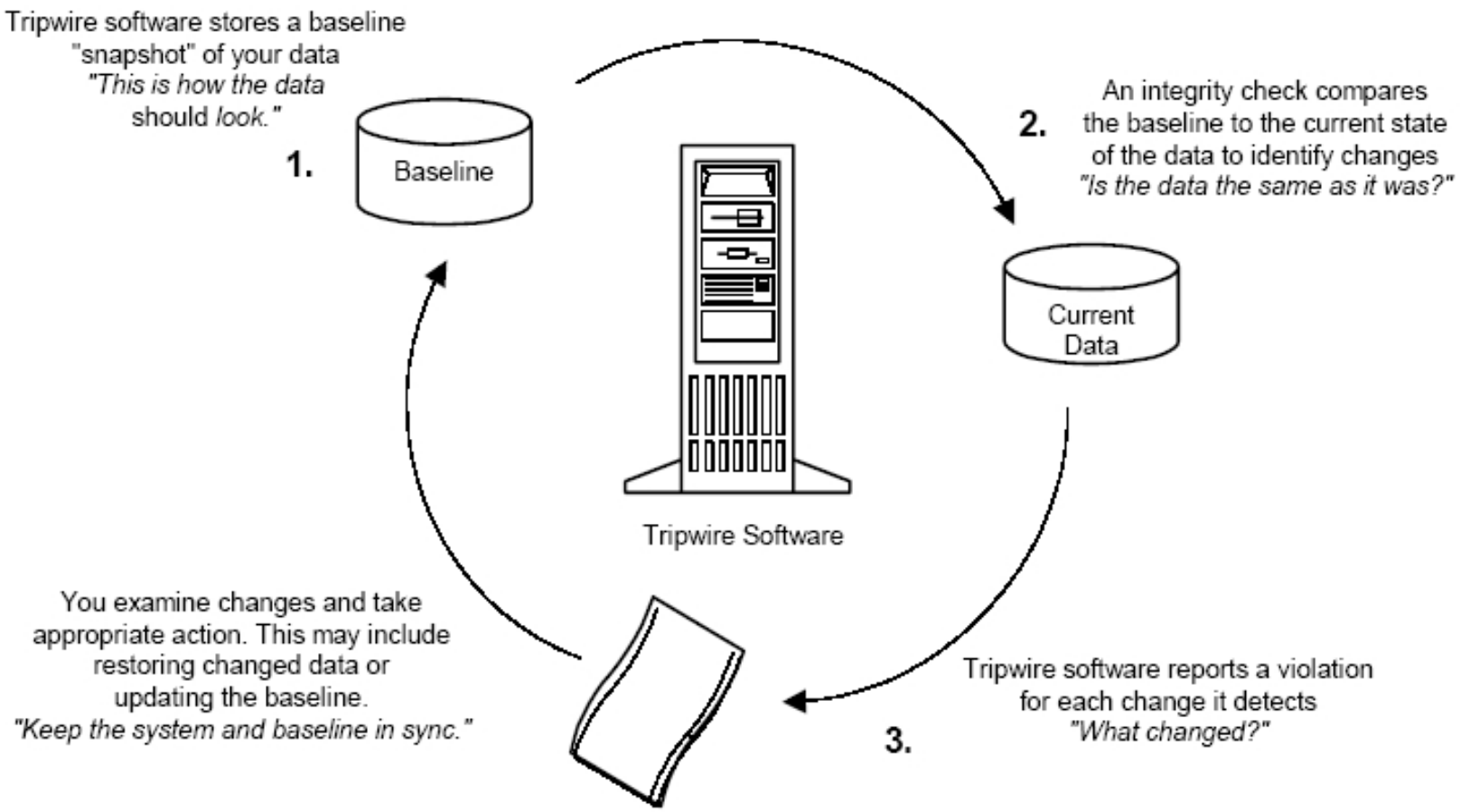
How does Tripwire work

Tripwire files are signed and encrypted using site and local keys.

These protect the configuration, policy, database, and report files from being viewed or altered except by users who know the site and/or local passphrases.

This means that, even if an intruder can obtain root access to your system, they will not be able to alter the Tripwire files to hide their tracks unless they also know the passphrases.

How does Tripwire work



Monitors Windows NT/2000

File System:

- Archive flag
- Read only flag
- Hidden flag
- Offline flag
- Temporary flag
- System flag
- Directory flag
- Last access time
- Last write time
- Create time
- File size
- MS-DOS 8.3 name
- NTFS Compressed flag

Registry:

- Maximum length of data for any value in the key
- Security descriptor control
- Size of security descriptor
- Last write time
- Registry type: key or value
- Type of value data
- Length of value data
- CRC-32 hash of the value data
- MD5 hash of the value data
- SHA hash of the value data
- HAVAL hash of the value data
- Type of value data

Using tripwire

- **Update the Tripwire policy file:** To change the list of files Tripwire monitors or how it treats integrity violations.
- **Build a database of critical system files** to monitor based on the contents of the new, signed Tripwire policy file.
- **Run a Tripwire integrity check:** Compare the newly-created Tripwire database with the actual system files, looking for missing or altered files.
- **Examine the Tripwire report file:** View the Tripwire report file using **`twprint`** to note integrity violations.

Policy File

- Contains comments, rules, directives, and variables to check your system.
- Each rule in the policy file specifies the files and directories you wish to monitor.
- Encrypted to prevent unauthorised modifications.

Tripwire Policies

Tripwire Policies are folders and specified actions to be taken.

- Check policies on the host system:

```
gedit /etc/tripwire/twpol.txt
```

```
# Variables to make configuration easier

SEC_CRIT=$(IgnoreNone)-Sha; # Critical files that cannot change
SEC_BIN=$(ReadOnly); # Binaries that should not change
SEC_CONFIG=$(Dynamic);# Config files that are changed infrequently but accessed often
SEC_LOG=$(Growing); # Files that grow, but that should never change ownership
SEC_INVARIANT=+tpug; # Directories that should neverchange permission or ownership
SIG_LOW=33; # Non-critical files that are of minimal security impact
SIG_MED=66; # Non-critical files that are of significant security impact
SIG_HI=100; # Critical files that are significant points of vulnerability
```

Tripwire Policy Example

```
# Tripwire Binaries
( rulename = "Tripwire Binaries", severity = $(SIG_HI)){
    $(TWBIN)/siggen   -> $(SEC_BIN);
    $(TWBIN)/tripwire -> $(SEC_BIN);
    $(TWBIN)/twadmin  -> $(SEC_BIN);
    $(TWBIN)/twprint  -> $(SEC_BIN) ;
}

# Critical Libraries
( rulename = "Root file-system libraries", severity = (SIG_HI)){
    /lib              -> $(SEC_BIN) ;
    /bin              -> $(SEC_BIN) ;
}

# log Libraries
( rulename = "log", severity = (SIG_HI)){
    /var/log/         -> $(SEC_LOG) ;
}
```

Tripwire Policy Example

```
# Commonly accessed directories that should remain static with
    regards to owner and group
(
 rulename = "Invariant Directories",
 severity = $(SIG_MED)
)
{
    /           -> $(SEC_INVARIANT) (recurse = 0);
    /home       -> $(SEC_INVARIANT) (recurse = 0);
    /tmp        -> $(SEC_INVARIANT) (recurse = 0);
    /usr        -> $(SEC_INVARIANT) (recurse = 0);
    /var        -> $(SEC_INVARIANT) (recurse = 0);
    /var/tmp    -> $(SEC_INVARIANT) (recurse = 0);
}
```

Summary

- Intrusion detection and prevention systems (IDPSs) add another line of defense behind firewalls and antivirus software
- IDPS components include sensors, management servers, command consoles, and databases of signatures
- A network-based IDPS (NIDPS) uses sensors positioned at key points on the network

References

- NIST 800-94 – A guide to Intrusion Detection and Prevention Systems.
- A survey on Intrusion Detection and Prevention in Wireless Ad-hoc Networks - DOI: 10.1016/j.sysarc.2019.101701

Next: Deception Systems and Honeypots