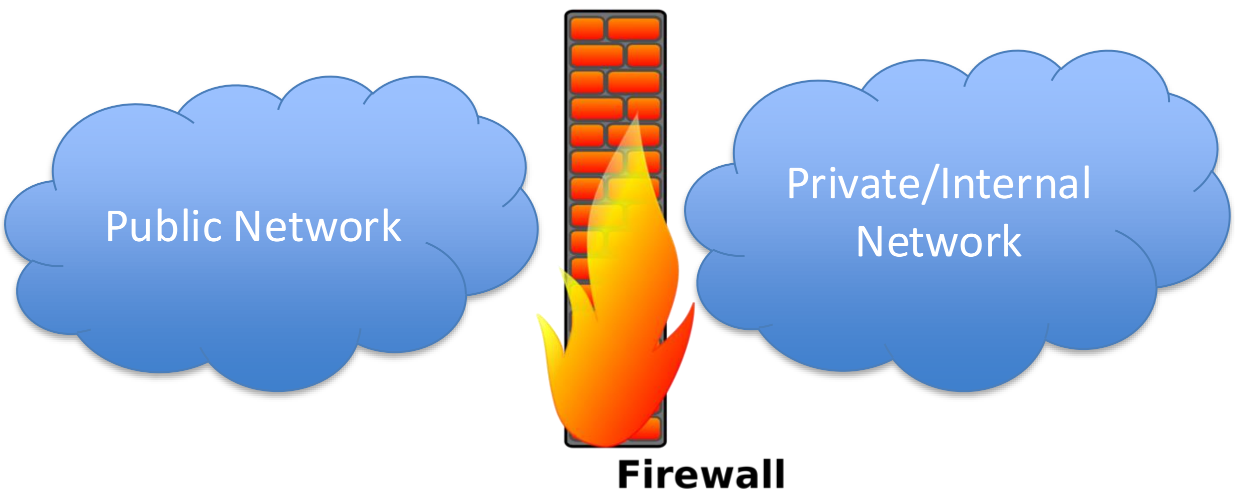
Network Security Applications

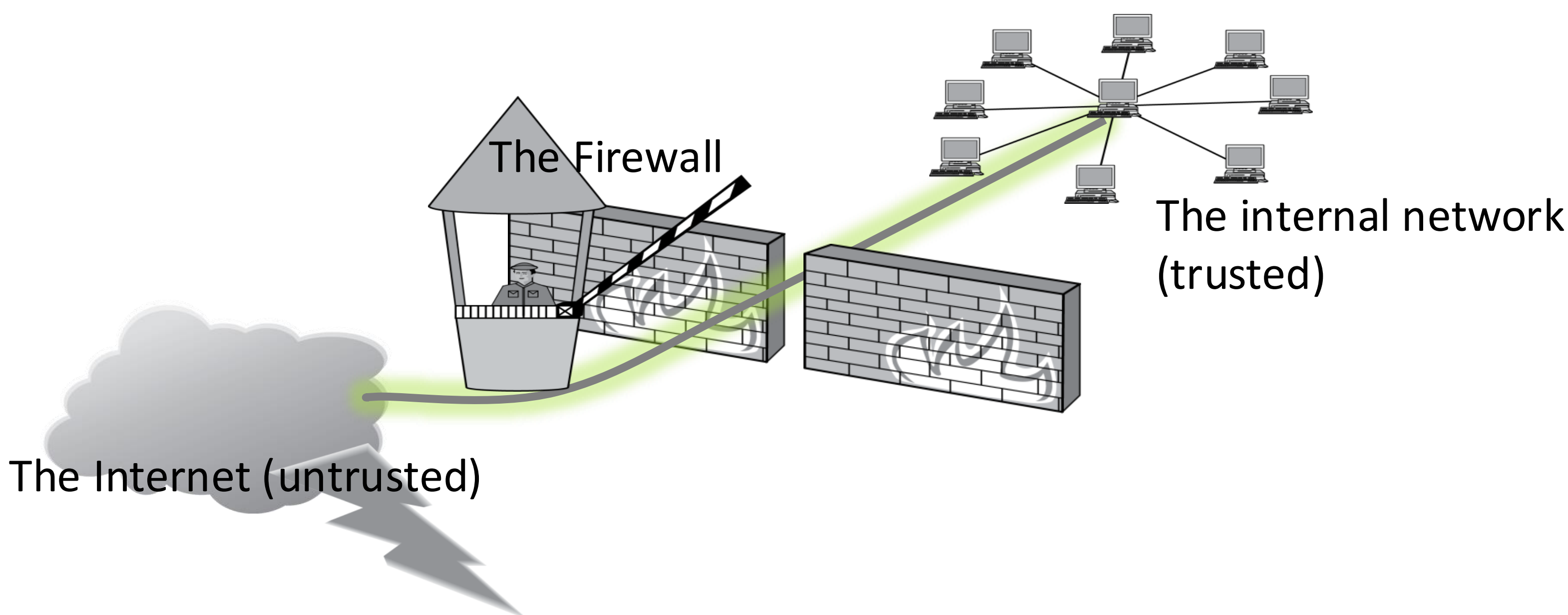
Firewall

- A firewall is a network component used to prevent unauthorized access to a networked system.
  - Similar to firewalls in building construction with respect to isolating different areas.



Firewall

- Essential Firewall Functionalities
  - Mediate Network Traffic
    - E.g., all network traffic exchanged between internal and external network will go through a firewall
  - Enforce Security Policies
    - Firewall policies are represented by a set of rules
    - Usually work in the network layer and transport layer
    - Certain firewalls also works at application layer



Firewall Policy

- A firewall rule/policy can be considered as:

If (the ***properties*** of this packet satisfies certain condition) {
    Take this ***action***;
}

Policy Actions

- A firewall policy enforces one of the three actions for a network packet
  - **Accepted**: permitted through the firewall
  - **Dropped**: discard the packet without notifying the source
  - **Rejected**: discard the packet and notify the source

Properties of a Packet

- Several properties of each packet will be investigated
  - Protocol Type (TCP or UDP)
  - The source and destination IP addresses
  - The source and destination ports
  - The application-layer information of the packet (e.g., certain string that indicates a virus)

Firewall Rule Examples

- A complete rule is defined by the ordered tuple
  - <Dir, Proto, Src IP, Src Port, Des IP, Dest Port, Action>
- Example:
  - <OUT, TCP, 192.168.20.*, ANY, 64.233.179.104, 80, ACCEPT>

If (the ***properties*** of this packet satisfies ***certain condition***)
{
Take this ***action***;
}

No Rule Is Matched?

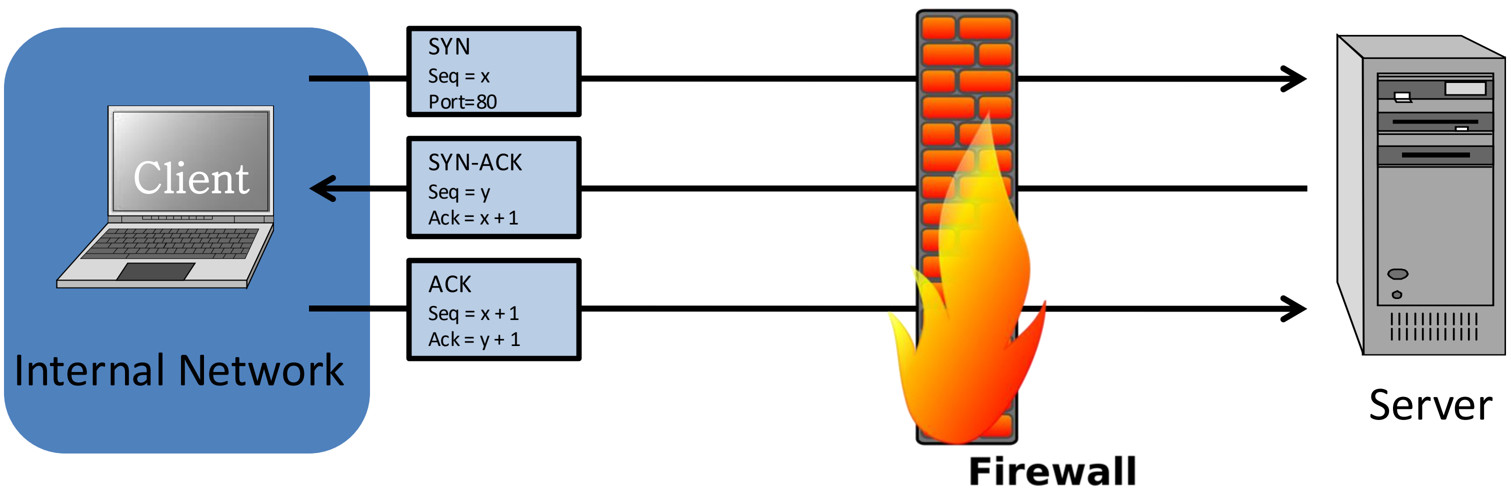
- Blacklist Approach (default-allow)
  - If there is no rule to match this packet, this packet will be allowed through by default
  - Pros: service flexibility
  - Cons: malicious traffic could go through
- Whitelist Approach (default-deny)
  - If there is no rule to match this packet, this packet will be dropped by default
  - Pros: a safer approach to define the rule set
  - Cons: must consider all possible legitimate traffic

Firewall Types

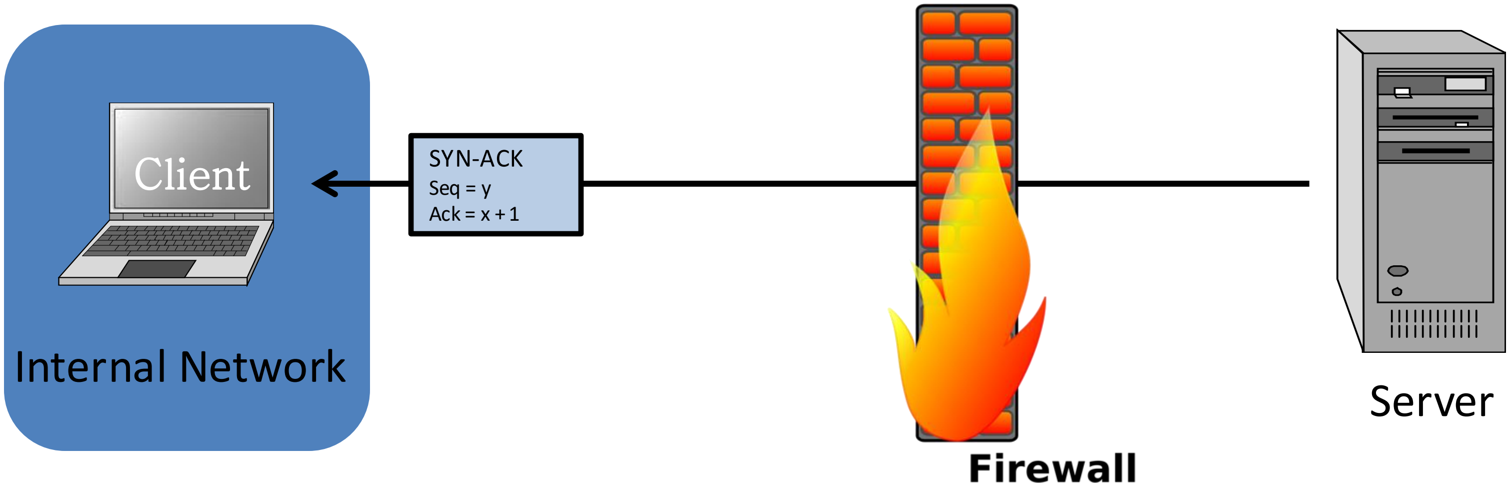
- Stateless Firewall
  - A stateless firewall does not maintain the context (or the “state”) of a packet when processing it
  - It take each packet in isolation without considering other packets related to this packet
- Stateful Firewall
  - A stateful firewall maintains information of the network session where each packet belongs to (i.e., context information)

Stateless Firewall

- Objective
  - Allow internal hosts to establish TCP connections with external web services (e.g., port 80)
- Rules
  - Allow outbound SYN and ACK packets whose destination port is 80
  - Allow inbound SYN-ACK packets whose source port is 80



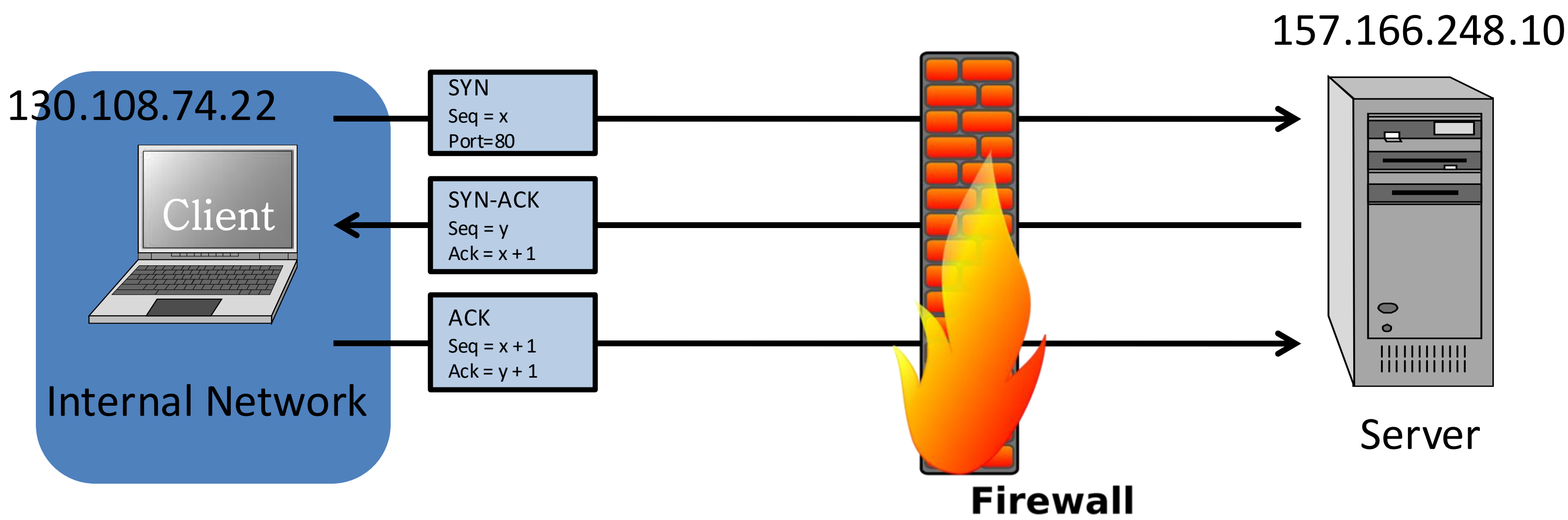
Stateless Firewall



Stateful Firewall

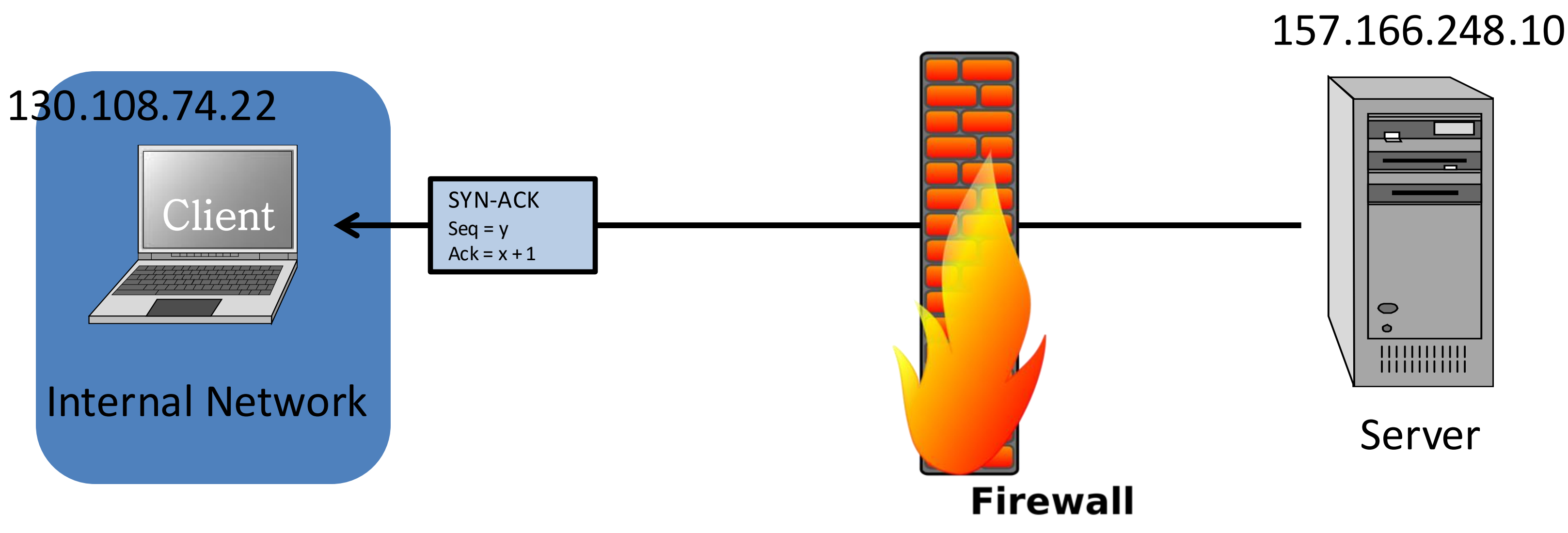
- Objective
  - Allow internal hosts to establish TCP connections with external web services (e.g., port 80)
- Rules
  - Allow outbound SYN and ACK packets whose destination port is 80
  - Allow inbound SYN-ACK packets that are in response to a connection initiated from hosts in the internal network

Stateful Firewall



Internal IP	Internal Port	External IP	External Port	Proto	SYN	ACK	Est?

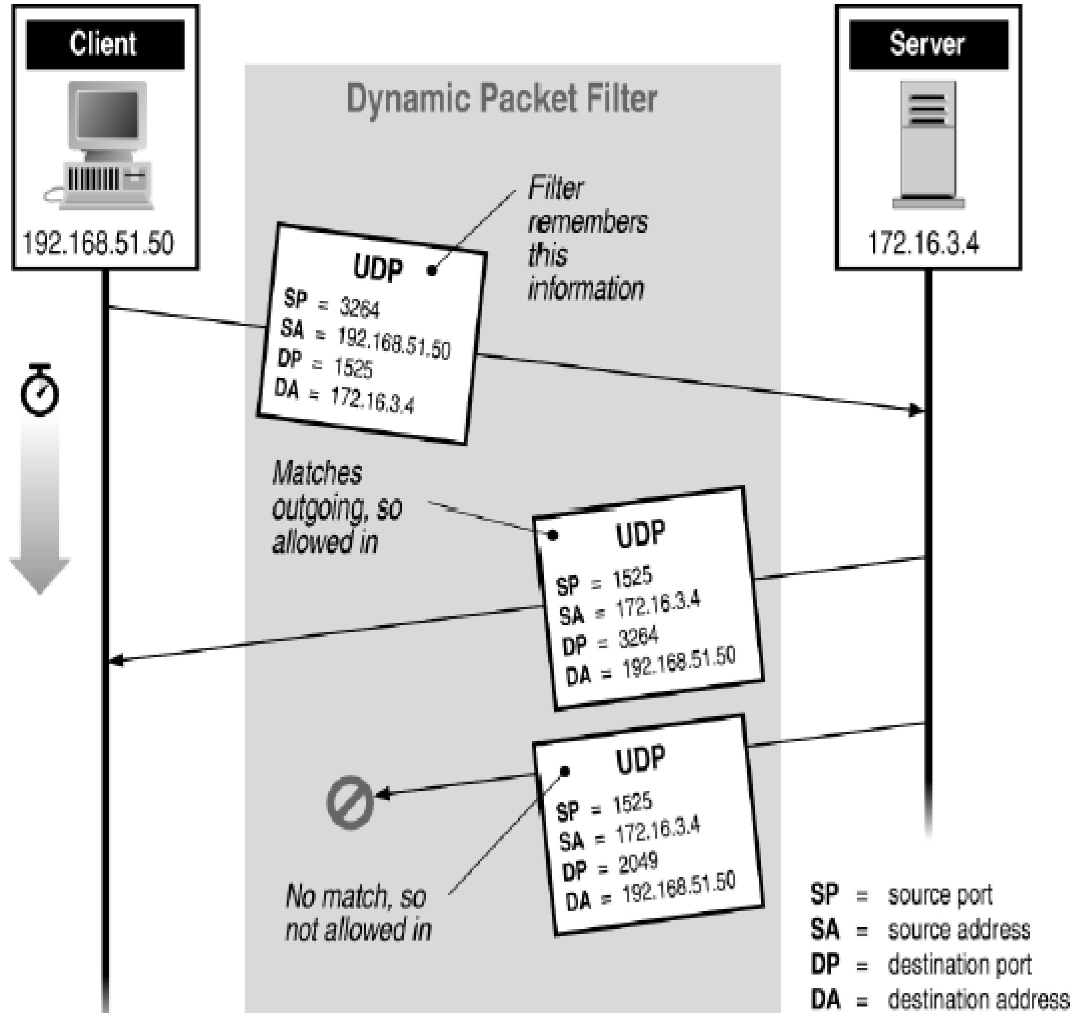
Stateful Firewall



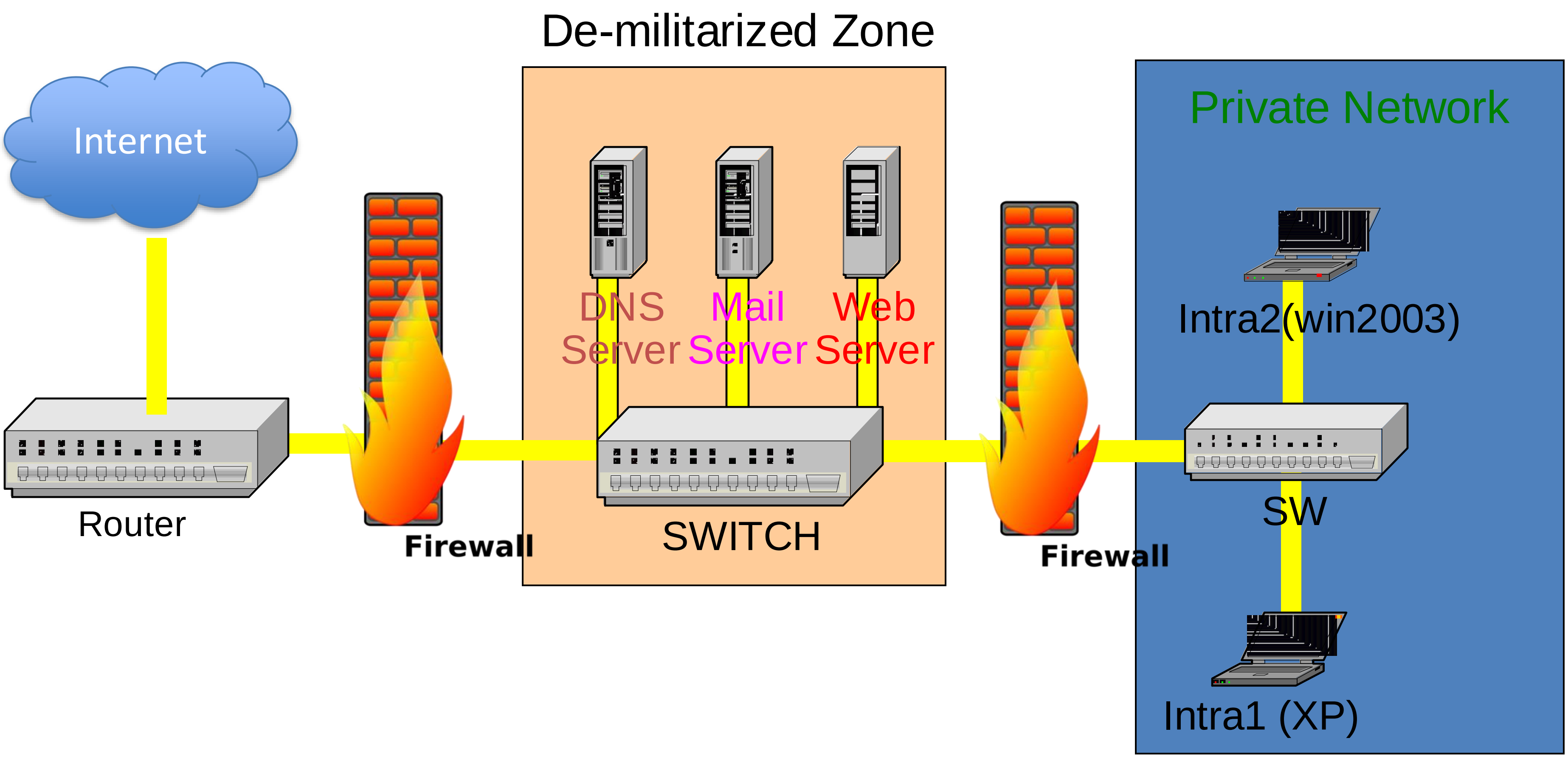
Internal IP	Internal Port	External IP	External Port	Proto	SYN	ACK	Est?

Stateful Firewall

- Objective
  - Allow internal hosts to establish UDP connections with external services



Typical Deployment of Firewalls for an Enterprise Network



Packet Filter - Example

Examples

This example shows how to build a fundamental packet filter set for SMTP based traffic:
Scenario 1: Allowing inbound and outbound SMTP (sending and receiving electronic mail). Our initial packet filter rule set would be:

Rule	Direction	Src Address	Dest Address	Protocol	Dest Port	Action
A	In	External	Internal	TCP	25	Permit
B	Out	Internal	External	TCP	>1023	Permit
C	Out	Internal	External	TCP	25	Permit
D	In	External	Internal	TCP	>1023	Permit
E	Either	Any	Any	Any	Any	Deny

Rule A and B allow inbound SMTP connections (incoming email).
Rule C and D allow outbound SMTP connections (outgoing email).
Rule E is the default rule that applies if all else fails.

[2]

Packet Filter - Example

Our host has IP address 172.16.1.1. Someone is trying to send us email from a remote host with IP address 192.168.3.4. The sender's SMTP client uses port 1234 to talk to our SMTP server, which is on port 25. This example filtered through our rule set would produce the following results:

Packet	Direction	Src Address	Dest Address	Protocol	Dest Port	Action
1	In	192.168.3.4	172.16.1.1	TCP	25	Permit (A)
2	Out	172.16.1.1	192.168.3.4	TCP	1234	Permit (B)

Scenario 2: Outgoing mail would adhere to these rules:

Packet	Direction	Src Address	Dest Address	Protocol	Dest Port	Action
3	Out	172.16.1.1	192.168.3.4	TCP	25	Permit (C)
4	In	192.168.3.4	172.16.1.1	TCP	1357	Permit (D)

[2]

Packet Filter - Example

Scenario 3: Someone from the outside world (10.1.2.3) attempts to open a connection from port 5150 on his or her end to the web proxy server on port 8080 on one of your internal systems (172.16.3.4) in order to carry out an attack. This would look like this:

Packet	Direction	Src Address	Dest Address	Protocol	Dest Port	Action
5	In	10.1.2.3	172.16.3.4	TCP	8080	Permit (D)
6	Out	172.16.3.4	10.1.2.3	TCP	5150	Permit (B)

This attack could succeed because your original filter set rules B and D allow all connections where both ends are using ports above 1023. Do not assume that each rule or group of rules is okay, because then the whole rule set is okay. To correct such oversights, you need to consider the entire rule set as a whole.

- Attack succeeds because of rules B and D
- More secure to add source ports to rules

Packet Filter - Example

Rule	Direction	Source Address	Dest Address	Protocol	Source Port	Dest Port	Action
A	In	External	Internal	TCP	>1023	25	Permit
B	Out	Internal	External	TCP	25	>1023	Permit
C	Out	Internal	External	TCP	>1023	25	Permit
D	In	External	Internal	TCP	25	>1023	Permit
E	Either	Any	Any	Any	Any	Any	Deny

Applying this new rule set, here are the same six sample packets from our three scenarios:

Rule	Direction	Source Address	Dest Address	Protocol	Source Port	Dest Port	Action
1	In	192.168.3.4	172.16.1.1	TCP	1234	25	Permit (A)
2	Out	172.16.1.1	192.168.3.4	TCP	25	1234	Permit (B)
3	Out	172.16.1.1	192.168.3.4	TCP	1357	25	Permit (C)
4	In	192.168.3.4	172.16.1.1	TCP	25	1357	Permit (D)
5	In	10.1.2.3	172.16.3.4	TCP	5150	8080	Deny (E)
6	Out	172.16.3.4	10.1.2.3	TCP	8080	5150	Deny (E)

[2]

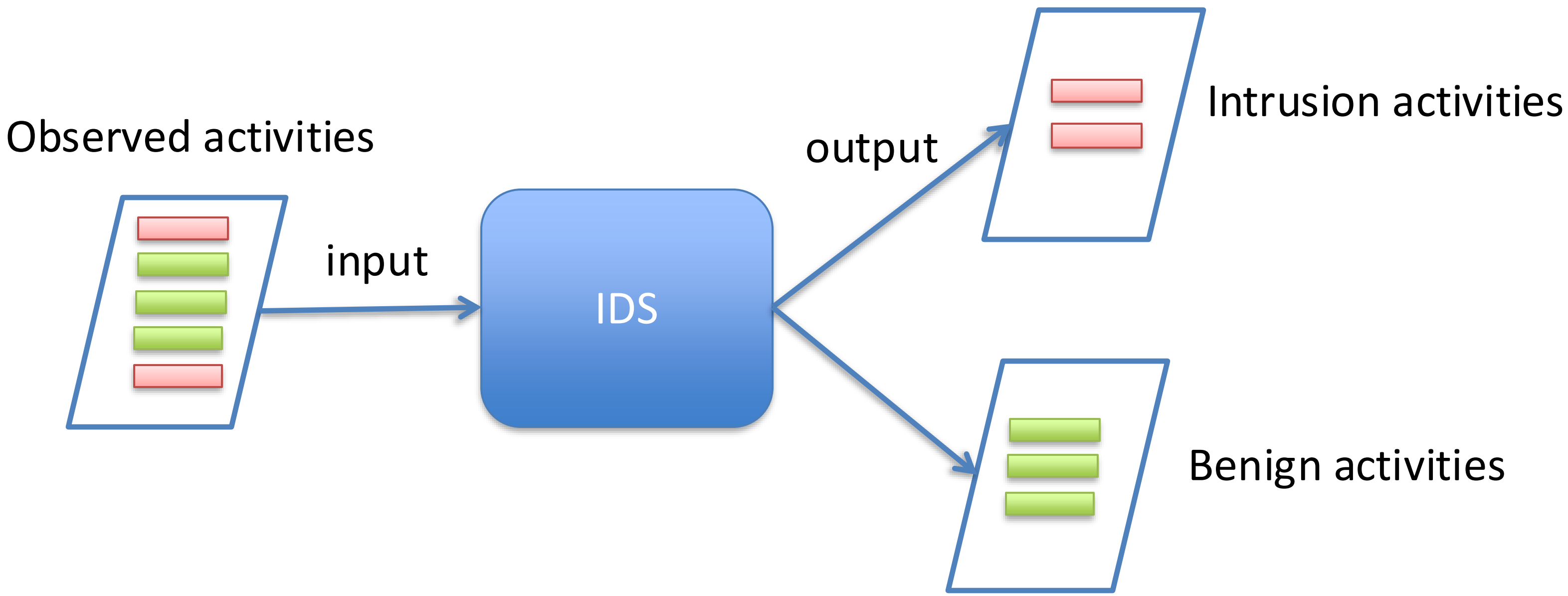
Coming up: Packet Filter - Example

Intrusion Detection

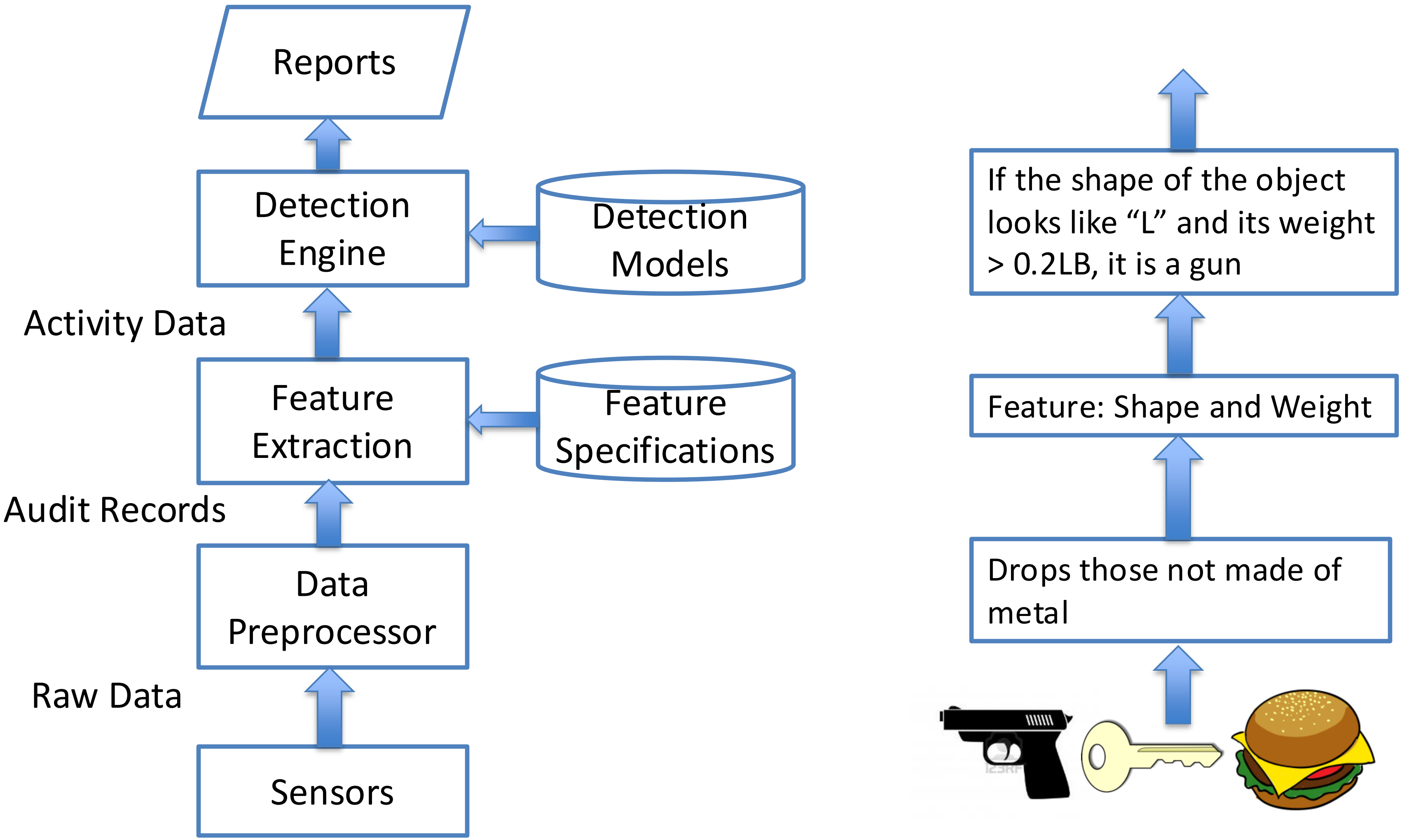
- Intrusion ≈ Attacks
  - Activities whose objective is to compromise the security properties of a target system
    - Scanning to discover vulnerable services
    - Exploiting the discovered vulnerability
    - Running malicious logic in a target system
    - Attempting to break into a computer by trying a large number of passwords
- Intrusion Detection
  - Detecting intrusion activities

A High-Level Overview of IDS

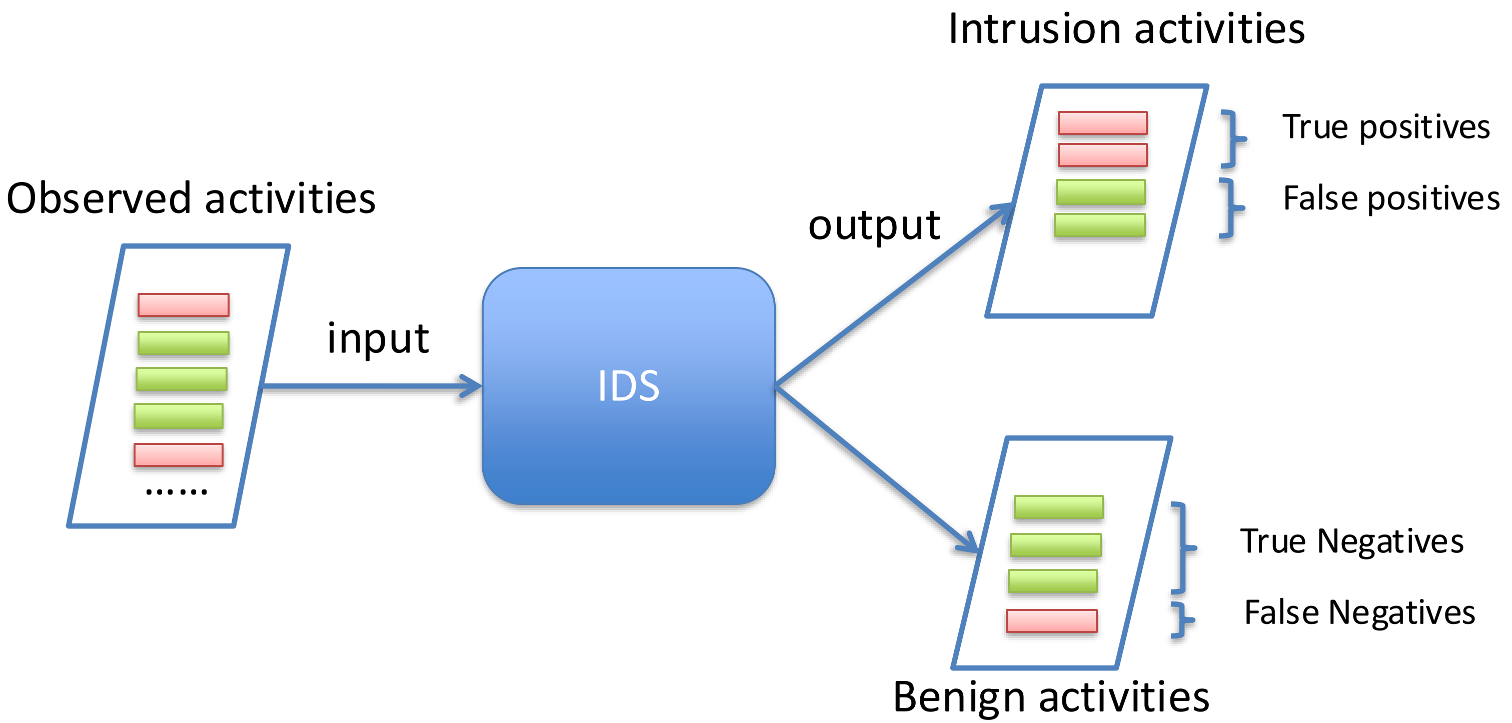
- Intrusion Detection System (IDS)
  - The system that performs intrusion detection



Typical Components of an IDS



Detection Accuracy



- False Positives (FP): Benign activities that are falsely identified as attacks
- False Negatives (FN): Intrusion activities that are falsely identified as benign activities
- True Positives (TP): Intrusion activities that are correctly identified as malicious activities
- True Negatives (TN): Benign activities that are correctly identified as benign activities

Detection Rate: TP/(TP+FN)

False Positive Rate: FP/(FP+TN)

Key Metrics of IDS

- Effectiveness
  - High detection rates
  - low false positive rates
- Efficiency
  - The throughput of IDS, the scalability?
  - The impact on the system performance?
- Resilience to attacks
  - IDS can be the target of attackers
  - How hard is it for an attacker to evade the IDS?

Categories of IDSs

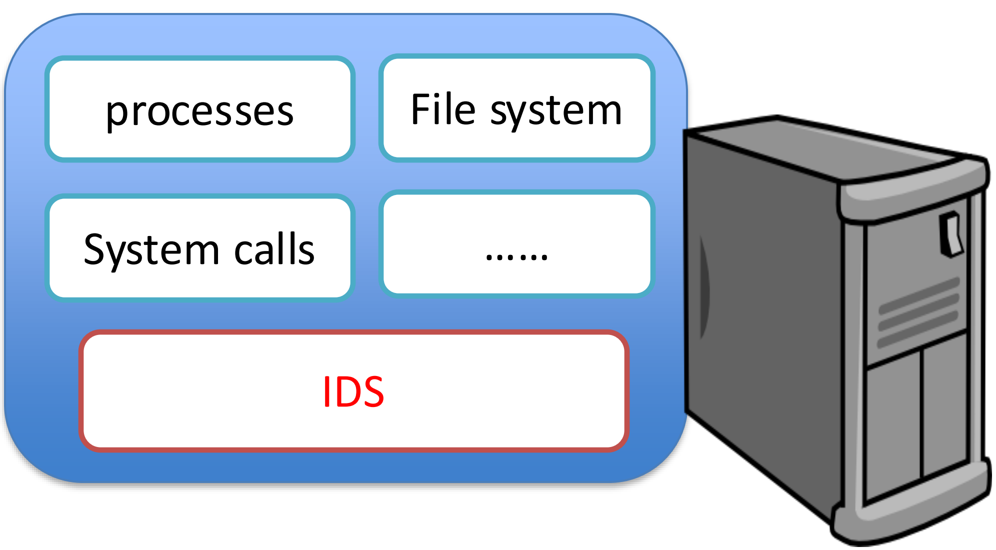
- How to detect
  - Misuse
  - Anomaly
  - Hybrid (e.g., learning-based)

Categories of IDSs

- Where to deploy
  - Host-based IDS (HIDS)
  - Network-based IDS (NIDS)

Host-based IDS

- Using OS auditing mechanisms
  - Logs all direct or indirect events generated by a user (e.g., BSM on Solaris)
  - Analyzes system calls made by a program (e.g., strace)
- Monitoring user activities
  - Analyzes shell commands

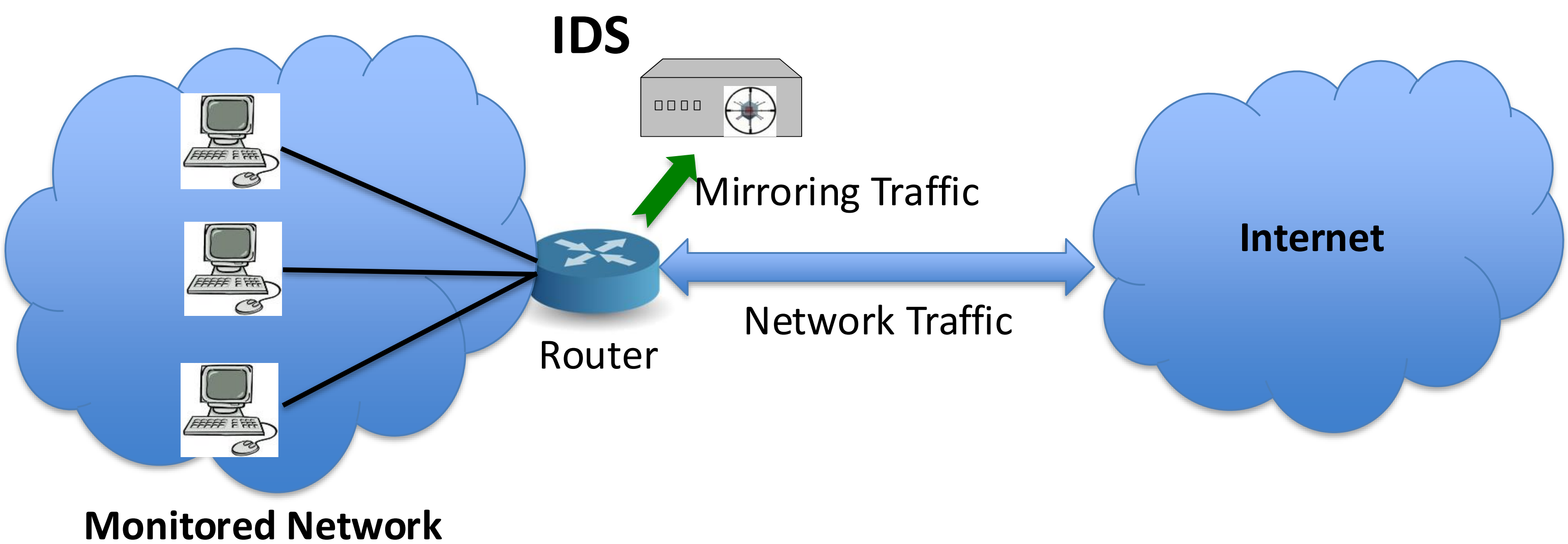


Host-based IDS

- Advantages
  - Fine-grained knowledge about system-level behaviors
- Disadvantages
  - Operational cost: need to install IDS on all user machines
  - Limited visibility: ineffective for large-scale attacks
    - An attacker scan all hosts; for each host, he scans once
  - Can be contaminated by attackers/malware

Network-based IDS

- Monitor network-level behaviors
  - E.g., network packets across a network boundary
  - E.g., network flow logs generated by a router
  - E.g., DNS queries and responses collected from campus DNS servers



Network-based IDS

- Advantages
  - Operation cost is low
    - One IDS can monitor behaviors for all hosts in the network
  - Higher visibility of all hosts' behaviors
    - Facilitate correlation among different hosts
  - Less likely to be contaminated by attackers
    - Isolated from compromised hosts
- Disadvantages
  - Coarse-grained knowledge about the system-level behavior
    - E.g., it is hard for NIDS to obtain the plaintext for encrypted data

Misuse Detection

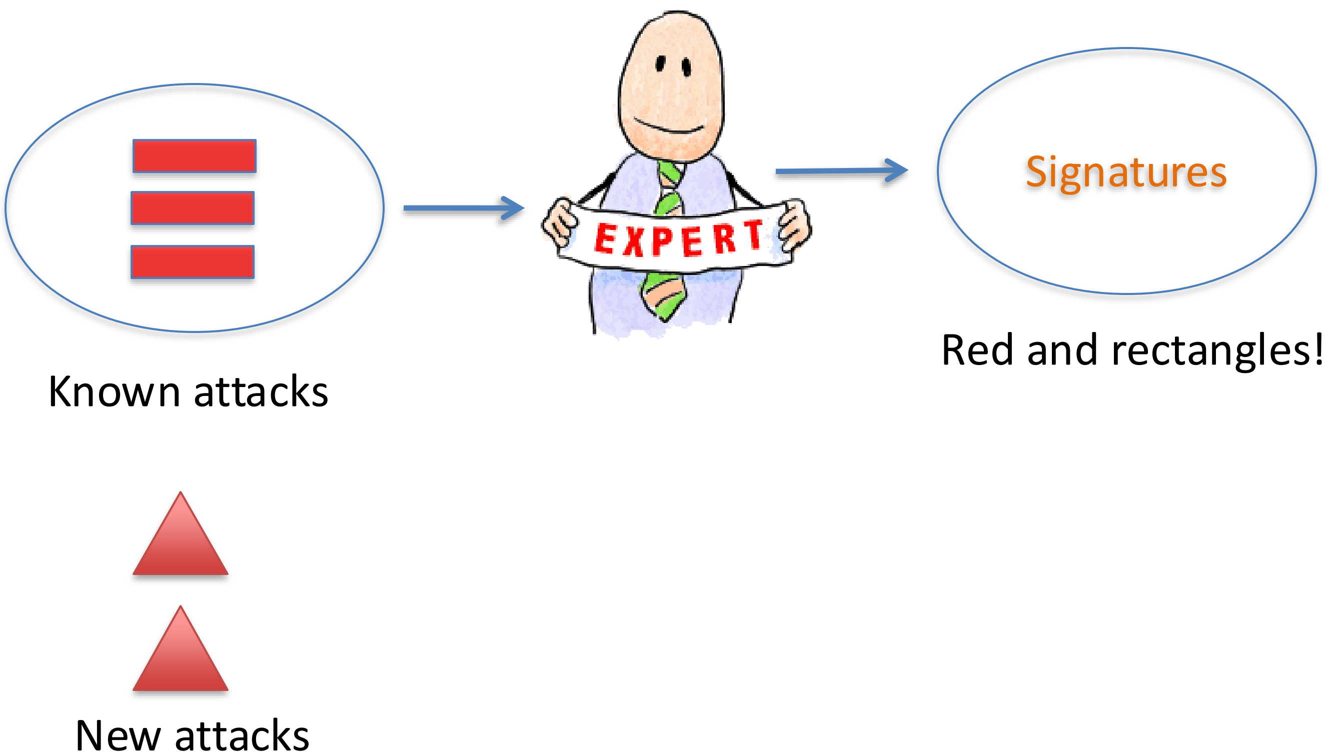
- Define the profile of known attacks
  - Any activities that are consistent with the profile will be labeled as attacks
  - Otherwise, they are labeled as benign activities

Known attacks

Red and rectangle!

Misuse Detection

- Disadvantage
  - Cannot detect new attacks (a.k.a, high false negatives for new attacks)



Anomaly Detection

- Define the profile of benign activities
  - Any activities that are consistent with the profile will be labeled as benign
  - Otherwise, they are labeled as attacks

Benign Samples

Any behavior that deviates from the normal profile will be considered as anomaly

Hybrid Detection

- Characterize the profiles of benign activities and intrusion activities, respectively
  - E.g., detection system based on statistical classifier

Misuse V.S. Anomaly

- Advantage
  - Relatively low false positive rate in practical deployment
  - Offers information for reaction
- Disadvantage
  - Cannot detect novel attacks (e.g., an variant of a malware sample)

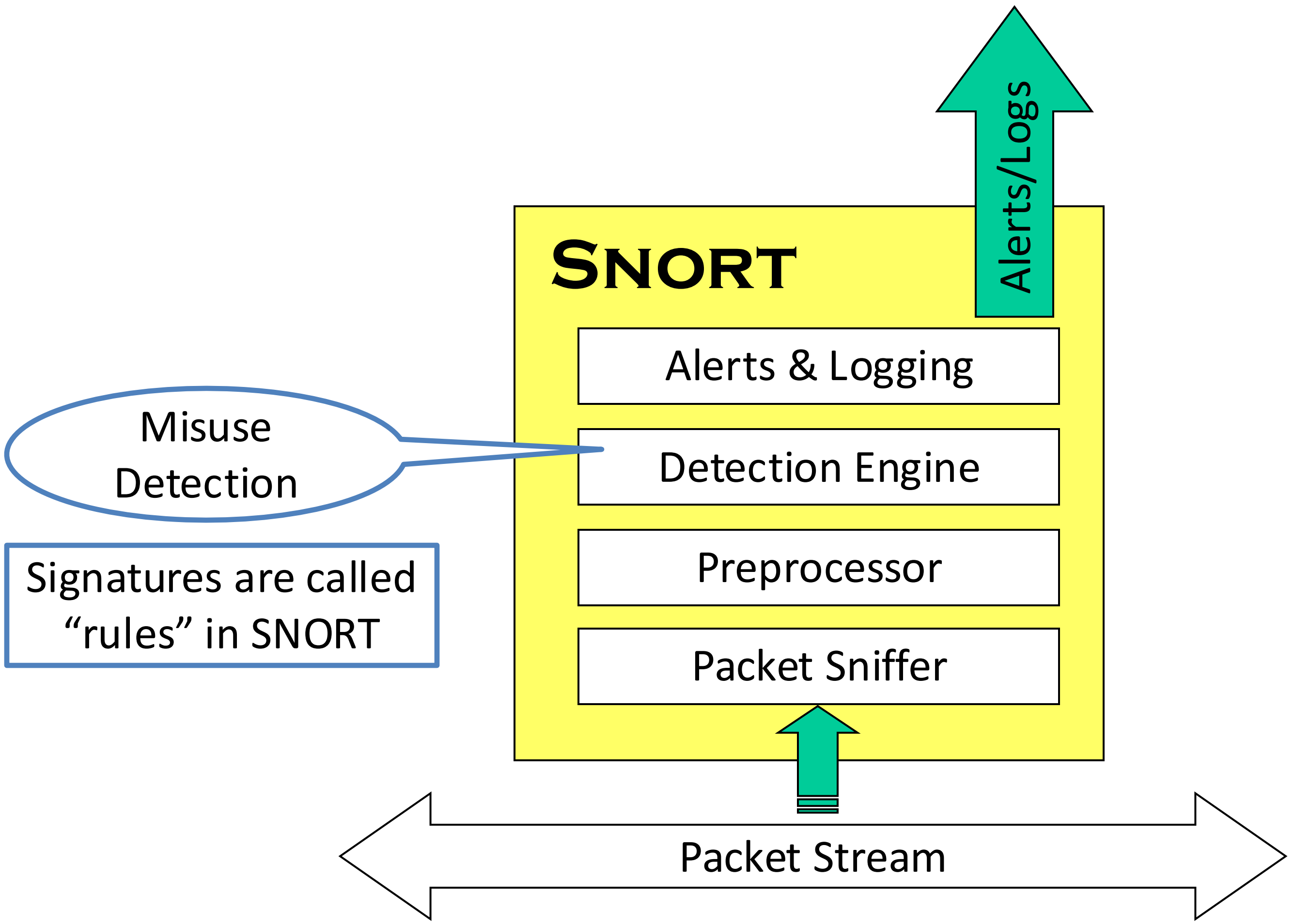
Misuse V.S. Anomaly

- Advantage
  - Is promising to detect new attacks
- Disadvantage
  - Relatively high false positive rate in practical deployment
  - Offers little information for reaction

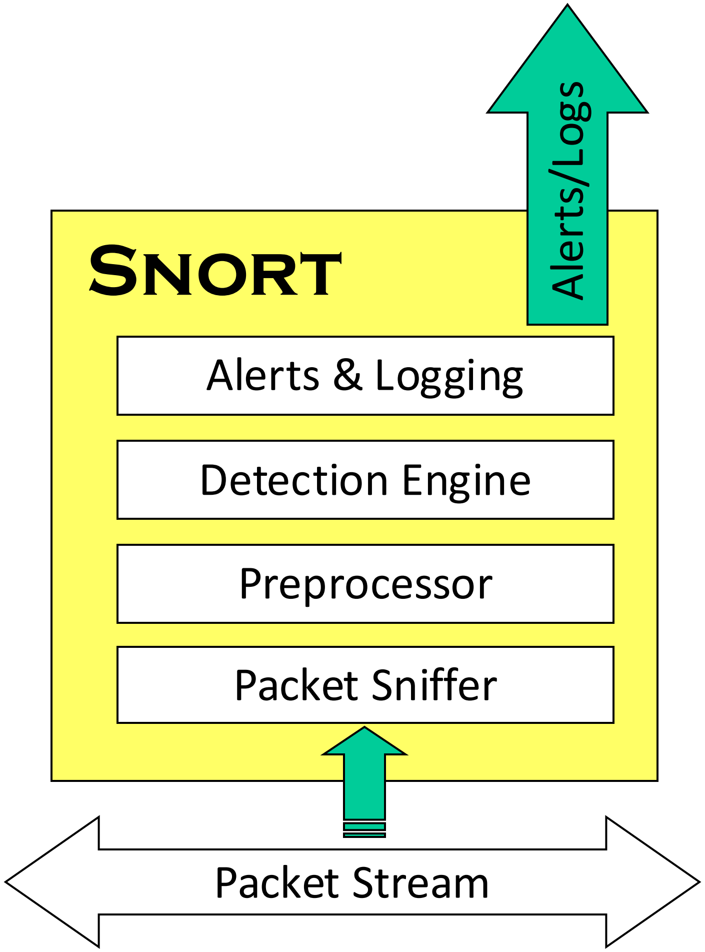
SNORT: A Misuse IDS

- What is Snort?
  - Snort is a multi-mode packet analysis tool
    - Sniffer
    - Packet Logger
    - Forensic Data Analysis tool
    - Network Intrusion Detection System

SNORT Architecture

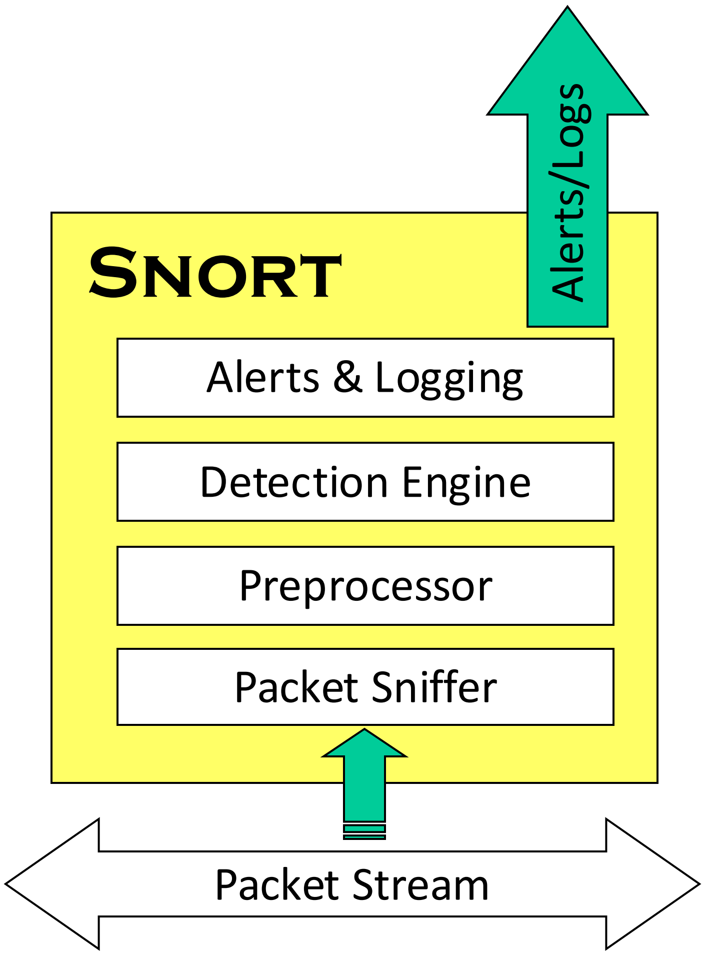


SNORT Architecture



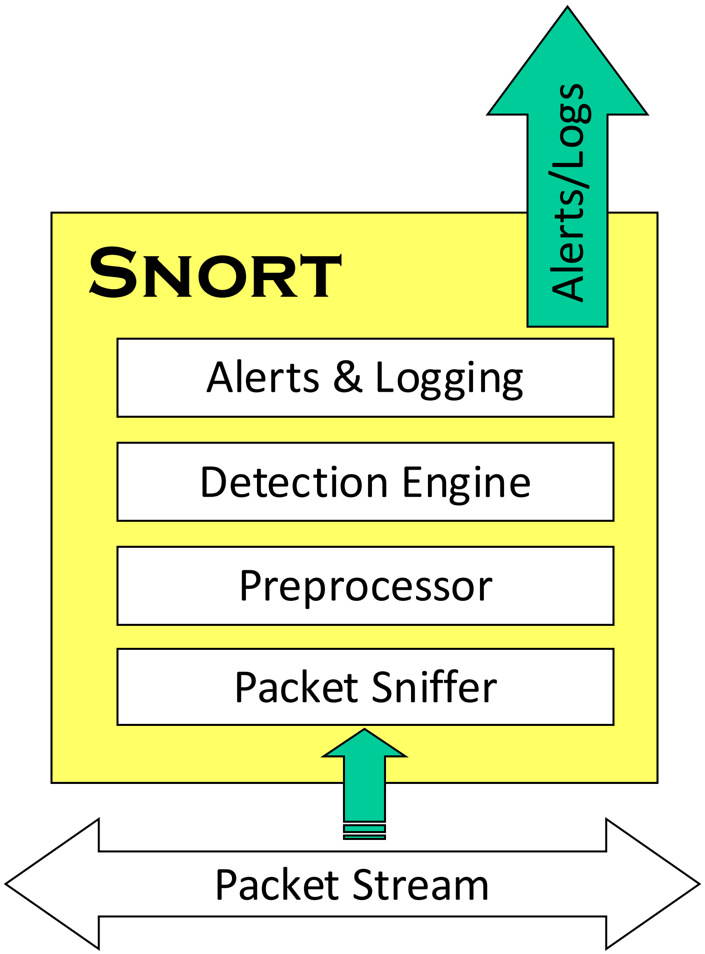
- Packet Sniffer
  - Captures packets from the network, usually using pcap
  - Decodes/parses packets
- Preprocessor
  - A set of plug-ins for
    - E.g., fragment reassemble
    - E.g., integrating anomaly detection capabilities (detect port scanning)

SNORT Architecture



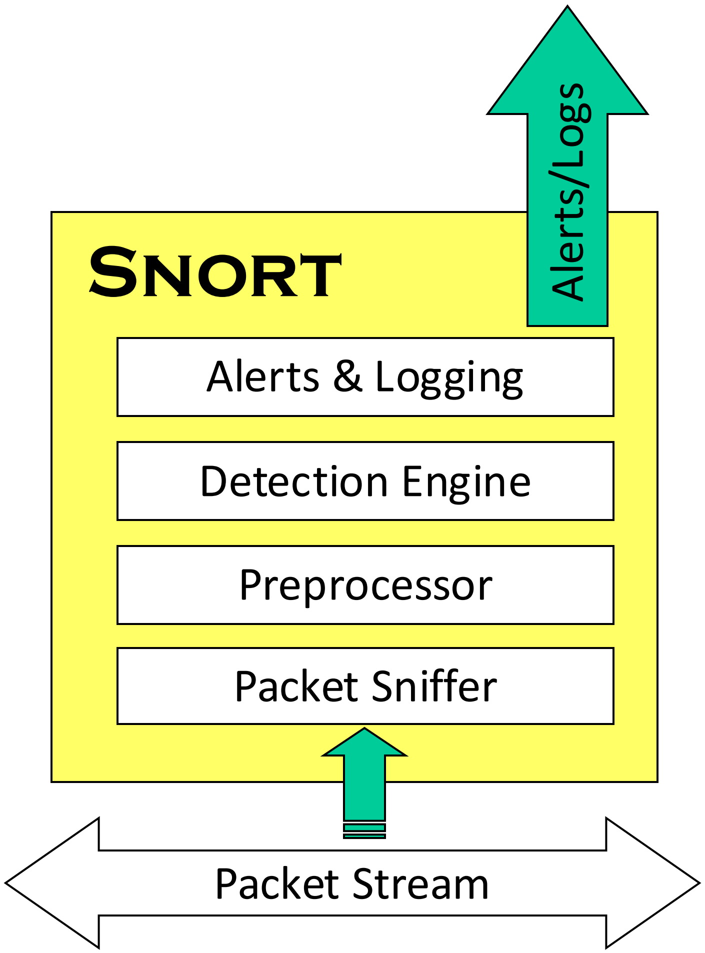
- Detection Engine
  - Misuse-based (signature-based) IDS
  - Signatures are called “rules” in SNORT
  - Rule
    - Header
    - Option

SNORT Architecture



- Alerting and Logging
  - Various plugins to format alerts
    - E.g., html-based output, email alerts, and etc.
  - Log "interesting packets"

SNORT Rules



- Rule
  - Header
    - Action to take
    - Type of packet
    - Source, destination IP address
    - ……
  - Option
    - Content of packet that should make the packet match the rule

SNORT Rule Example

alert tcp $External_NET any -> $Home_NET 21 (msg: “ftp Exploit”; flow_to_server, established; content: “|31c031db 41c9b046 cd80 31c031db|”; reference: bugtraq,1387; classtype:attempted-admin; sid 344; rev:4;)

SNORT Rule Example

alert tcp $External_NET any -> $Home_NET 21 (……)

- Rule Header
  - Alert / log / pass / dynamic / activate
  - tcp: Protocol being used. UDP / IP / ICMP
  - $External_NET: This is the source IP, default is any.
  - any: This is the source port set to “any”
  - ->: Direction of connection (e.g., who initiates the connection?).
  - $Home_NET: This is a variable that Snort will replace with
  - 21: Port to be monitored.
- The header concerns all tcp packages coming from any port from the external network to port 21 on the monitored (internal) network

SNORT Rule Example

…… (msg: “ftp Exploit”; flow_to_server, established; content: “|31c031db 41c9b046 cd80 31c031db|”; reference: bugtraq,1387; classtype:attempted-admin; sid 344; rev:4;)

- Rule Options
  - msg: “ftp Exploit”;
  - flow_to_server, established;
  - content: “|31c031db 41c9b046 cd80 31c031db|”; Snort will look whether the packet contains such content.
  - reference: bugtraq,1387; for identifying information third-party warnings.
  - classtype:attempted-admin;
  - sid 344; the unique identifier for this Snort rule (www.snort.org/snort-db).
  - rev:4; the revision number for this rule.

Anomaly Detection

- Phase I: Training Phase
  - Input: a collection of benign samples
  - Output: a model to describe benign samples (the normal profile)

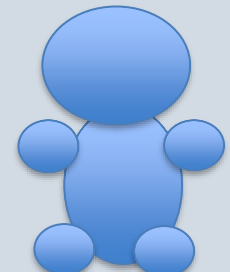
- Phase II: Detection Phase
  - Input: unknown samples that contain both benign samples and malicious samples.
  - Output: attacks (malicious samples)

Anomaly Detection

Training Phase

Benign Samples

The model that describes the normal profile



Deviates from the normal profile?

Detection Phase

unknown samples

Attacks!

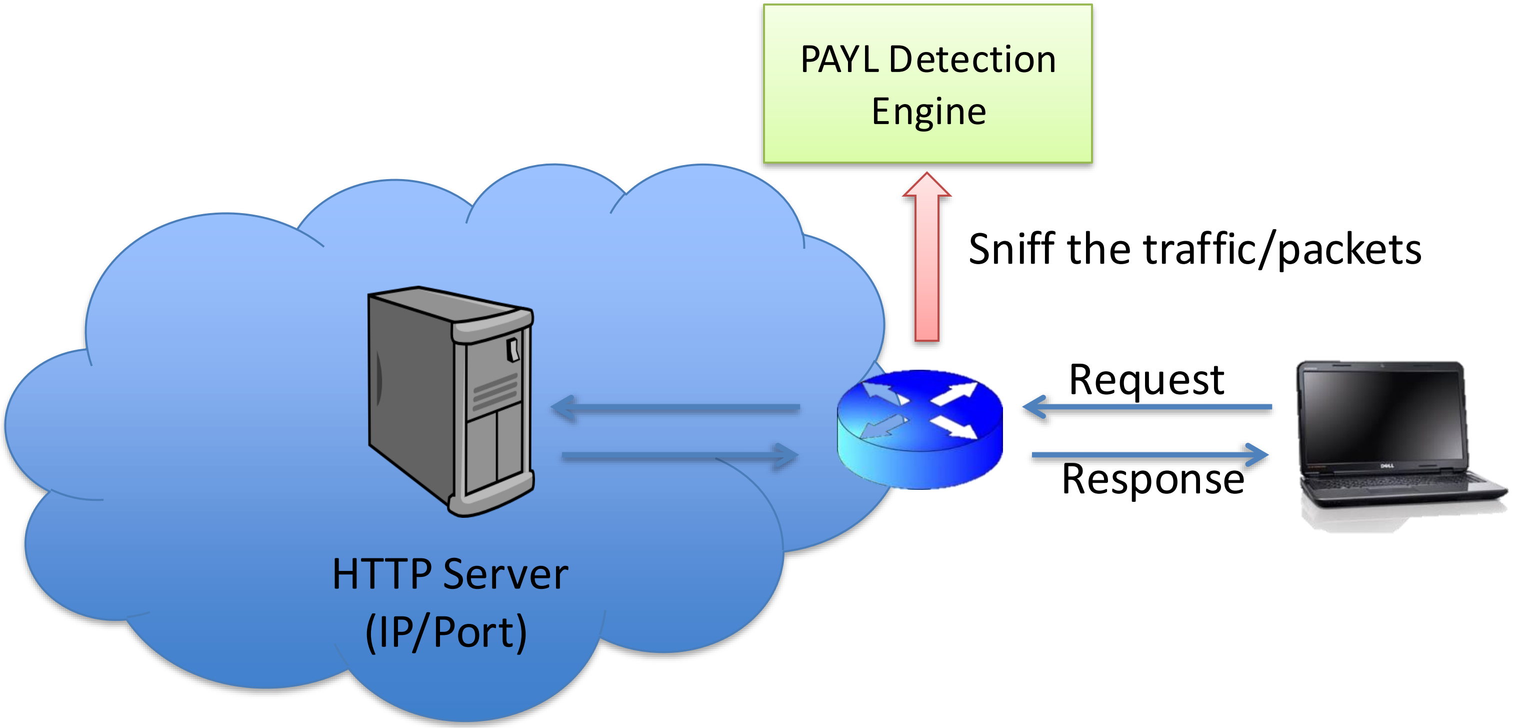
PAYL

- A network-based anomaly detection system
- "***Anomalous Payload-based Network Intrusion Detection***", in the proceedings of RAID 2004.

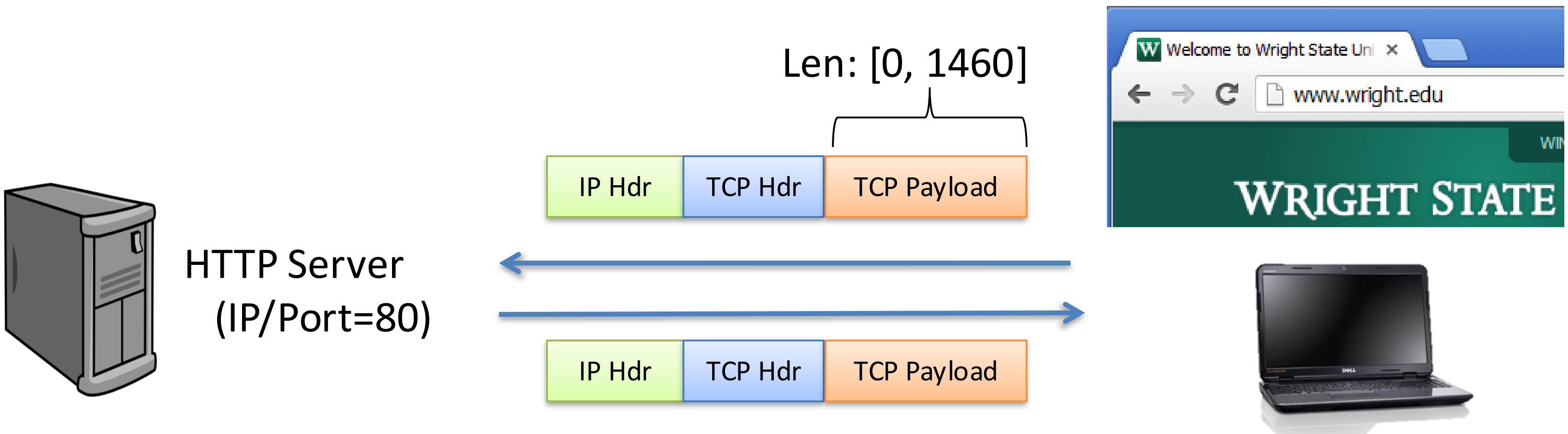
  -Ke Wang and Salvatore J. Stolfo

We will use HTTP service as example in this lecture, but you should be familiar with other services discussed in the paper

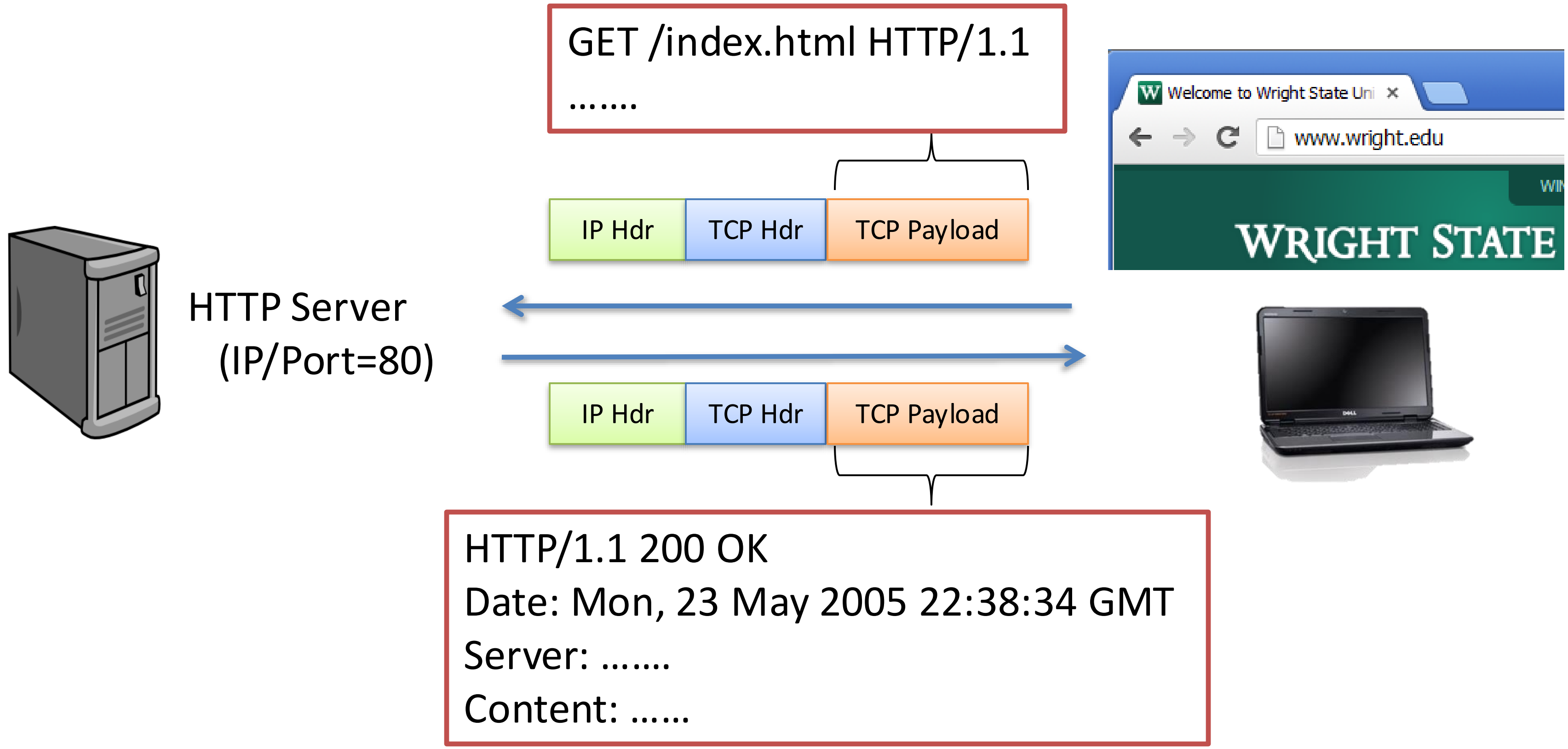
PAYL


PAYL Detection Engine
Sniff the traffic/packets
Request
Response
HTTP Server
(IP/Port)

An Example of HTTP Service



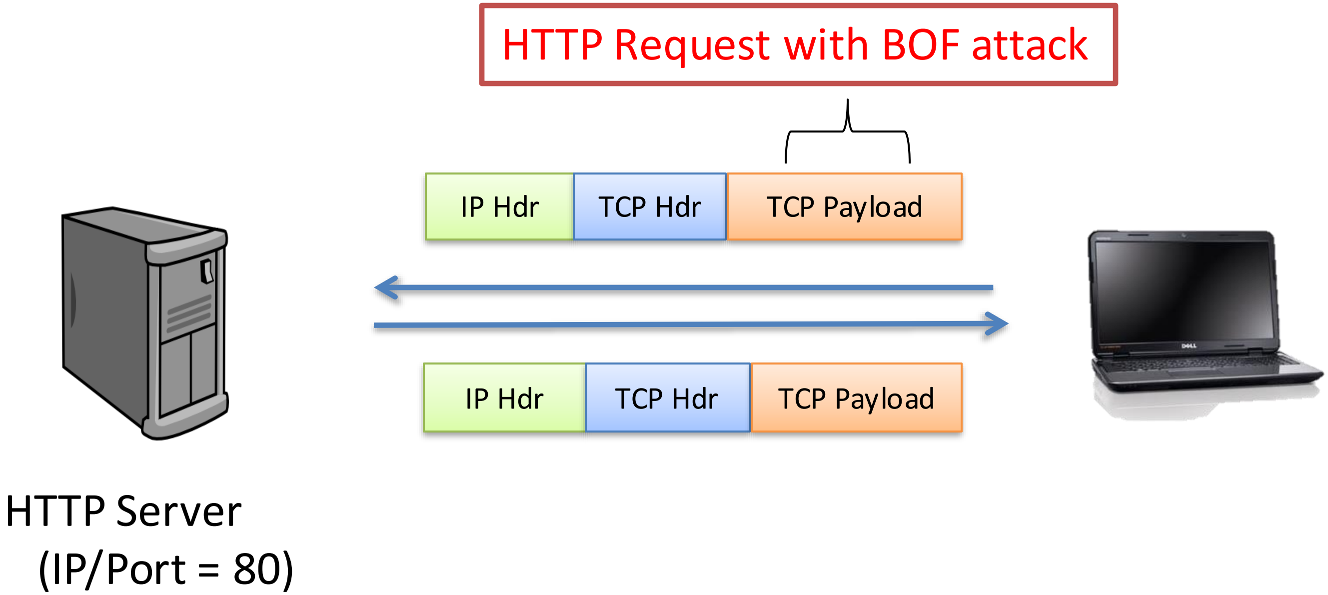
An Example of HTTP Service



An Examp

```
GET./default.ida?XXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX%u9090
%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u
00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
```

Figure 3.3: Raw packet of CRII; only the first 301 bytes are shown for brevity.



Architectural Overview of PAYL

GET /index.html HTTP/1.1

GET /events/page1.html HTTP/1.1

GET /grades?name=JZ&id=001 HTTP/1.1

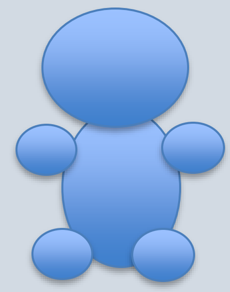
GET /grades?name=ZK&id=003 HTTP/1.1

....

Benign Samples

Training Phase

The model that describes the normal profile



Attack!

```
GET./default.ida?XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX%u9090
%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u
00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
```

Figure 3.3: Raw packet of CRII; only the first 301 bytes are shown for brevity.

Detection Phase

The Simplest Model

- Training Phase:
  - the normal profile = the collection of benign samples we have observed
  - (assumption: all samples in the training phase are benign)

- Detection Phase:
  - If a new sample is contained in the collection, label it as benign
  - Otherwise, label it as attack

- Effective on detecting BOF attacks?
- What is the problem for this model?

PAYL

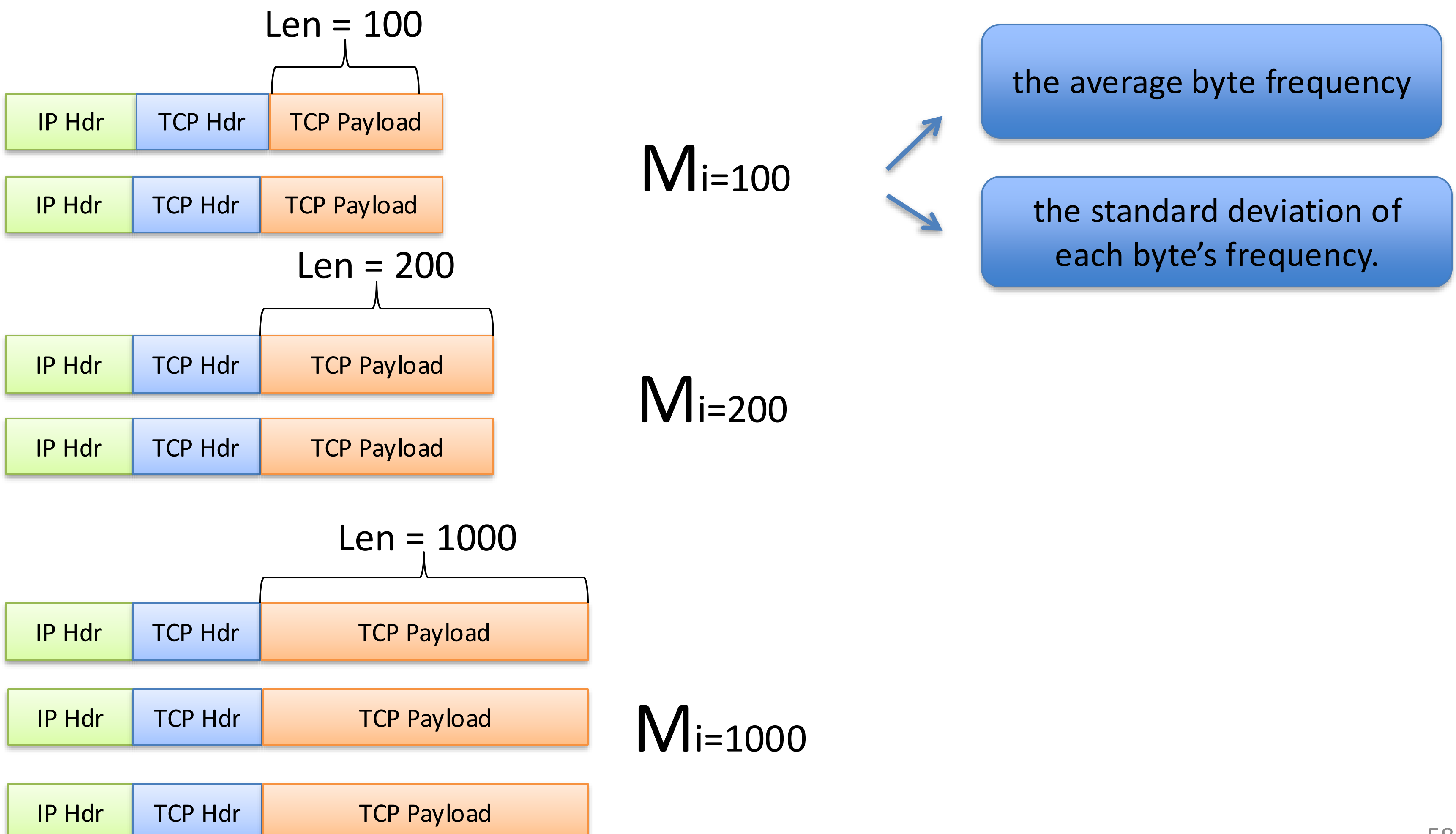
- Intuition
  - The distribution of byte frequency varies drastically between normal HTTP requests and the BOF attacks.
    - Normal HTTP requests:
      - Readable characters
      - Common keywords/strings
      - Perhaps some images
    - BOF attacks
      - NOP
      - Executable instructions/binary

PAYL

- Given a training data set (packet payload (i.e., HTTP Request)), PAYL builds a set of models M_i
  - i: the length of the TCP payload
  - M_i
    - the average byte frequency of n-gram (We will use 1-gram as the example)
    - the standard deviation of each byte's frequency.

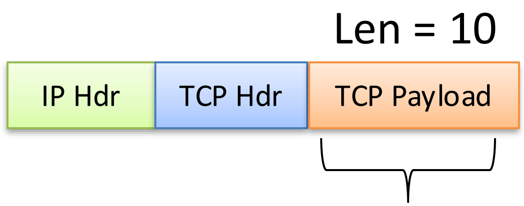
PAYL (using HTTP as the example)

- M_i, 1-gram



PAYL: ***Training***

- For length i (the number of bytes in the TCP payload)



Benign samples	TCP Payload
1	GET /ABBBC
2	GET /ABDEC
3	GET /ACBBE
4	GET /ACEED

	A	B	C	D	E	G	T	\s	/
1	1/10	3/10	1/10	0	1/10	1/10	1/10	1/10	1/10
2	1/10	1/10	1/10	1/10	2/10	1/10	1/10	1/10	1/10
3	1/10	2/10	1/10	0	2/10	1/10	1/10	1/10	1/10
4	1/10	0	1/10	1/10	3/10	1/10	1/10	1/10	1/10

PAYL: *Training*

	A	B	C	D	E	G	T	\s	/
1	1/10	3/10	1/10	0	1/10	1/10	1/10	1/10	1/10
2	1/10	1/10	1/10	1/10	2/10	1/10	1/10	1/10	1/10
3	1/10	2/10	1/10	0	2/10	1/10	1/10	1/10	1/10
4	1/10	0	1/10	1/10	3/10	1/10	1/10	1/10	1/10

$M_{i=10}$

	A	B	C	D	E	G	T	\s	/
Avg Byte Freq	0.1	0.15	0.1	0.05	0.2	0.1	0.1	0.1	0.1
Std Dev of Byte Freq	0	0.13	0	0.06	0.08	0	0	0	0

PAYL: ***Detection***

- Given an unknown sample, we compute the byte frequency for this particular sample.
- Does this significantly deviate from the normal profile?
  - Define a distance to evaluate the deviation

$$d(x, \overline{y}) = \sum_{i=0}^{n-1} (| x_i - \overline{y_i} | / (\overline{\sigma_i} + \alpha))$$

The normal profile: $M_{i=10}$

	A	B	C	D	E	G	T	\s	/
Avg Byte Freq	0.1	0.15	0.1	0.05	0.2	0.1	0.1	0.1	0.1
Std Dev of Byte Freq	0	0.13	0	0.06	0.08	0	0	0	0

unkown samples	TCP Payload
1	GET /ABBBC
2	GET /ABEEC
3	GET /XXXXX

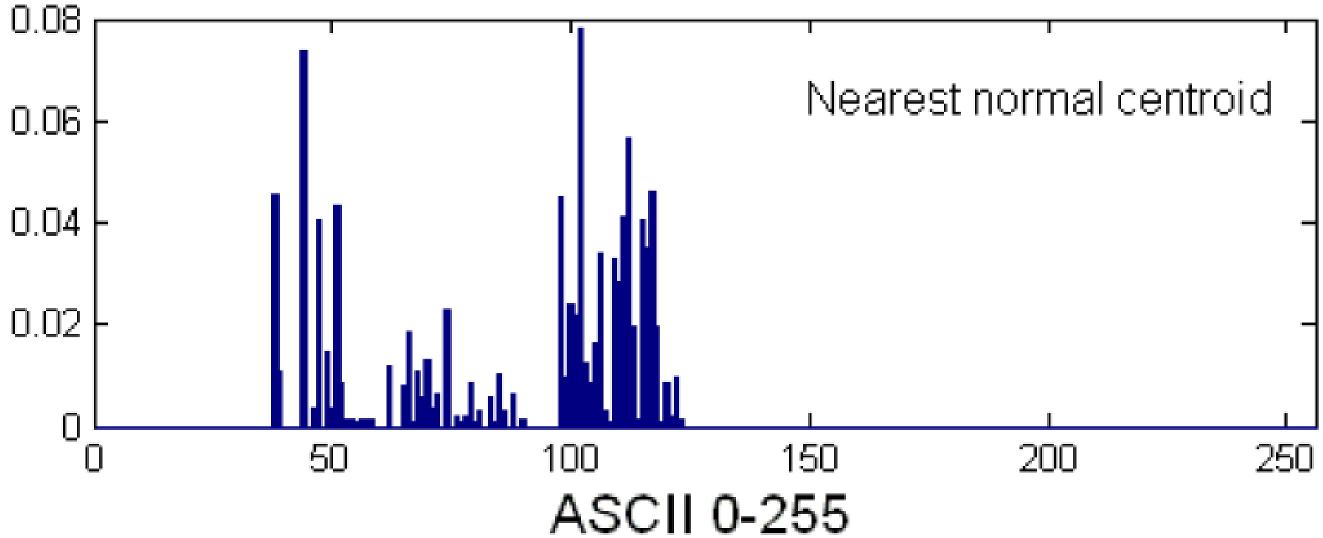
Unknown samples	A	B	C	D	E	G	T	X	\s	/
1	0.1	0.3	0.1	0	0.1	0.1	0.1	0	0.1	0.1
2	0.1	0.1	0.1	0	0.3	0.1	0.1	0	0.1	0.1
3	0	0	0	0	0.1	0.1	0.1	0.5	0.1	0.1

$$d(x,\overline{y}) = \sum_{i=0}^{n-1} \underbrace{(|x_i - \overline{y_i}|/(\overline{\sigma_i} + \alpha))}$$

Unknown samples	A	B	C	D	E	G	T	X	\s	/	D, given a=0.01
1	0/a	0.15/(0.13+a)	0/a	0.05/(0.06+a)	0.1/(0.08+a)	0/a	0/a	0/a	0/a	0/a	2.9
2	0/a	0.05/(0.13+a)	0/a	0.05/(0.06+a)	0.1/(0.08+a)	0/a	0/a	0/a	0/a	0/a	2.2
3	0.1/a	0.15/(0.13+a)	0	0.1/a	0.1/(0.08+a)	0/a	0/a	0.5/a	0/a	0/a	72.2

PAYL

- Model for $M_{i=1360}$ (the std dev of byte frequency is omitted for brevity)

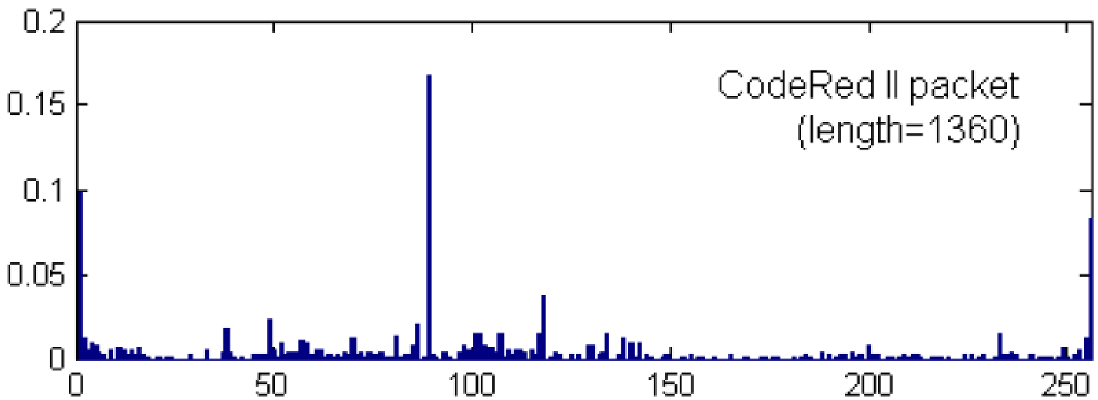


PAYL

- TCP Payload for Code Red II (len = 1360)

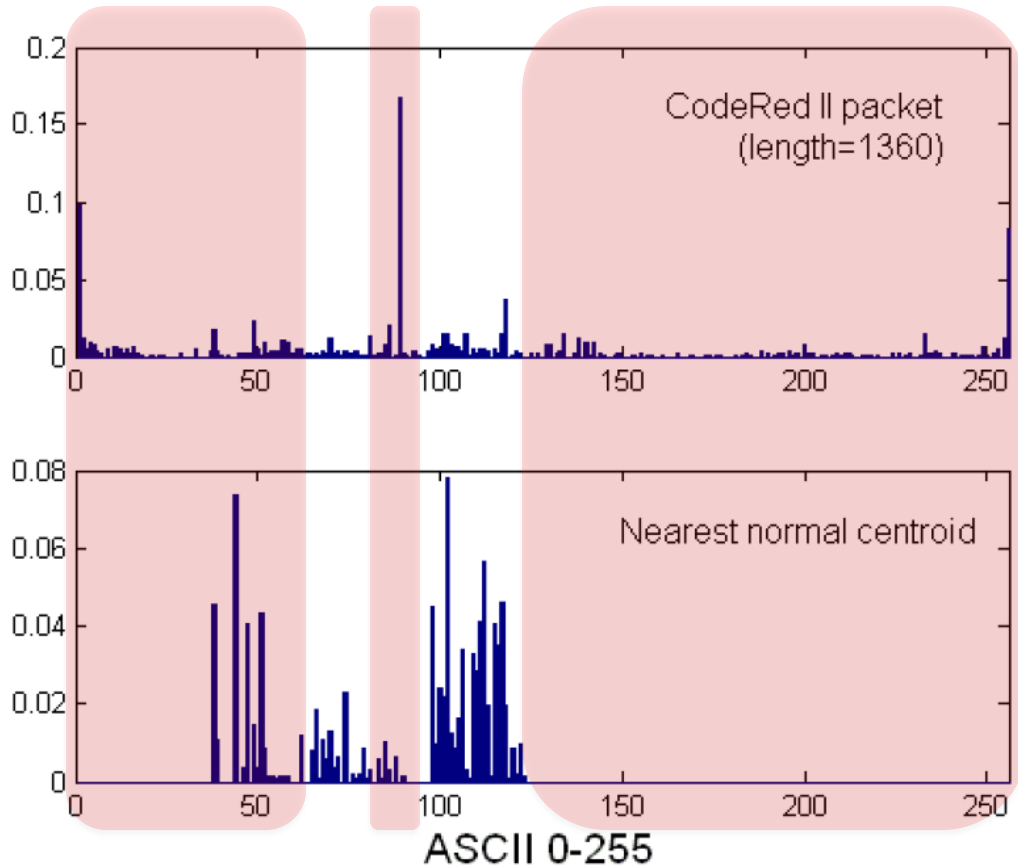
```
GET./default.ida?XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX%u9090
%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u
00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
```

Figure 3.3: Raw packet of CRII; only the first 301 bytes are shown for brevity.

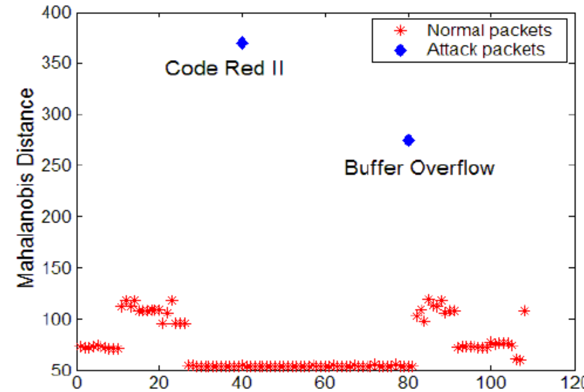


PAYL

- Code Red II v.s. the Normal Profile $M_{i=1360}$

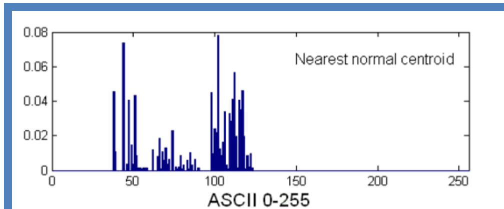


PAYL: Experiments on Detection



std dev of byte freq

$M_{i=1360}$



std dev of byte freq

Get distance (deviation from the normal profile)

$$d(x,\overline{y}) = \sum_{i=0}^{n-1}(|x_i - \overline{y_i}|/(\overline{\sigma_i}+\alpha))$$

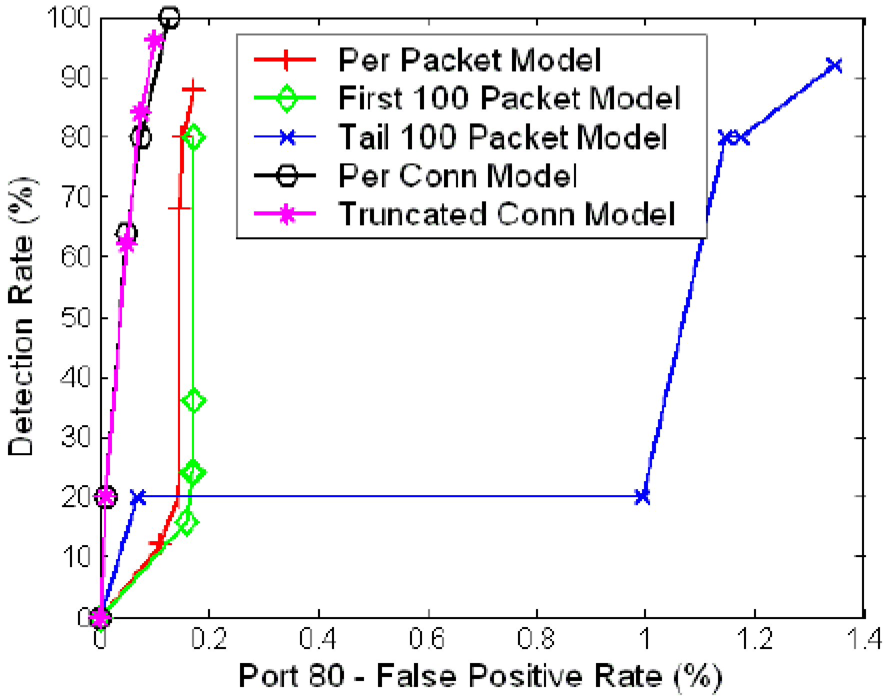
Unknown Samples

IP	TCP	Code Red II
IP	TCP	BOF
IP	TCP	TCP Payload
IP	TCP	TCP Payload
IP	TCP	TCP Payload

Around 120 benign samples

PAYL

- False Positive Rate v.s. False Negative Rate
  - Tune the threshold

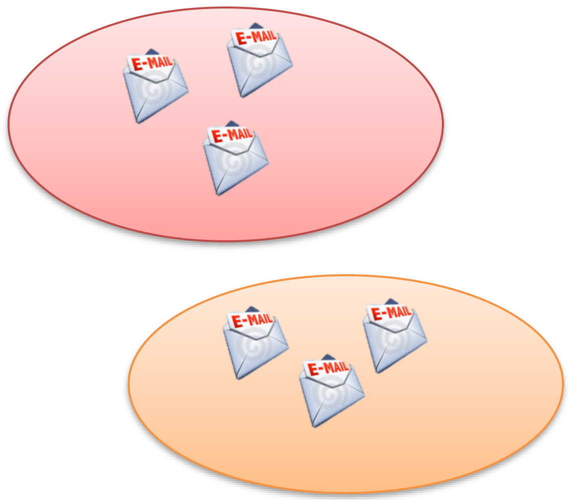


Hybrid Detection

- Characterize the profiles of benign activities and intrusion activities, respectively
- Spam Detection as an Example
  - detection system based on statistical classifier

Statistical Classifier in A Nutshell

Given:



Ham

Spam

Target:

Ham or Spam?

Statistical Classifier in A Nutshell

- Represent an email using a feature vector

The features are designed by YOU!

[f1, f2, f3..., fn]

Feature	Description
f1	Has URL?
f2	Short message? (< 100 words)
f3	Commercial words? (free, money, click…)

Hi all,

Click www.malware.com for greeting card. Free! Free! Free!

Sam

[f1 = true, f2 = true, f3 = true]

Statistical Classifier in A Nutshell

- Learn a classifier

Ham

Spam

	f1	f2	f3	Spam
email1	True	True	True	YES
email2	True	False	True	YES
…	….	….	….	….
$email_M$	False	True	False	NO

Learn a classifier

F($email_X$): the confidence of $email_X$ (represented by a feature vector) being spam

[f1 = true, f2 = true, f3 = true]

[f1 = false, f2 = false, f3 = false]

F($email_X$)

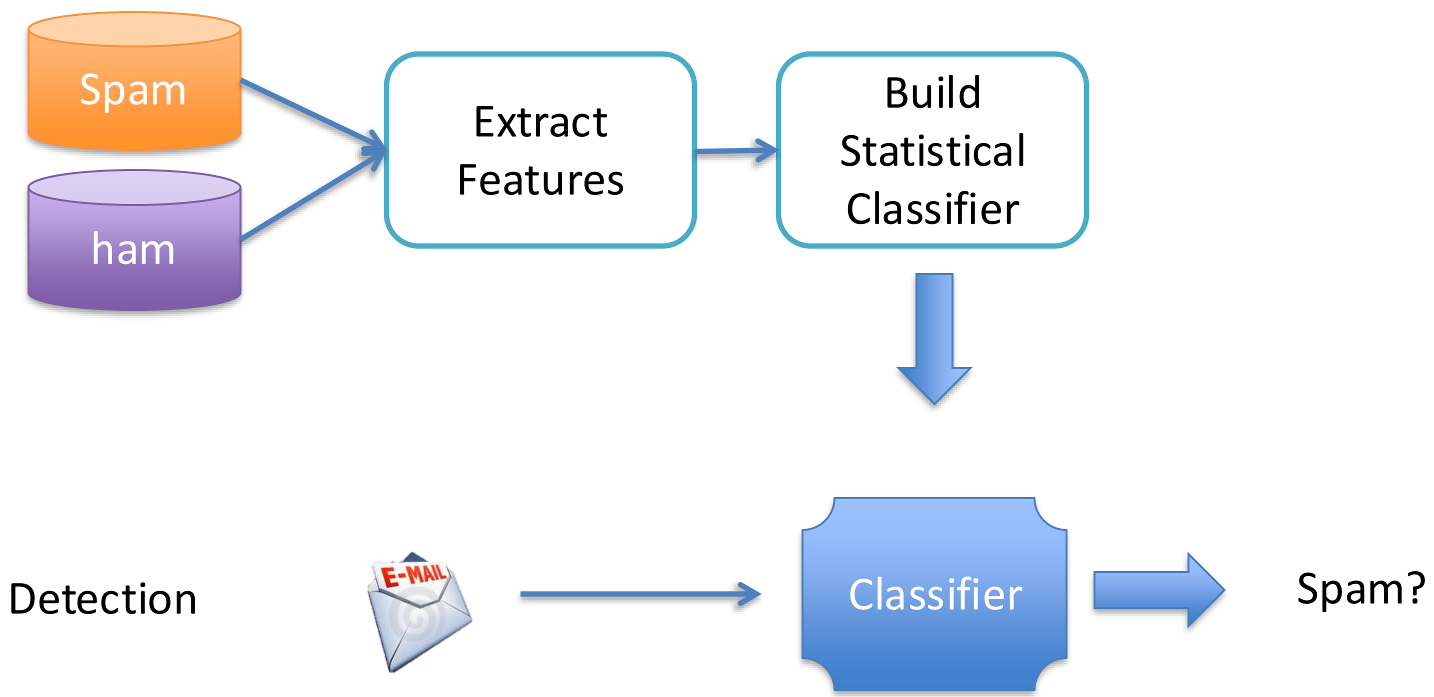
if ($email_X$.f1 == "True" || $email_X$.f3 == "True"))

output 1;

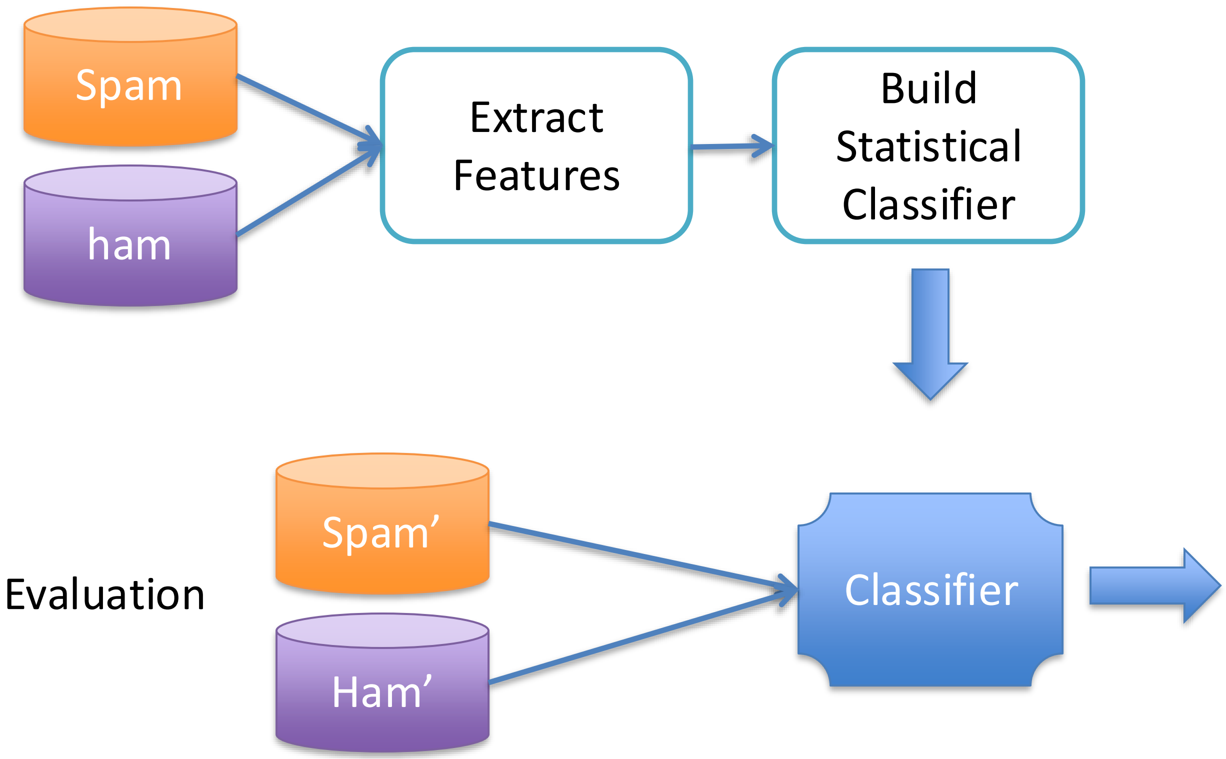
otherwise

output 0

Overview


Spam
ham
Extract Features
Build Statistical Classifier
Detection
E-MAIL
Classifier
Spam?

Overview



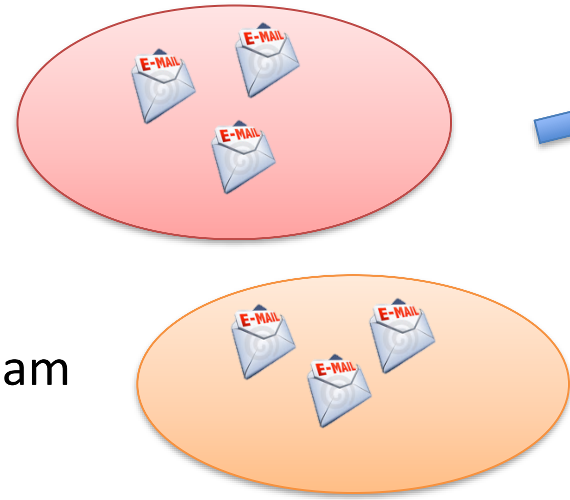
	Detected as Spam
Spam’	Detection Rate
Ham’	False Positive Rate

Statistical Classifier in A Nutshell

- Learn a classifier

Ham

Spam



	f1	f2	f3	Spam
email1	True	True	True	YES
email2	True	False	True	YES
…	….	….	….	….
$email_M$	False	True	False	NO

Learn a classifier

F(email$_X$): the confidence of email$_X$ (represented by a feature vector) being spam

Example: Naïve Bayesian

Tool: Weka, a package for machine learning tools

An Example

- “Detecting Fake Anti-Virus Software Distribution Webpages”
  - Dae Wook Kim, Peiying Yan, Junjie Zhang, Journal of Computers and Security, Nov. 2014

Fake-Antivirus Software

Fig. 1 – An example screenshot of a fake AV webpage.

Data Collection and Labeling

- Data Sources
  - Search Engines
  - What keywords used for searching?
    - Wordstream (a third-party word set) + a random word
- Tools
  - Instrumented Browsers
    - Relationships between different webpages

Data Collection and Labeling

- Labeling
  - Step 1
    - Manual Analysis
    - Does this webpage encourage you to download a anti-virus software system?
  - Step 2
    - If it is true for step 1, consult public domain reputation system.
  - Step 3
    - If it is true for step 1, download the binary and check it using public malware detection servcie

Data Collection and Labeling

- Benign
  - Security-Popular websites: 210
  - Security-Unpopular websites: 17,530
  - Security-Irrelevant: 538
- Malicious (fake A-V software distribution)
  - 1230

System Architecture

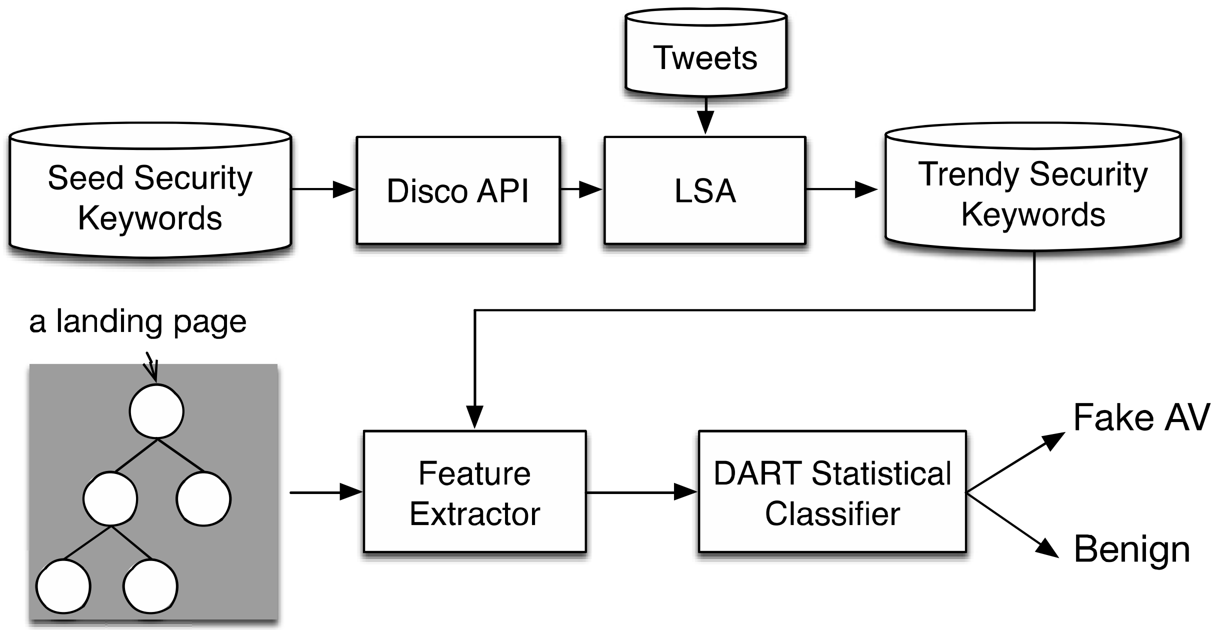
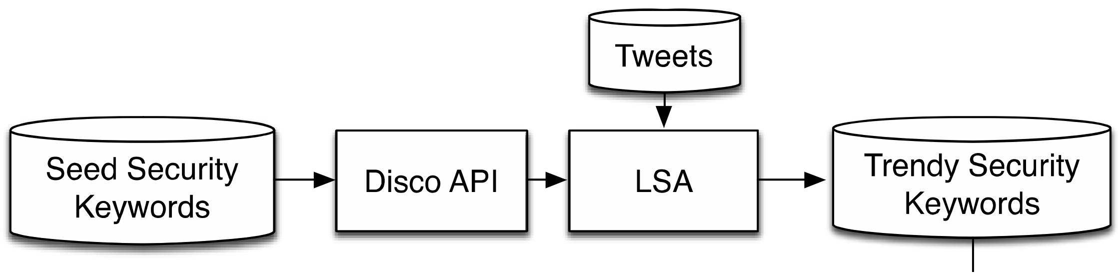


Fig. 2 – System architecture of DART.

Discovering Trendy and Diverse Security Keywords



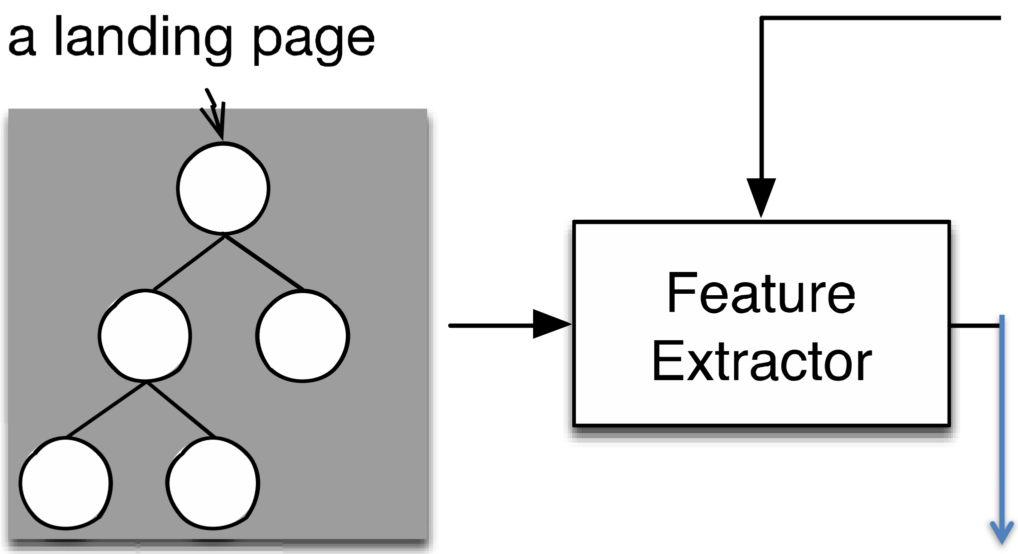
- Step 1
  - Input: Security keywords in OpenDirectoryProject
  - Operation: DISCO API
  - Output: Diverse security keywords
- Step 2
  - Input: output from Step 1 + 10.9 million tweets
  - Operation: LSA (latent semantic analysis)
  - Output: Trendy and diverse security keywords

Discovering Trendy and Diverse Security Keywords

Table 1 – Examples of security keywords discovered in each step.

Seed security keywords	Expanded by DISCO	Expanded by LSA
Security	Firewall	Cybercrook
Antivirus	Encryption	Typosquatting
……	Anti-spyware	RogueAntivirus
	Anti-spam	Zombiecomputer
	Rootkits	Maladvertising
	Backdoor	Snoopware
	Privacy	Ransomware
	……	KeyBoy
		……

Feature Extraction



	Feature 1	Feature 2	Feature 3
Landing page 1	XXX?	XXX?	XXX?

Categories of Features

- Human-Perception Features
  - Trick you into installing the binary
- Search Engine Optimization Features
  - Stay on the top of the search engine results
- Networking Features
  - Staying resilient against disruption

CDF

Definition

The cumulative distribution function of a random variable X is the function

$$F(x) = P(X \leq x).$$

The cumulative distribution function gives the probability that the variable takes a value less than or equal to x and is defined for all real x .

If f is the probability mass function of a discrete random variable X with range $\{x_1, x_2, \ldots,\}$ and F is its cumulative distribution function, then

$$F(x) = \sum_{x_i \leq x} f(x_i).$$

Human-Perception Features

- ***Feature 1: image identity***. This feature quantifies the extent to which the images loaded by a landing page are similar to those images of the authentic anti-virus webpages. Specifically, *DART* aggregates all images loaded by a landing page into a set denoted as V_{web}. Given a set of images (V_{auth}) composed of authentic anti-virus logos or icons, *DART* enumerates each pair of images v_i and v_j where $v_i \in V_{web}$ and $v_j \in V_{auth}$, and then computes their similarity score using an image similarity function denoted as *similar*(). The visual identity feature for a landing page is represented by the maximum value of similarity scores.

Human-Perception Features

Fig. 3 — Examples of icons from authentic AV software.

Fig. 4 — Image similarity between an authentic McAfee image (Left) and fake one (Right).

Method:
Normalized RGB-based Histogram +
Bhattacharyya Measure

Feature 1

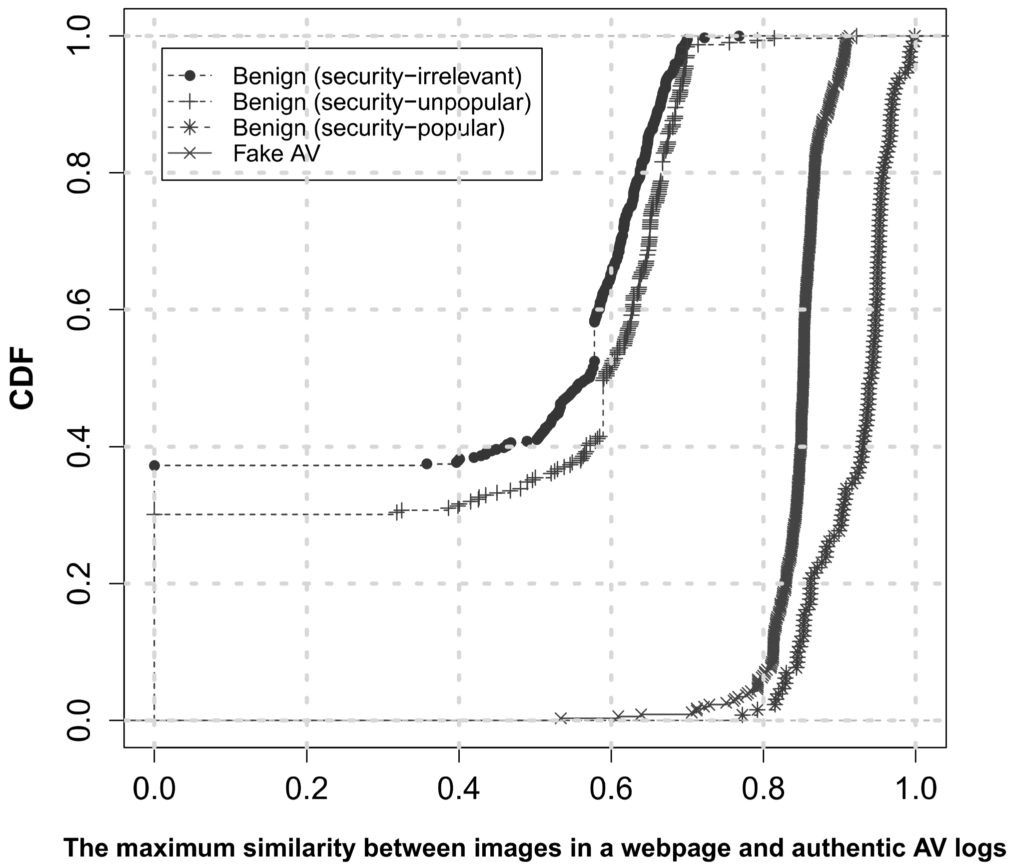


Fig. 5 – Image identity feature.

Human-Perception Features

- ***Feature 2: domain identity***. We split the domain name of a visited landing page into tokens by the delimiter ".", where each token defines a single level of the domain name and the level increases from right to left. The rightmost level is corresponding to the 1-level domain (a.k.a., top-level domain). We identify all tokens that contain any word in $W_{security}$ and then accumulate their levels as the value for this feature. For instance, the domain name for a fake AV website "www.norton-antivirus.en.softonic.com" contains textual identities "norton" and "antivirus" in the $4th$ level domain, resulting in a value of 8 for its feature.

Human-Perception Features

Table 2 – Domain examples for authentic and fake anti-virus landing pages.

Categories	Domain examples
Authentic AV	http://us.norton.com http://www.pandasecurity.com http://www.avast.com http://www.avg.com
Fake AV	http://norton-antivirus.en.softonic.com http://panda.brothersoft.com http://avast.softpedia.com http://avg-antivirus-free.soft32.com

Feature 2

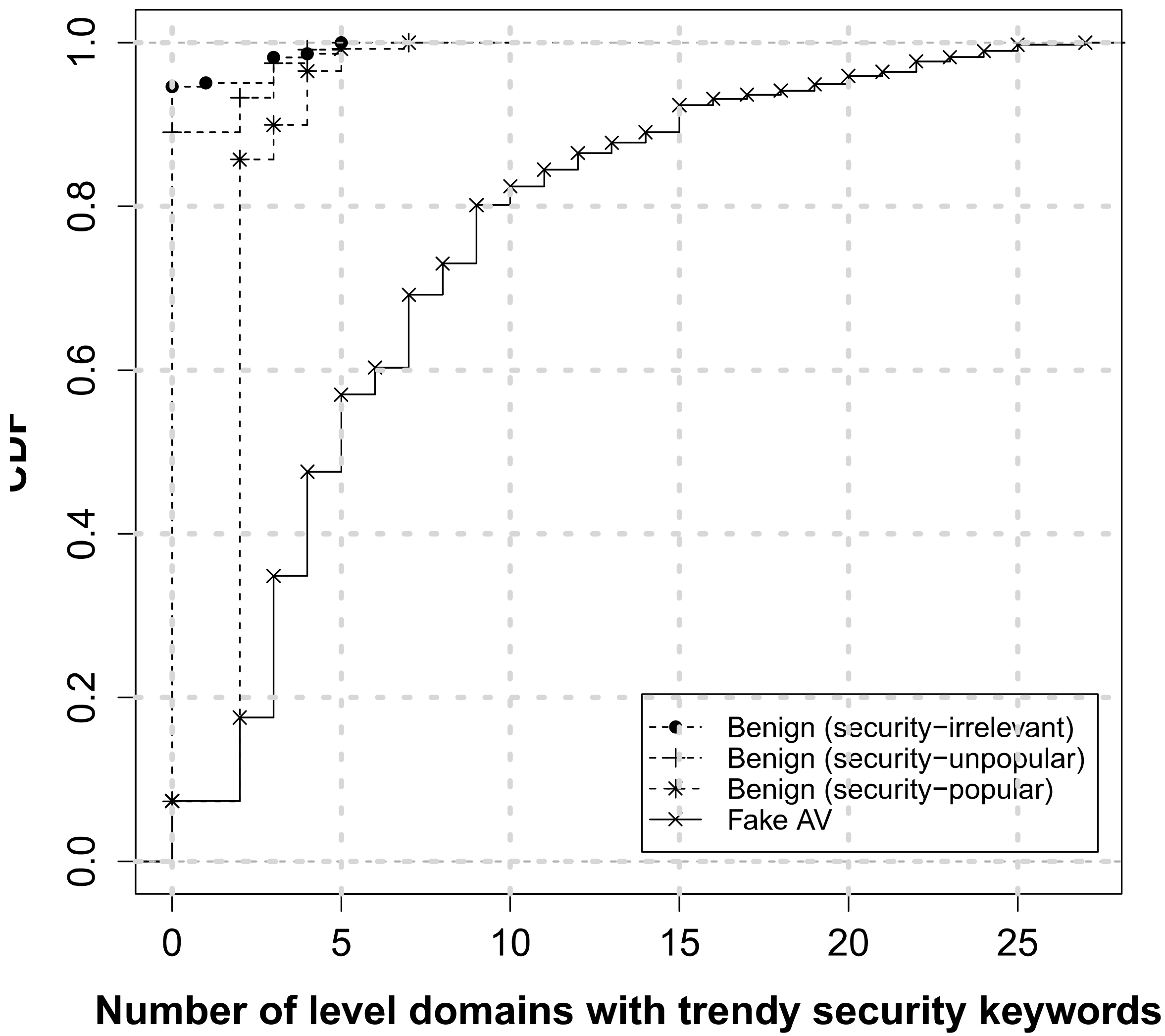


Fig. 6 – Domain identity feature.

Human-Perception Features

- ***Feature 3: content identity***. Although we can directly extract security keywords from preserved source codes for all webpages associated with a landing page, many security keywords are dynamically generated (e.g., by JavaScript) or actually presented in images. Therefore, *DART* performs Optical Character Recognition (OnlineOCR, 2013) analysis on the snapshot of the fully loaded landing page and extract all words. The value of this feature represents the total occurrence of words that belong to $W_{security}$.

Feature 3

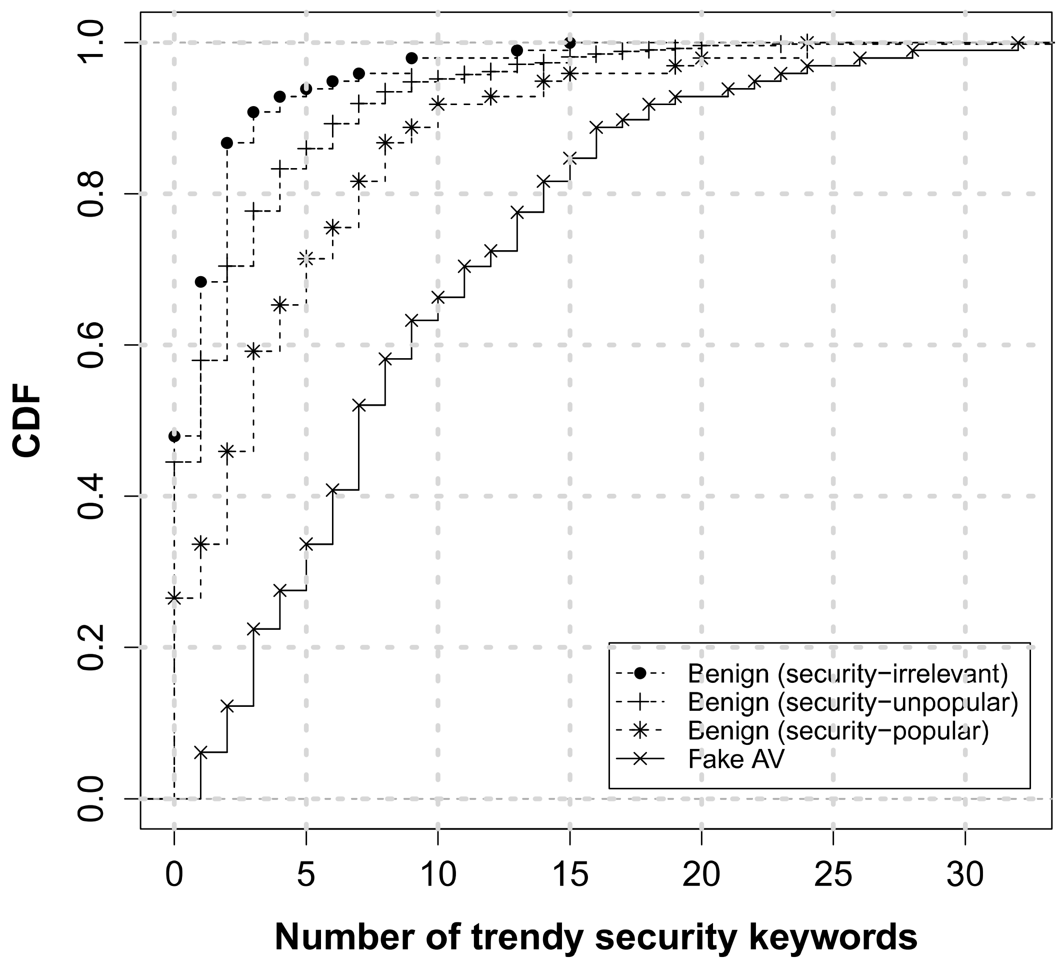


Fig. 7 – Content identity feature.

SEO Features

- ***Feature 4: path keywords***. *DART* divides the path of a URL into tokens using the delimiter "/" from left to right, where each token usually represents a directory in the web server. For example, "Spyware" resides in the 2*nd* level of the fake AV URL "www.xyz.com/Antivirus/Spyware/Dist/worm.html". Second, we accumulate the levels of directories that contain security keywords from $W_{security}$. For instance, suppose $\{antivirus, spyware, worm\} \subset W_{security}$, the value of this feature for the aforementioned URL will be 7 since 1*st*, 2*nd*, and 4*th* directories contain "Antivirus", "Spyware", and "worm", respectively.

Feature 4

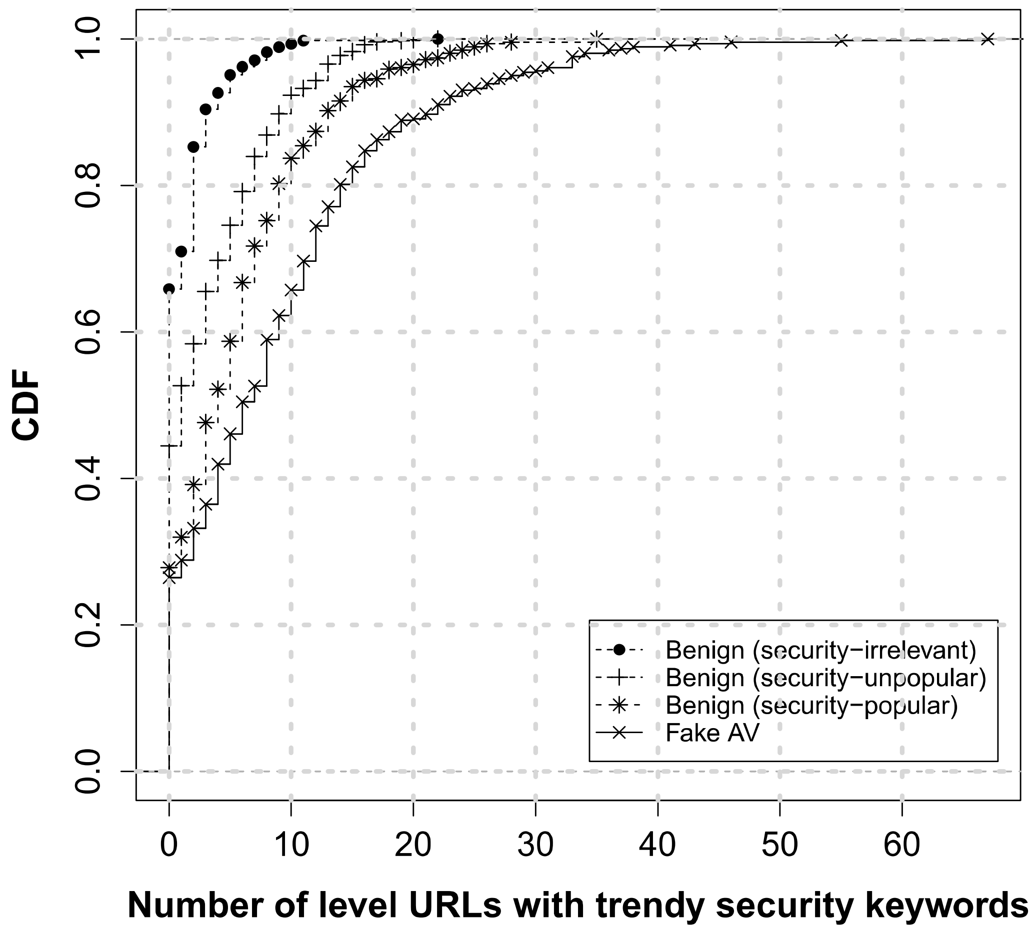


Fig. 8 – Path keyword feature.

SEO Features

- ***Feature 5: content keywords.*** Words in the webpage source codes are commonly employed for search engine for the webpage indexing. Attackers excessively tend to inject words of various security semantics into the webpage source codes, which can be easily analyzed by search engines. This feature represents the occurrence of words belonging to $W_{security}$ in the source codes of a landing page and all other webpages loaded by it.

Feature 5

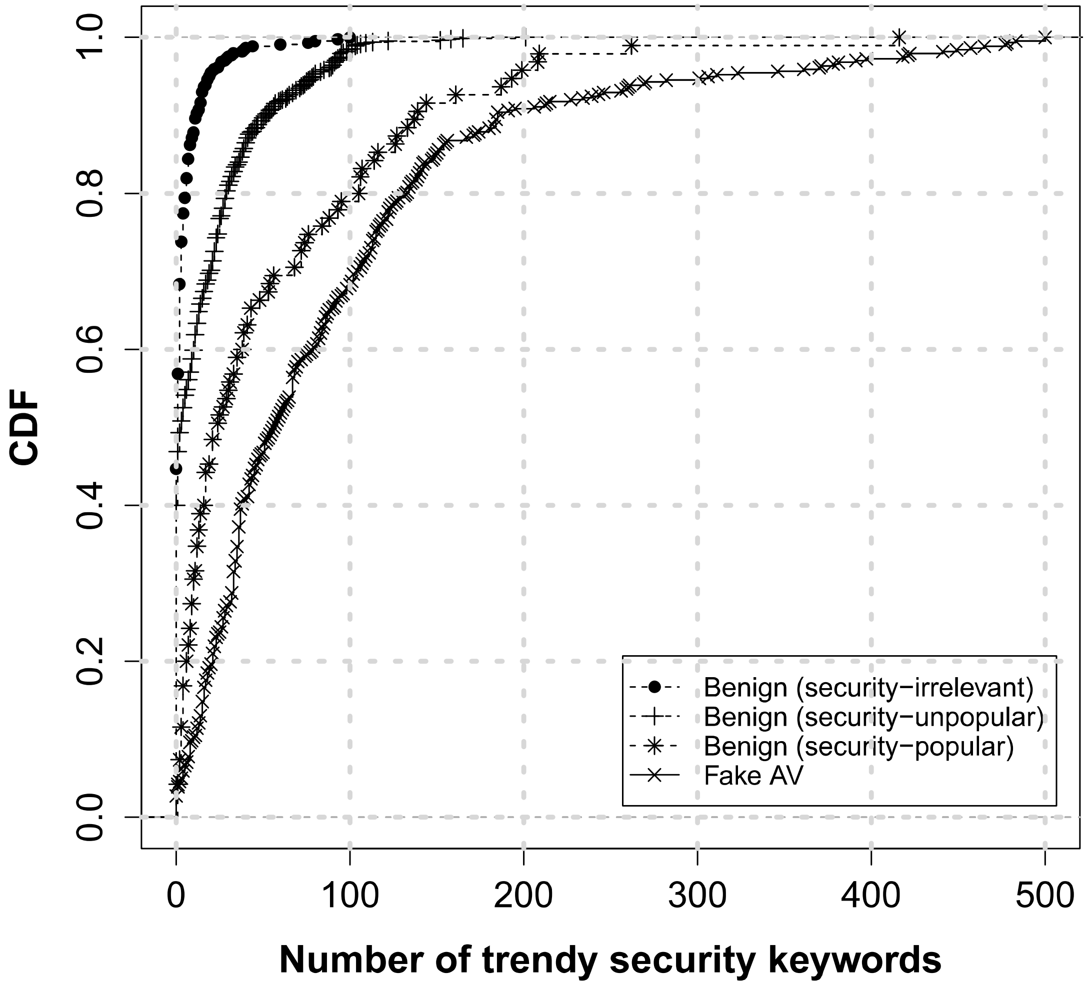
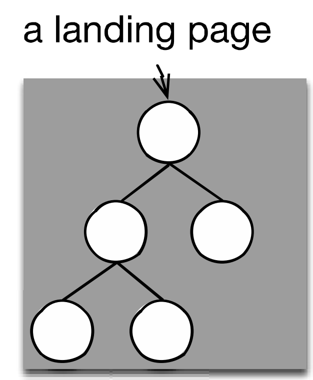


Fig. 9 – Content keyword feature.

Networking Features

- ***Feature 6: redirection***. For each webpage that is triggered by rendering the landing page, *DART* identifies the IP address(es) for the domain name in its URL and subsequently acquires the Autonomous System Number(s) (ASN) for the IP address(es) using public IP-to-ASN services Cymru (2013). This operation will result in a set of ASNs, which is denoted as ASN_{All}. If we denote the set of ASN(s) for the landing page as $ASN_{Landing}$ ($ASN_{Landing} \subseteq ASN_{All}$), then the value of this feature is defined as $|ASN_{All} - ASN_{Landing}|/|ASN_{All}|$.



Feature 6

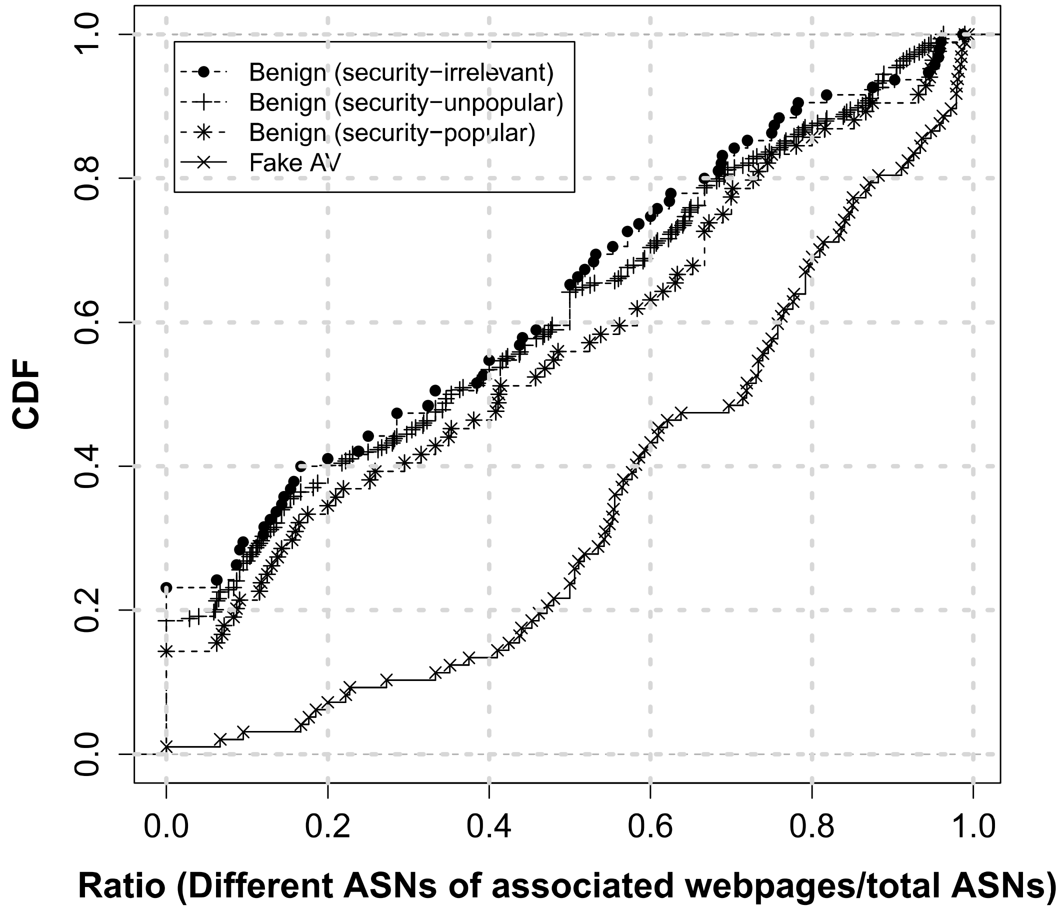


Fig. 10 – Redirection feature.

Networking Features

- ***Feature 7: maliciousness score***. An increasing number of domains are created and registered by attackers for malicious users such as fake AV webpages. The malicious score is to quantify the maliciousness of a collection of observed domains (denoted as $D_{observed}$) given their correlation with a set of known fake-AV domains (denoted as D_{seed}). We start

Feature 7

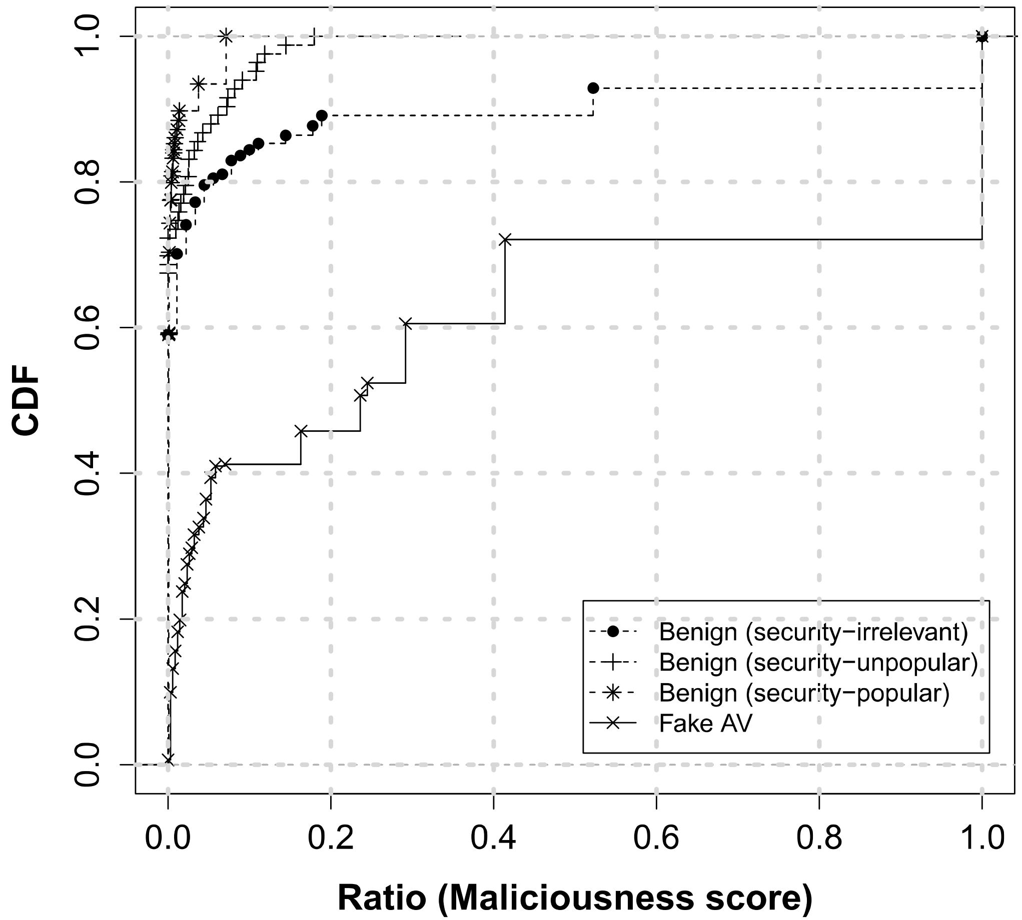


Fig. 11 – Maliciousness score feature.

Statistical Classifier and Detection

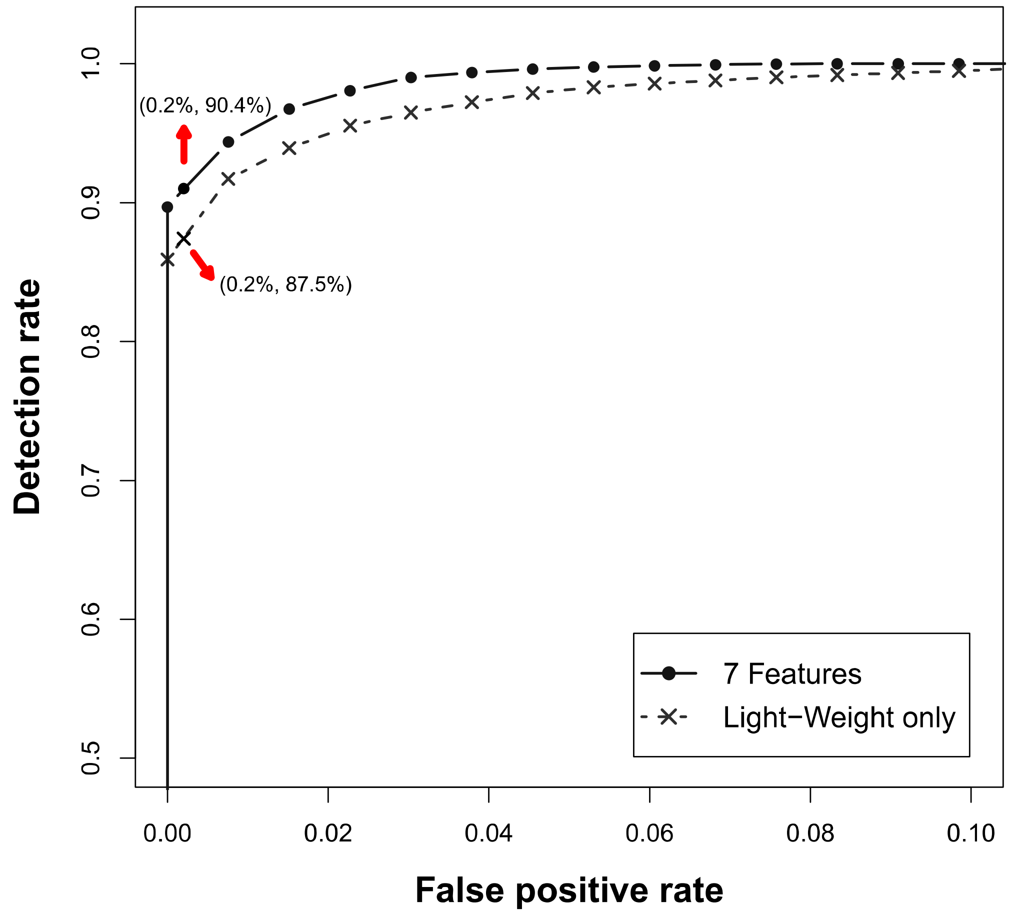


Fig. 12 — ROC comparison between light-weight and all features.

Feature Redundancy

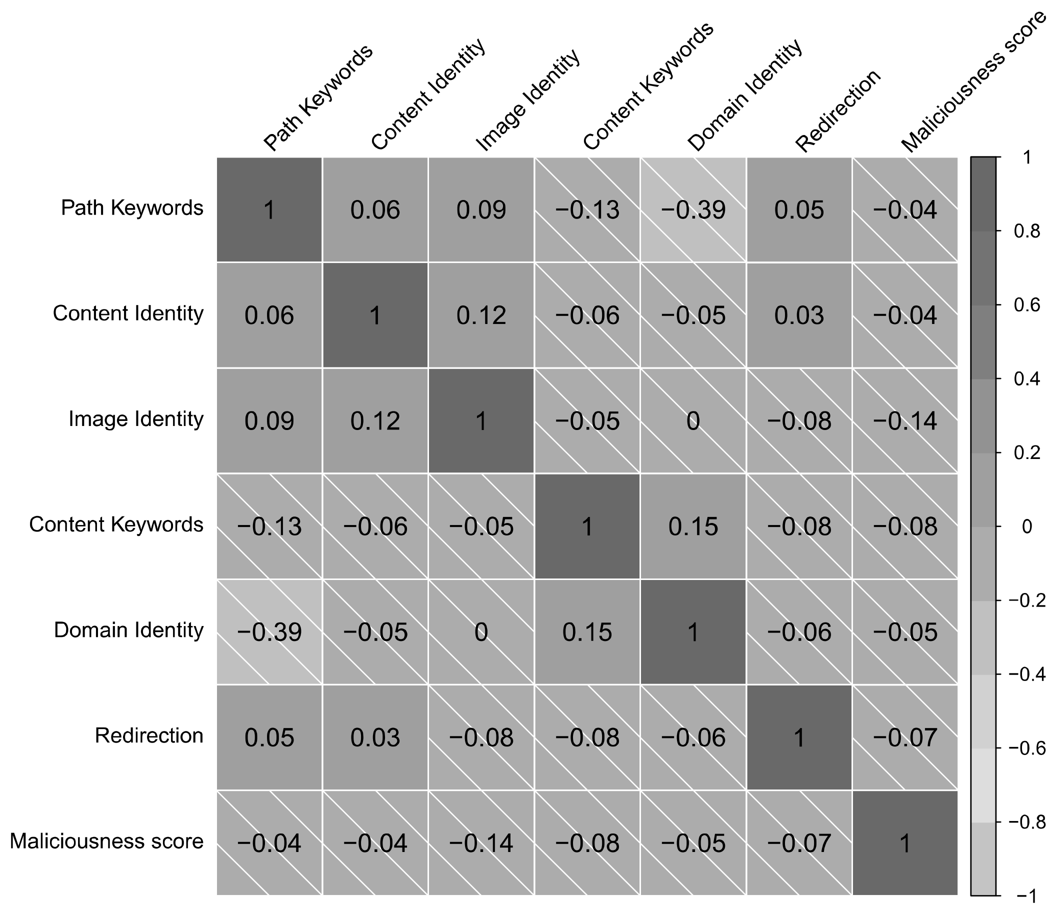


Fig. 13 — Correlation matrix of detection features.

END

CDF

Definition

The cumulative distribution function of a random variable X is the function
$$F(x) = P(X \leq x).$$

The cumulative distribution function gives the probability that the variable takes a value less than or equal to x and is defined for all real x .

If f is the probability mass function of a discrete random variable X with range $\{x_1, x_2, \ldots, \}$ and F is its cumulative distribution function, then
$$F(x) = \sum_{x_i \leq x} f(x_i).$$

CDF

If $p(x)$ is a density function for some characteristic of a population, then

$$\int_a^b p(x)\,dx = \begin{pmatrix} \text{fraction of the} \\ \text{population for} \\ \text{which } a \leq x \leq b \end{pmatrix}$$

Cumulative Distribution Function

Suppose $p(x)$ is a density function for a quantity.

The *cumulative distribution function* (cdf) for the quantity is defined as

$$P(x) = \int_{-\infty}^{x} p(t)\,dt$$

Gives:

- The proportion of population with value less than x
- The probability of having a value less than x.