Detecting Anomalous Behaviors in Computer Infrastructures

Dipartimento di Elettronica e Informazione
Politecnico di Milano
`http://www.vplab.elet.polimi.it`

Federico Maggi

My PhD has been sponsored by the Italian Governament and partially by the IST-216026-WOMBAT FP7.

Feb 25, 2010

The Internet is the largest computer infrastructure

And many people, institutions and firms “live” into it.

The Internet is the largest computer infrastructure

And many people, institutions and firms “live” into it.

Some numbers (2008-2009):

The Internet is the largest computer infrastructure

And many people, institutions and firms "live" into it.

Some numbers (2008-2009):

- **Google** estimates more than 1 trillion unique URLs,

The Internet is the largest computer infrastructure

And many people, institutions and firms “live” into it.

Some numbers (2008-2009):

- **Google** estimates more than 1 trillion unique URLs,
- **Facebook** has more than 250 millions active users (65 millions on mobile devices),

The Internet is the largest computer infrastructure

And many people, institutions and firms “live” into it.

Some numbers (2008-2009):

- **Google** estimates more than 1 trillion unique URLs,
- **Facebook** has more than 250 millions active users (65 millions on mobile devices),
- (Mar 2008) **YouTube** stores more than 70 million videos and the most popular video has been viewed 112,486,327 times.

...unfortunately...

Turns out it's also extremely insecure

Turns out it's also extremely insecure

Some numbers (2005-2009):

Turns out it's also extremely insecure

Some numbers (2005-2009):

- **1,250** breaches reported to authorities [9],

Turns out it's also extremely insecure

Some numbers (2005-2009):

- **1,250** breaches reported to authorities [9],
- **263,470,869** compromised records [1],

Turns out it's also extremely insecure

Some numbers (2005-2009):

- **1,250** breaches reported to authorities [9],
- **263,470,869** compromised records [1],
- (2008) **1 million** individual computers infected by Conficker [11],

Turns out it's also extremely insecure

Some numbers (2005-2009):

- **1,250** breaches reported to authorities [9],
- **263,470,869** compromised records [1],
- (2008) **1 million** individual computers infected by Conficker [11],
- (2008) **75,000 bots/day** [11].

Turns out it's also extremely insecure

Some numbers (2005-2009):

- **1,250** breaches reported to authorities [9],
- **263,470,869** compromised records [1],
- (2008) **1 million** individual computers infected by Conficker [11],
- (2008) **75,000 bots/day** [11].

Turns out it's also extremely insecure

Some numbers (2005-2009):

- **1,250** breaches reported to authorities [9],
- **263,470,869** compromised records [1],
- (2008) **1 million** individual computers infected by Conficker [11],
- (2008) **75,000 bots/day** [11].

Note: these only refer to the facts that have been **detected** and **reported**.

Main causes of todays' Internet security issues

Main causes of todays' Internet security issues

- vulnerable software,

Main causes of todays' Internet security issues

- vulnerable software,
- **efficient exploitation**,

Main causes of todays' Internet security issues

- vulnerable software,
- **efficient exploitation**,
- **well-organized and powerful cyber-crime**.

Main causes of todays' Internet security issues

- vulnerable software,
- **efficient exploitation**,
- **well-organized and powerful cyber-crime**.

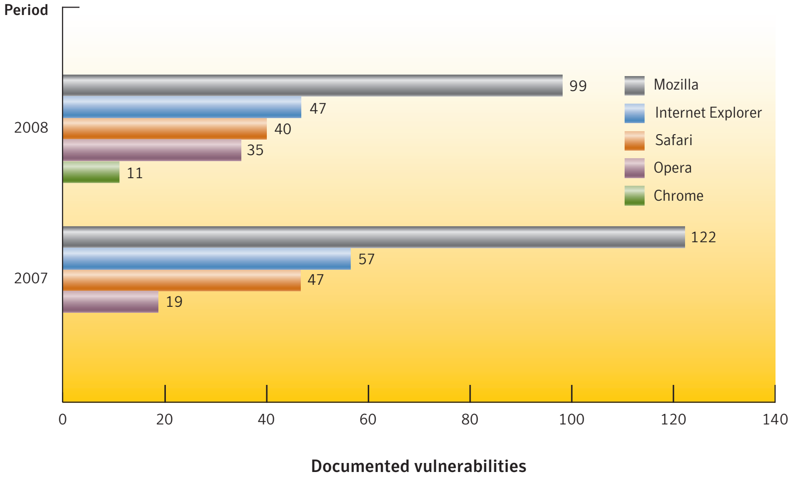
Main causes of todays' Internet security issues

- vulnerable software,
- **efficient exploitation**,
- **well-organized and powerful cyber-crime**.

This is actually a “lethal cocktail”: let's see why.

The most popular software tool is flawed

BTW, looks like MS IE is more secure than Mozilla :)



Actually, browsers do not really matter

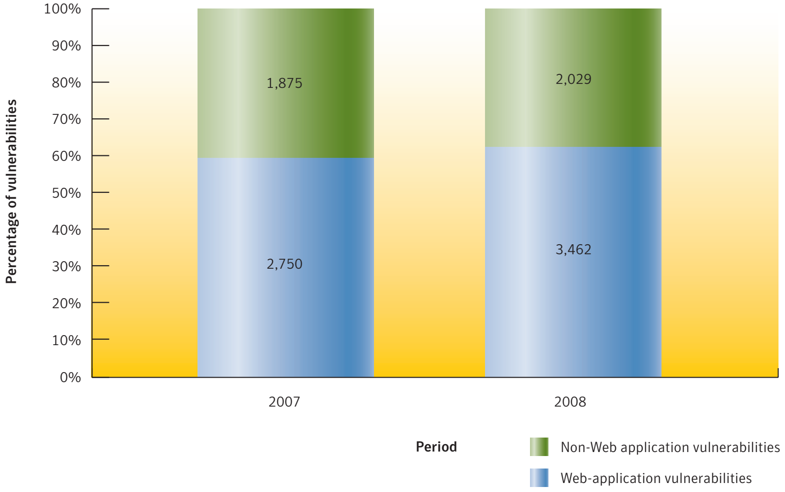
Here is the real culprit

Actually, browsers do not really matter

Here is the real culprit

Plug-in	2008 Top Category	2007 Top Category
Adobe Acrobat Reader	Memory corruption	Memory corruption/content injection/ command execution
Adobe Flash Player	Memory corruption/origin validation/ elevated security context	Elevated security context
ActiveX	Memory corruption	Memory corruption
Java	Elevated security context	Elevated security context
Mozilla Extensions	Content injection	Content injection
QuickTime	Memory corruption	Memory corruption
Windows Media Player	Memory corruption	Memory corruption/DoS

The most accessible applications are flawed too

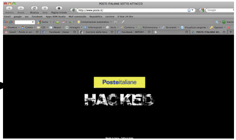


Overview of a modern exploitation work-flow

Overview of a modern exploitation work-flow

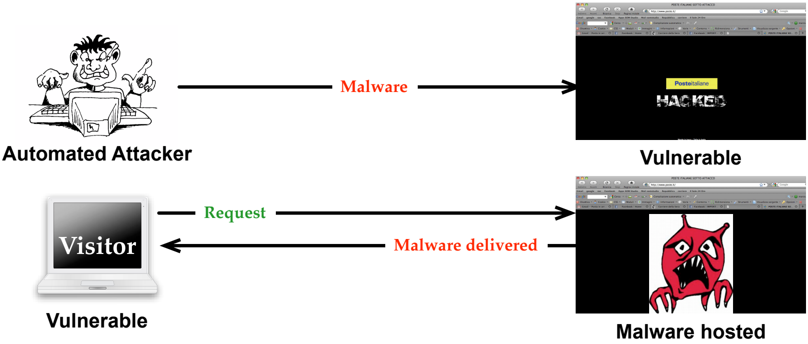
Automated Attacker

Malware

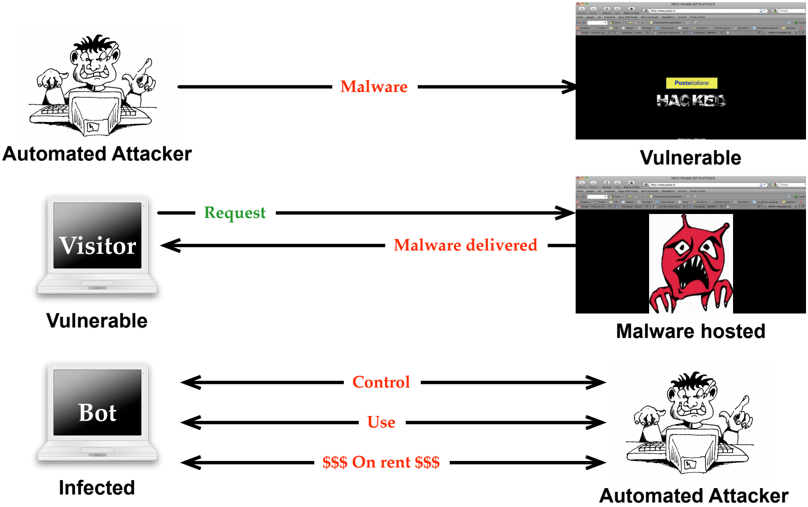


Vulnerable

Overview of a modern exploitation work-flow



Overview of a modern exploitation work-flow



Wait a minute...

..do they really rent compromized resources?

Wait a minute...

..do they really rent compromized resources?

Why not? Amazon S3/EC2 rents computers as a service.
Cyber-criminals do that too.

Wait a minute...

..do they really rent compromized resources?

Why not? Amazon S3/EC2 rents computers as a service.
Cyber-criminals do that too.

- Not only they trade stolen information (old news):

Wait a minute...

..do they really rent compromized resources?

Why not? Amazon S3/EC2 rents computers as a service. Cyber-criminals do that too.

- Not only they trade stolen information (old news):
  - Credit cards go as low as $0.30-60.0

Wait a minute...

..do they really rent compromized resources?

Why not? Amazon S3/EC2 rents computers as a service. Cyber-criminals do that too.

- Not only they trade stolen information (old news):
  - Credit cards go as low as $0.30-60.0
  - Bank accounts for less than $1,000

Wait a minute...

..do they really rent compromized resources?

Why not? Amazon S3/EC2 rents computers as a service.
Cyber-criminals do that too.

- Not only they trade stolen information (old news):
  - Credit cards go as low as $0.30-60.0
  - Bank accounts for less than $1,000
- Attack services run using botnets:

Wait a minute...

..do they really rent compromized resources?

Why not? Amazon S3/EC2 rents computers as a service.
Cyber-criminals do that too.

- Not only they trade stolen information (old news):
  - Credit cards go as low as $0.30-60.0
  - Bank accounts for less than $1,000
- Attack services run using botnets:
  - DDoS

Wait a minute...

..do they really rent compromized resources?

Why not? Amazon S3/EC2 rents computers as a service.
Cyber-criminals do that too.

- Not only they trade stolen information (old news):
  - Credit cards go as low as $0.30-60.0
  - Bank accounts for less than $1,000
- Attack services run using botnets:
  - DDoS
  - Phishing campaigns

Wait a minute...

..do they really rent compromized resources?

Why not? Amazon S3/EC2 rents computers as a service.
Cyber-criminals do that too.

- Not only they trade stolen information (old news):
  - Credit cards go as low as $0.30-60.0
  - Bank accounts for less than $1,000
- Attack services run using botnets:
  - DDoS
  - Phishing campaigns
  - Spamming campaigns,

Wait a minute...

..do they really rent compromized resources?

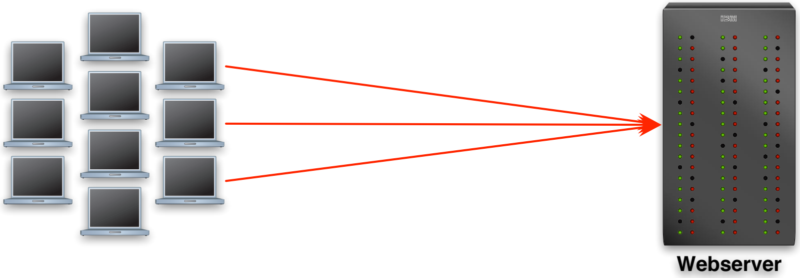
Why not? Amazon S3/EC2 rents computers as a service.
Cyber-criminals do that too.

- Not only they trade stolen information (old news):
  - Credit cards go as low as $0.30-60.0
  - Bank accounts for less than $1,000
- Attack services run using botnets:
  - DDoS
  - Phishing campaigns
  - Spamming campaigns,
  - Scam web-sites design!

Pick your choice from the attack-as-a-service gourmet menu

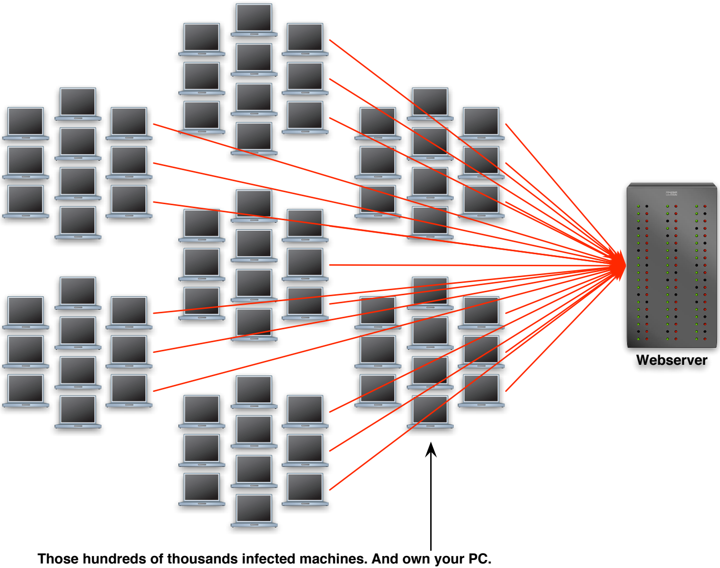
2008 Rank	2007 Rank	Item	2008 Percentage	2007 Percentage	Range of Prices
1	1	Credit card information	32%	21%	\$0.06–\$30
2	2	Bank account credentials	19%	17%	\$10–\$1000
3	9	Email accounts	5%	4%	\$0.10–\$100
4	3	Email addresses	5%	6%	\$0.33/MB–\$100/MB
5	12	Proxies	4%	3%	\$0.16–\$20
6	4	Full identities	4%	6%	\$0.70–\$60
7	6	Mailers	3%	5%	\$2–\$40
8	5	Cash out services	3%	5%	8%–50% or flat rate of \$200–\$2000 per item
9	17	Shell scripts	3%	2%	\$2–\$20
10	8	Scams	3%	5%	\$3–\$40/week for hosting, \$2–\$20 design

A few years ago...

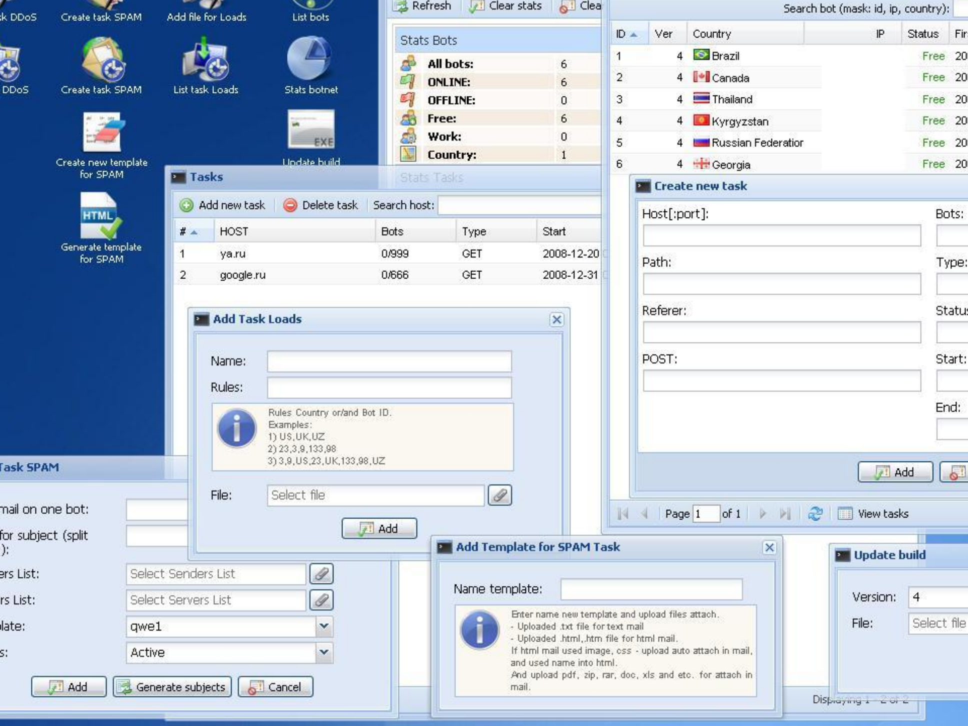


…and nowadays

It's just a multiplication factor but it is damn significant!



...and they come with a sweet graphical user interface...



k DDoS
Create task SPAM
Add file for Loads
List bots
DDoS
Create task SPAM
List task Loads
Stats botnet
Create new template for SPAM
Update build
Generate template for SPAM
Refresh
Clear stats
Stats Bots
All bots: 6
ONLINE: 6
OFFLINE: 0
Free: 6
Work: 0
Country: 1
Stats Tasks
Search bot (mask: id, ip, country):
ID | Ver | Country | IP | Status
1 | 4 | Brazil | | Free
2 | 4 | Canada | | Free
3 | 4 | Thailand | | Free
4 | 4 | Kyrgyzstan | | Free
5 | 4 | Russian Federation | | Free
6 | 4 | Georgia | | Free
Tasks
Add new task
Delete task
Search host:
| HOST | Bots | Type | Start
1 | ya.ru | 0/999 | GET | 2008-12-20
2 | google.ru | 0/666 | GET | 2008-12-31
Create new task
Host[:port]:
Bots:
Path:
Type:
Referer:
POST:
Start:
End:
Add
Page 1 of 1
View tasks
Add Task Loads
Name:
Rules:
Rules Country or/and Bot ID.
Examples:
1) US,UK,UZ
2) 23,3,9,133,98
3) 3,9,US,23,UK,133,98,UZ
File:
Select file
Add
Task SPAM
mail on one bot:
for subject (split
):
ers List:
Select Senders List
rs List:
Select Servers List
late:
qwe1
s:
Active
Add
Generate subjects
Cancel
Add Template for SPAM Task
Name template:
Enter name new template and upload files attach.
- Uploaded .txt file for text mail
- Uploaded .html,.htm file for html mail.
If html mail used image, css - upload auto attach in mail, and used name into html.
And upload pdf, zip, rar, doc, xls and etc. for attach in mail.
Update build
Version:
4
File:
Select file

One possible Mitigation Strategy

Attacks generate anomalous behavior

One possible Mitigation Strategy

Attacks generate anomalous behavior

- Any computer system generates **observable activity**,

One possible Mitigation Strategy

Attacks generate anomalous behavior

- Any computer system generates **observable activity**,
  - **Example:** processes, system calls, HTTP requests;

One possible Mitigation Strategy

Attacks generate anomalous behavior

- Any computer system generates **observable activity**,
  - **Example:** processes, system calls, HTTP requests;
- **Hyp.:** compromised systems generate **unusual activity**,

One possible Mitigation Strategy

Attacks generate anomalous behavior

- Any computer system generates **observable activity**,
  - **Example:** processes, system calls, HTTP requests;
- **Hyp.:** compromised systems generate **unusual activity**,
  - **Example:** too-long message content, too-many processes,

One possible Mitigation Strategy

Attacks generate anomalous behavior

- Any computer system generates **observable activity**,
  - **Example:** processes, system calls, HTTP requests;
- **Hyp.:** compromised systems generate **unusual activity**,
  - **Example:** too-long message content, too-many processes,
- learning models of benign activity to detect malicious behaviors.

One possible Mitigation Strategy

Attacks generate anomalous behavior

- Any computer system generates **observable activity**,
  - **Example:** processes, system calls, HTTP requests;
- **Hyp.:** compromised systems generate **unusual activity**,
  - **Example:** too-long message content, too-many processes,
- learning models of benign activity to detect malicious behaviors.
  - **Example:** $\rightarrow$

Simple example

HTTP messages (requests)

Simple example

HTTP messages (requests)

`/article/id/32`

Simple example

HTTP messages (requests)

```
/article/id/32
/comment/<par1>/<par1-val>
```

Simple example

HTTP messages (requests)

```
/article/id/32
/comment/<par1>/<par1-val>
/login/<par1>/<par1-val>/<par2>/<par2-val>
```

Simple example

HTTP messages (requests)

```
/article/id/32
/comment/<par1>/<par1-val>
/login/<par1>/<par1-val>/<par2>/<par2-val>
...
```

Simple example

HTTP messages (requests)

```
/article/id/32
/comment/<par1>/<par1-val>
/login/<par1>/<par1-val>/<par2>/<par2-val>
...
/<component1>/<par1>/<par1-val>/<par2>/<par2-val>
```

Simple example

HTTP messages (requests)

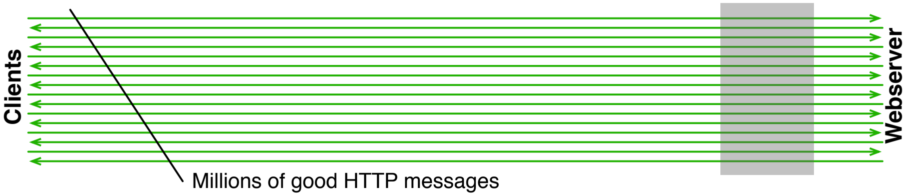
```
/article/id/32
/comment/<par1>/<par1-val>
/login/<par1>/<par1-val>/<par2>/<par2-val>
...
/<component1>/<par1>/<par1-val>/<par2>/<par2-val>
/<component2>/<par1>/<par1-val>
```

Simple example

Anomaly detection

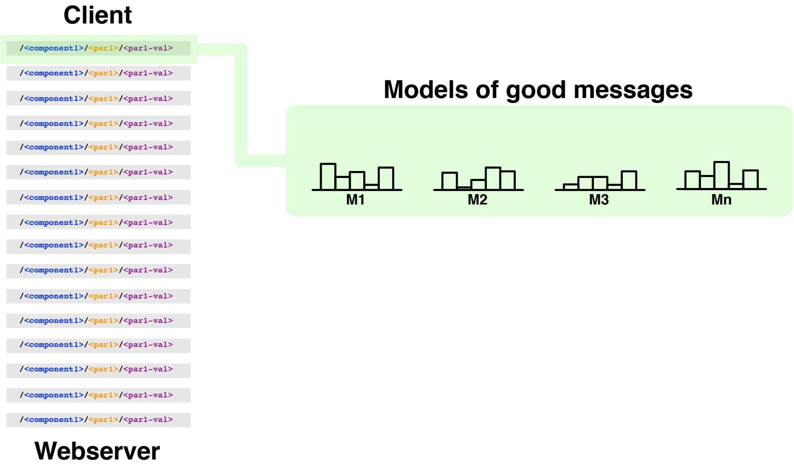
Simple example

Anomaly detection



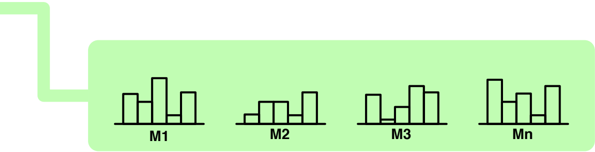
Simple example

Anomaly detection



Simple example

Anomaly detection

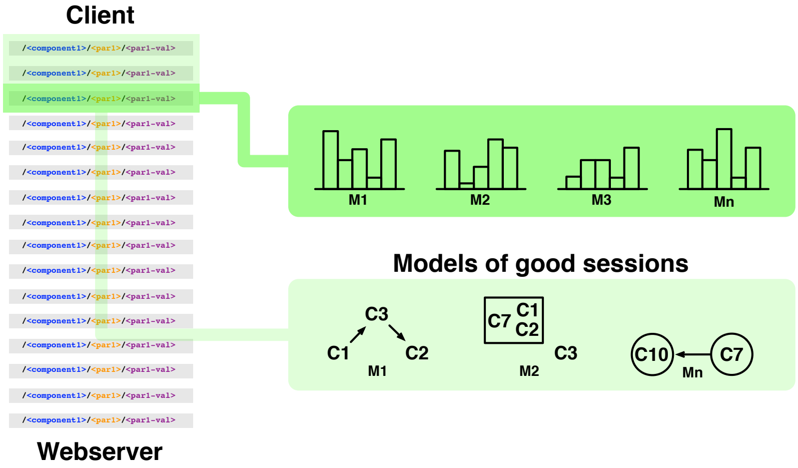


Example of models

— parameter string length
— numeric range
— probabilistic grammar of strings
— string character distribution

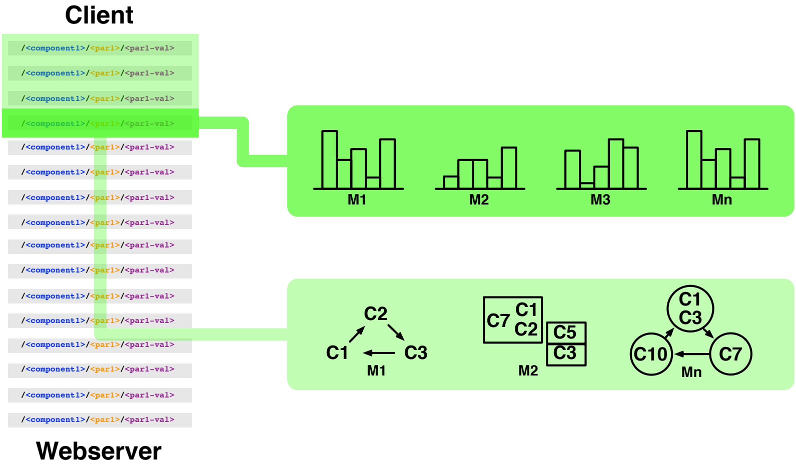
Simple example

Anomaly detection



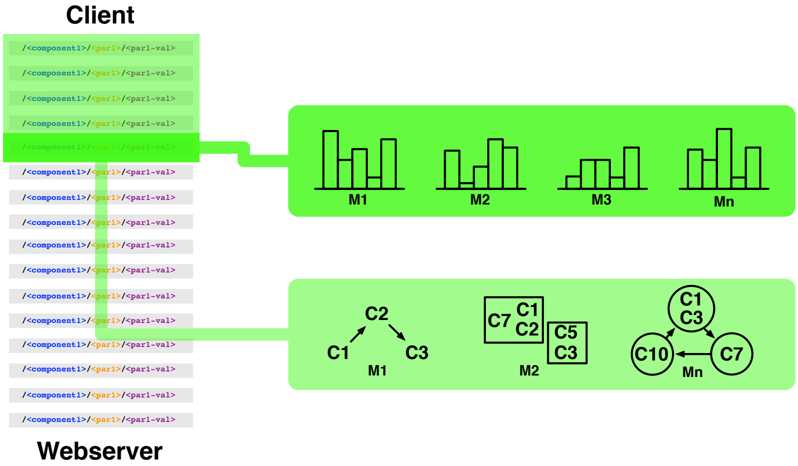
Simple example

Anomaly detection



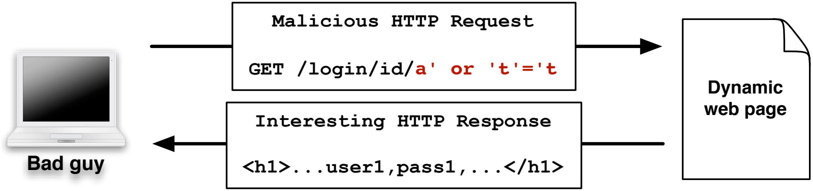
Simple example

Anomaly detection



Simple example

Anomaly detection

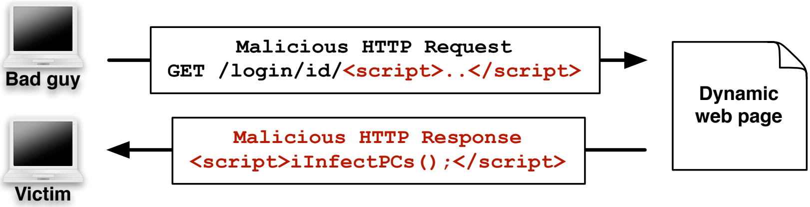


Simple example

Anomaly detection

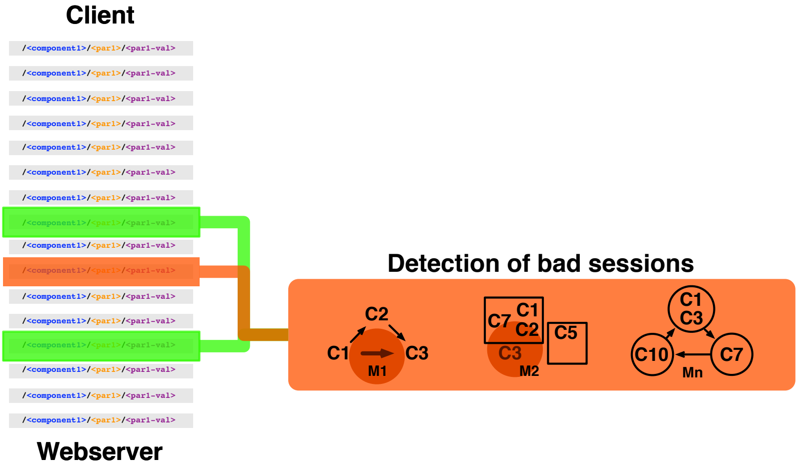
Simple example

Anomaly detection



Simple example

Anomaly detection



The same applies to any type of activity.
The **crucial** point is how to design **models**.

Our research

Three subjects

Our research

Three subjects

1. HTTP interactions,

Our research

Three subjects

1. HTTP interactions,
   - primary communication channel between malicious machines;

Our research

Three subjects

1. HTTP interactions,
   - primary communication channel between malicious machines;
2. operating system processes,

Our research

Three subjects

1. HTTP interactions,
   - primary communication channel between malicious machines;
2. operating system processes,
   - malicious code (i.e., virus) is the typical infection vector;

Our research

Three subjects

1. HTTP interactions,
    - primary communication channel between malicious machines;
2. operating system processes,
    - malicious code (i.e., virus) is the typical infection vector;
3. combination of the two,

Our research

Three subjects

1. HTTP interactions,
   - primary communication channel between malicious machines;
2. operating system processes,
   - malicious code (i.e., virus) is the typical infection vector;
3. combination of the two,
   - malicious network activity $\rightarrow$ malicious activity on the operating system.

1. HTTP interactions

Protecting web applications and clients

Protecting web applications and clients

Models of:

Protecting web applications and clients

Models of:

- HTTP requests,

to protect

Protecting web applications and clients

Models of:

- HTTP requests,
- HTTP responses,

to protect

Protecting web applications and clients

Models of:

- HTTP requests,
- HTTP responses,
- SQL queries,

to protect

Protecting web applications and clients

Models of:

- HTTP requests,
- HTTP responses,
- SQL queries,

to protect

Protecting web applications and clients

Models of:

- HTTP requests,
- HTTP responses,
- SQL queries,

to protect

- the server from malicious requests,

Protecting web applications and clients

Models of:

- HTTP requests,
- HTTP responses,
- SQL queries,

to protect

- the server from malicious requests,
- the client from infected sites,

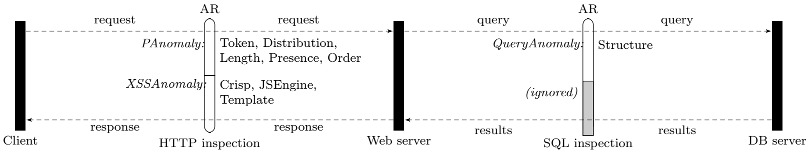
Protecting web applications and clients

Models of:

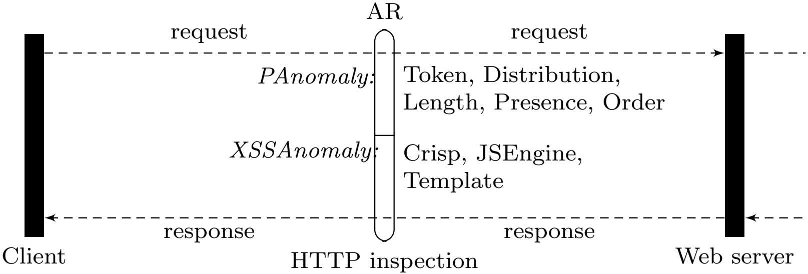
- HTTP requests,
- HTTP responses,
- SQL queries,

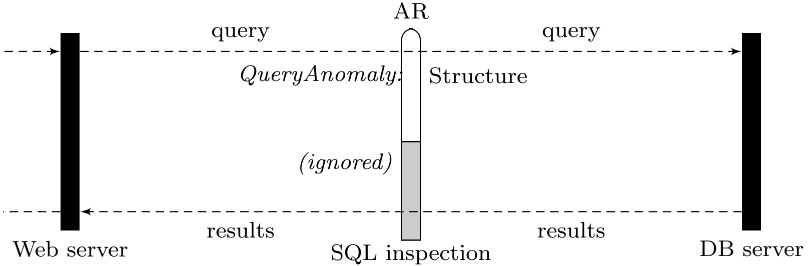
to protect

- the server from malicious requests,
- the client from infected sites,
- the database from malicious queries.



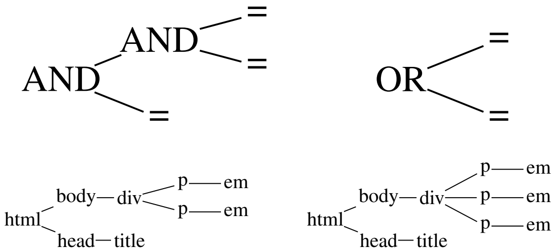
AR
request
request
PAnomaly:
Token, Distribution,
Length, Presence, Order
XSSAnomaly:
Crisp, JSEngine,
Template
response
response
Client
HTTP inspection
Web server
AR
query
query
QueryAnomaly:
Structure
(ignored)
results
results
SQL inspection
DB server


AR
request
request
PAnomaly:
Token, Distribution, Length, Presence, Order
XSSAnomaly:
Crisp, JSEngine, Template
response
response
Client
HTTP inspection
Web server

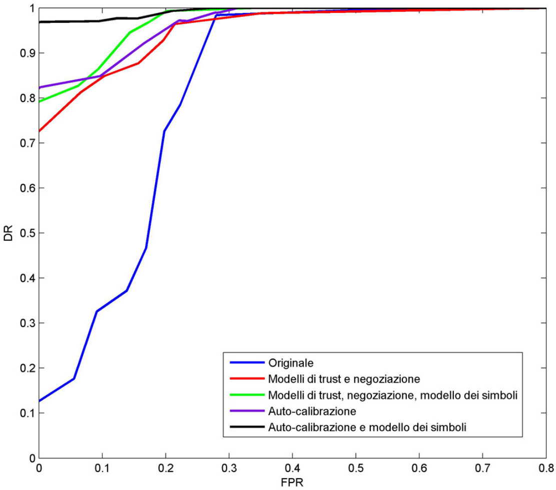

AR
query
query
QueryAnomaly:
Structure
(ignored)
results
results
Web server
SQL inspection
DB server

Example of very simple models

Example of very simple models



Overall detection capabilities



Tested on about HTTP 8,000 requests, 3000 attacks. EC2ND 2009 [2].

Updating obsolete models dynamically

What if the website changes?

Updating obsolete models dynamically

What if the website changes?

Models become **obsolete**, but HTTP **responses** contain good insights:

Updating obsolete models dynamically

What if the website changes?

Models become **obsolete**, but HTTP **responses** contain good insights:

- new links $\rightarrow$ potential requests $\rightarrow$ new models,

Updating obsolete models dynamically

What if the website changes?

Models become **obsolete**, but HTTP **responses** contain good insights:

- new links $\rightarrow$ potential requests $\rightarrow$ new models,
  - `<a href="/new/resource/path" />`

Updating obsolete models dynamically

What if the website changes?

Models become **obsolete**, but HTTP **responses** contain good insights:

- new links → potential requests → new models,
  - `<a href="/new/resource/path" />`
- new parameters → new models,

Updating obsolete models dynamically

What if the website changes?

Models become **obsolete**, but HTTP **responses** contain good insights:

- new links → potential requests → new models,
  - `<a href="/new/resource/path" />`
- new parameters → new models,
  - `<a href="/path?new_parameter" />`

Updating obsolete models dynamically

What if the website changes?

Models become **obsolete**, but HTTP **responses** contain good insights:

- new links $\rightarrow$ potential requests $\rightarrow$ new models,
  - `<a href="/new/resource/path" />`
- new parameters $\rightarrow$ new models,
  - `<a href="/path?new_parameter" />`
- new parameter values $\rightarrow$ new training values.

Updating obsolete models dynamically

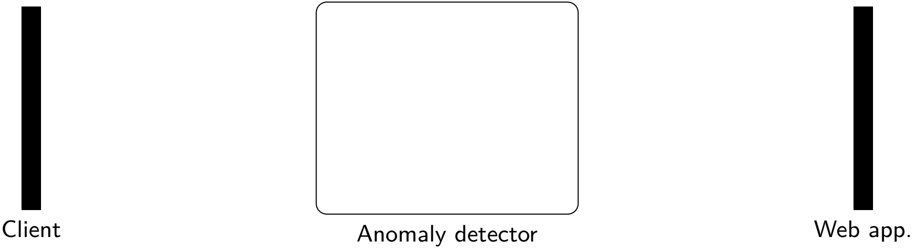
What if the website changes?

Models become **obsolete**, but HTTP **responses** contain good insights:

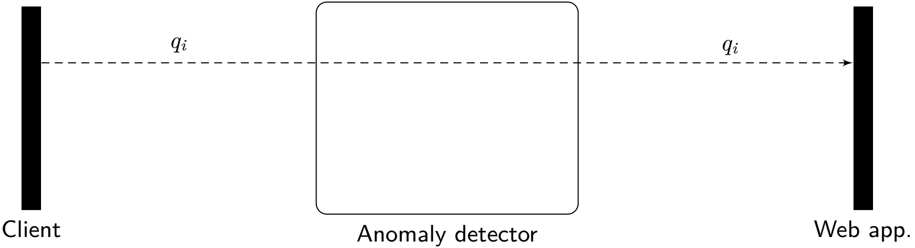
- new links → potential requests → new models,
  - `<a href="/new/resource/path" />`
- new parameters → new models,
  - `<a href="/path?new_parameter" />`
- new parameter values → new training values.
  - `<a href="/path?parameter=new_value" />`

Parsing HTTP responses to update models

Parsing HTTP responses to update models

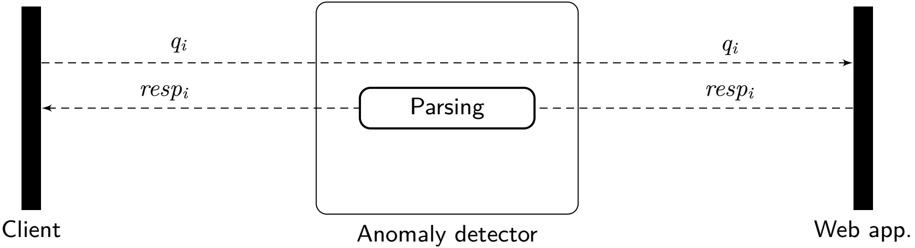


Parsing HTTP responses to update models



for each request q_i

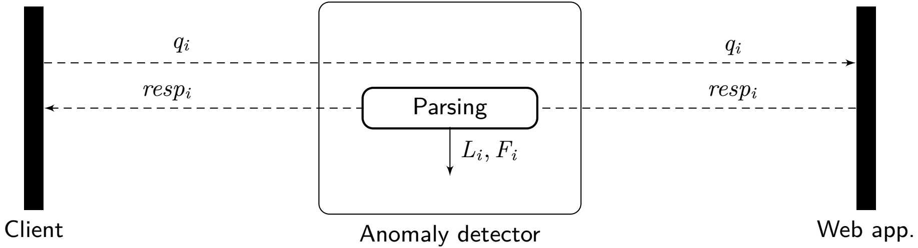
Parsing HTTP responses to update models



for each request q_i

intercept the corresponding response $resp_i$

Parsing HTTP responses to update models

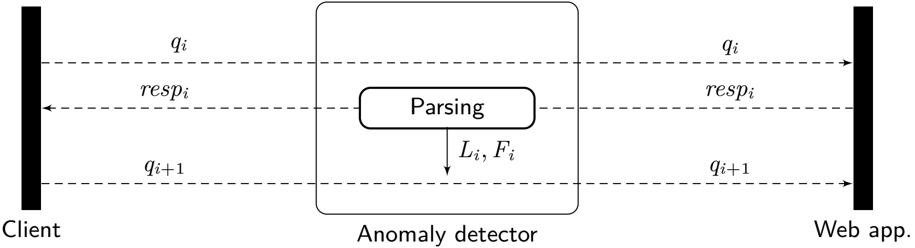


for each request q_i

intercept the corresponding response $resp_i$

extract parmeters and values from links, forms, fields

Parsing HTTP responses to update models

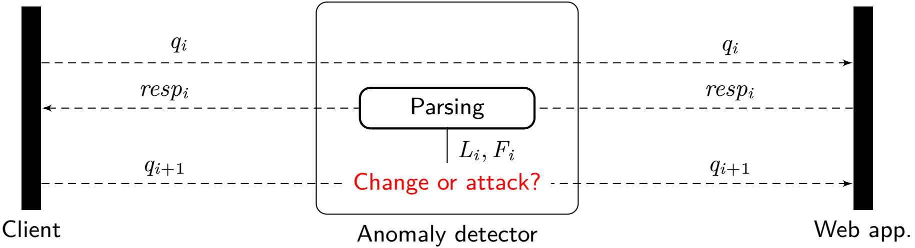


for each request q_i

- intercept the corresponding response $resp_i$
- extract parmeters and values from links, forms, fields

at next request q_{i+1}

Parsing HTTP responses to update models

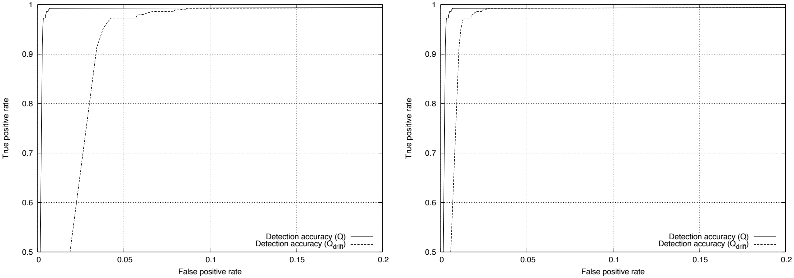


for each request q_i

- intercept the corresponding response $resp_i$
- extract parmeters and values from links, forms, fields

at next request q_{i+1}

- compare parameter and values to spot legit changes



(a) Response modeling disabled. (b) Response modeling enabled.

Tested on 823 web applications, 58,732,624 HTTP requests, 1000 attacks. RAID 2009 [6] (w/ UC Santa Barbara).

Training with almost no data

Some pages are infrequently accessed

Training with almost no data

Some pages are infrequently accessed

Scarce HTTP interactions $\rightarrow$ scarce training data, but:

Training with almost no data

Some pages are infrequently accessed

Scarce HTTP interactions $\rightarrow$ scarce training data, but:

- similar models have (i.e., capture) similar characteristics,

Training with almost no data

Some pages are infrequently accessed

Scarce HTTP interactions $\rightarrow$ scarce training data, but:

- similar models have (i.e., capture) similar characteristics,
- group similar models,

Training with almost no data

Some pages are infrequently accessed

Scarce HTTP interactions $\rightarrow$ scarce training data, but:

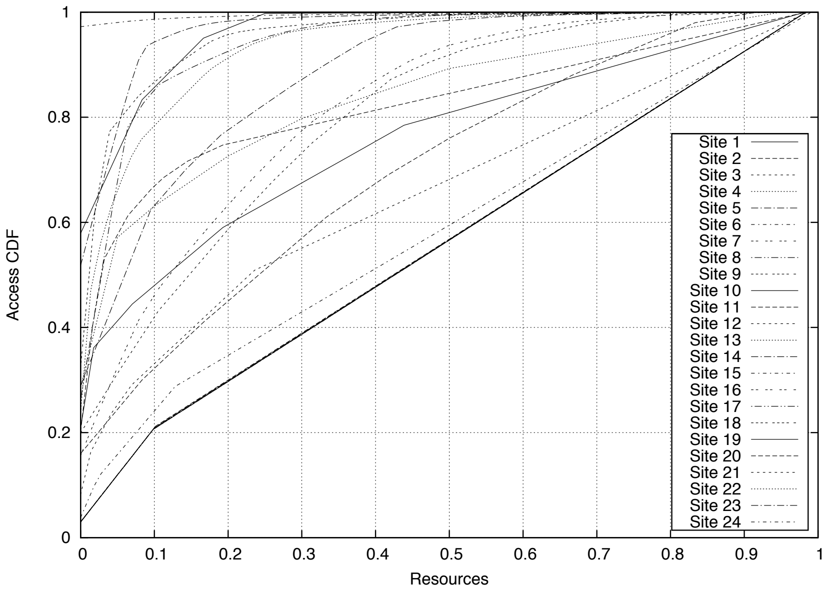
- similar models have (i.e., capture) similar characteristics,
- group similar models,
- rank models according to their completeness,

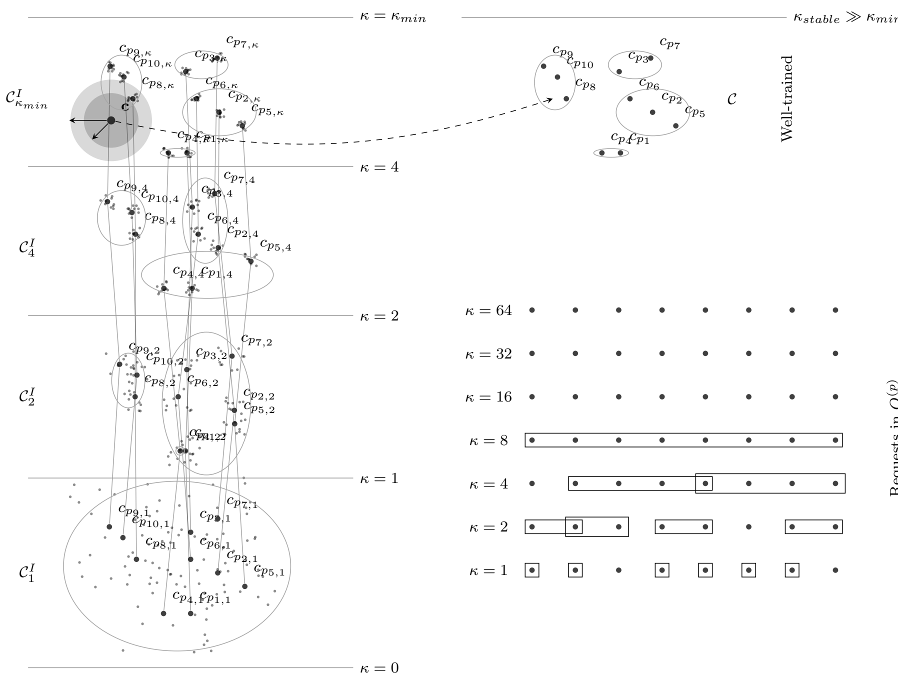
Training with almost no data

Some pages are infrequently accessed

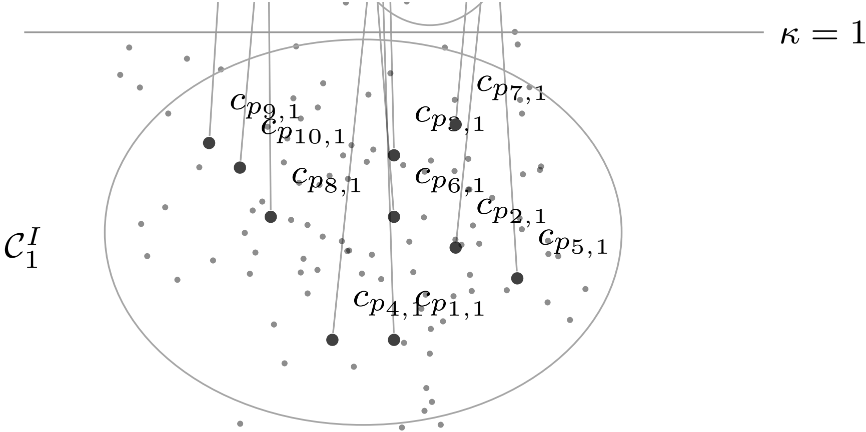
Scarce HTTP interactions $\rightarrow$ scarce training data, but:

- similar models have (i.e., capture) similar characteristics,
- group similar models,
- rank models according to their completeness,
- substitute a poorly-trained model with a similar one, but well-trained.

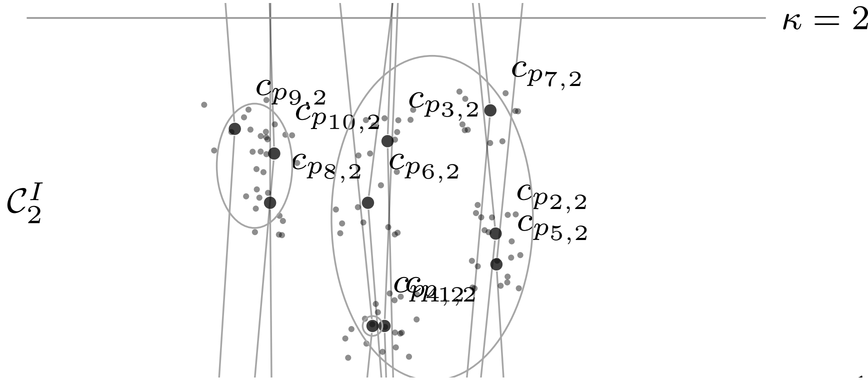

Access CDF
Resources
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
Site 1
Site 2
Site 3
Site 4
Site 5
Site 6
Site 7
Site 8
Site 9
Site 10
Site 11
Site 12
Site 13
Site 14
Site 15
Site 16
Site 17
Site 18
Site 19
Site 20
Site 21
Site 22
Site 23
Site 24

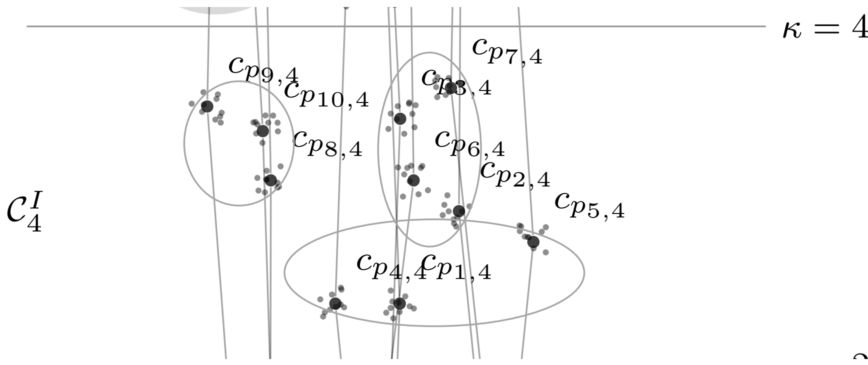


(a) (b)

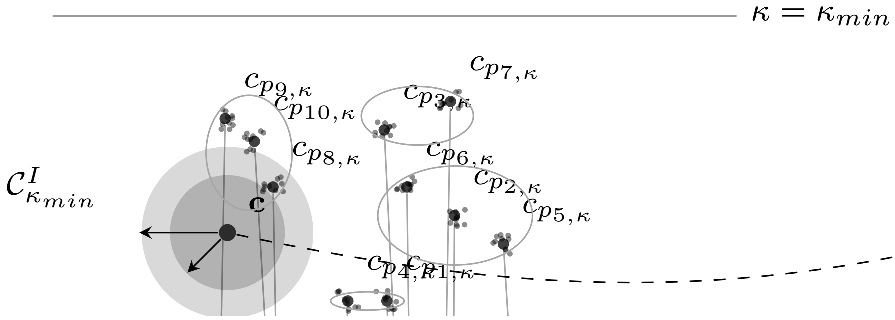


$\kappa = 1$
$c_{p_{9,1}}$
$c_{p_{10,1}}$
$c_{p_{7,1}}$
$c_{p_{3,1}}$
$c_{p_{8,1}}$
$c_{p_{6,1}}$
$c_{p_{2,1}}$
$c_{p_{5,1}}$
$\mathcal{C}_1^I$
$c_{p_{4,1}}$
$c_{p_{1,1}}$

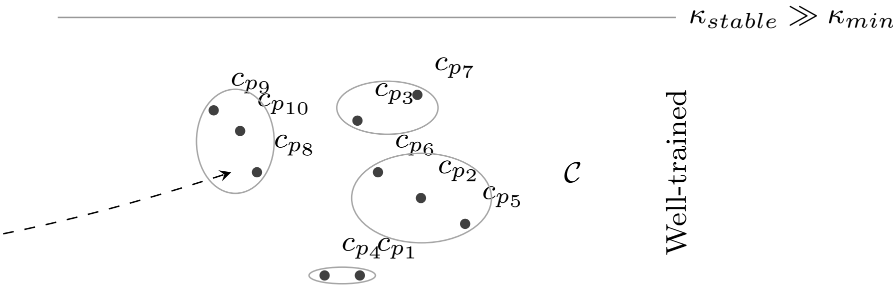

$\kappa = 2$
$\mathcal{C}_2^I$
$c_{p_{7,2}}$
$c_{p_{9,2}}$
$c_{p_{3,2}}$
$c_{p_{10,2}}$
$c_{p_{8,2}}$
$c_{p_{6,2}}$
$c_{p_{2,2}}$
$c_{p_{5,2}}$

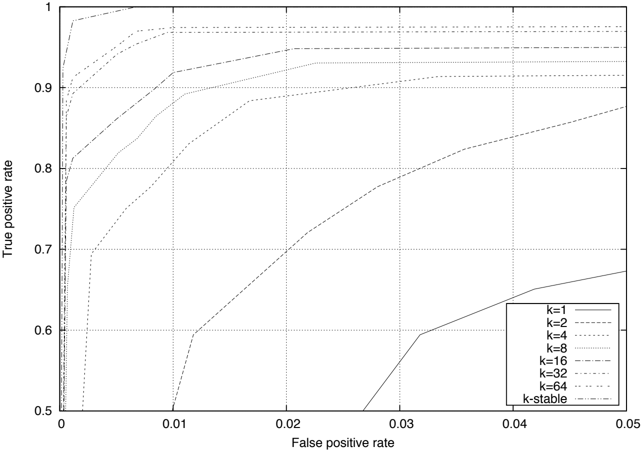


$\kappa = 4$
$c_{p_7,4}$
$c_{p_9,4}$
$c_{p_3,4}$
$c_{p_{10},4}$
$c_{p_8,4}$
$c_{p_6,4}$
$c_{p_2,4}$
$c_{p_5,4}$
$\mathcal{C}_4^I$
$c_{p_4,4}$
$c_{p_1,4}$



$\kappa = \kappa_{min}$
$c_{p_{7,\kappa}}$
$c_{p_{9,\kappa}}$
$c_{p_{10,\kappa}}$
$c_{p_{3,\kappa}}$
$c_{p_{8,\kappa}}$
$c_{p_{6,\kappa}}$
$\mathcal{C}^I_{\kappa_{min}}$
$c_{p_{2,\kappa}}$
$\mathbf{c}$
$c_{p_{5,\kappa}}$
$c_{p_{4,\kappa}}$
$c_{p_{1,\kappa}}$


$\kappa_{stable} \gg \kappa_{min}$
c_{p_7}
c_{p_9}
c_{p_3}
$c_{p_{10}}$
c_{p_8}
c_{p_6}
c_{p_2}
$\mathcal{C}$
c_{p_5}
c_{p_4}
c_{p_1}
Well-trained



Tested on 823 web applications, 58,732,624 HTTP requests, 1000 attacks. NDSS 2010 [10] (w/ UC Santa Barbara).

2. Operating system processes

Protecting the operating system

How to model a process' activity?

Protecting the operating system

How to model a process' activity?

A process can be simplified as a sequence of system calls:

Protecting the operating system

How to model a process' activity?

A process can be simplified as a sequence of system calls:

- intercept system calls and their arguments,

Protecting the operating system

How to model a process' activity?

A process can be simplified as a sequence of system calls:

- intercept system calls and their arguments,
- group similar calls to make the problem feasible,

Protecting the operating system

How to model a process' activity?

A process can be simplified as a sequence of system calls:

- intercept system calls and their arguments,
- group similar calls to make the problem feasible,
- encode the sequence of classes of calls as a Markov chain,

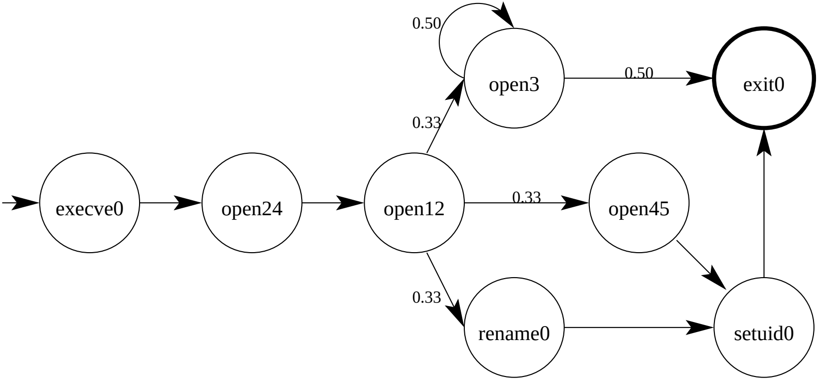
Protecting the operating system

How to model a process' activity?

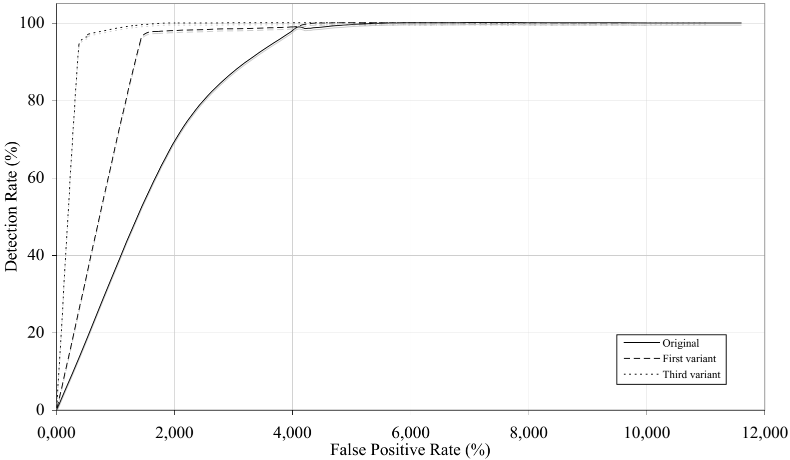
A process can be simplified as a sequence of system calls:

- intercept system calls and their arguments,
- group similar calls to make the problem feasible,
- encode the sequence of classes of calls as a Markov chain,
- deviant process $\rightarrow$ malicious process.

Example of model



Overall detection capabilities



Tested on one week of kernel activity (about 100,000 syscalls/day), 142 attacks. IEEE Transaction on Dep. and Secure Systems [4], ACM SIGOPS' O.S. Reviews [8].

Achieving better accuracy

Can we improve accuracy?

Achieving better accuracy

Can we improve accuracy?

Markov chains sometimes lead to false negatives (too permissive):

Achieving better accuracy

Can we improve accuracy?

Markov chains sometimes lead to false negatives (too permissive):

- use FSM instead,

Achieving better accuracy

Can we improve accuracy?

Markov chains sometimes lead to false negatives (too permissive):

- use FSM instead,
- avoid false positives due to string parameters by "compressing" them with Self-Organizing Maps.

Achieving better accuracy

Can we improve accuracy?

Markov chains sometimes lead to false negatives (too permissive):

- use FSM instead,
- avoid false positives due to string parameters by "compressing" them with Self-Organizing Maps.

Achieving better accuracy

Can we improve accuracy?

Markov chains sometimes lead to false negatives (too permissive):

- use FSM instead,
- avoid false positives due to string parameters by "compressing" them with Self-Organizing Maps.

Tested on one day of kernel activity (about 145,000 syscalls), 5 attacks. DIMVA 2009 [3].

3. Combination of the two

Aggregating alerts

Alert coming from different tools should be aggregated to avoid duplicates.

Aggregating alerts

Alert coming from different tools should be aggregated to avoid duplicates.

Time-based alert matching can be inaccurate, thus:

Aggregating alerts

Alert coming from different tools should be aggregated to avoid duplicates.

Time-based alert matching can be inaccurate, thus:

- model and alert as a fuzzy set (actually, interval),

Aggregating alerts

Alert coming from different tools should be aggregated to avoid duplicates.

Time-based alert matching can be inaccurate, thus:

- model and alert as a fuzzy set (actually, interval),
- model intrinsic measurement errors,

Aggregating alerts

Alert coming from different tools should be aggregated to avoid duplicates.

Time-based alert matching can be inaccurate, thus:

- model and alert as a fuzzy set (actually, interval),
- model intrinsic measurement errors,
- more robust than classic methods (e.g., sliding window).

Aggregating alerts

Alert coming from different tools should be aggregated to avoid duplicates.

Time-based alert matching can be inaccurate, thus:

- model and alert as a fuzzy set (actually, interval),
- model intrinsic measurement errors,
- more robust than classic methods (e.g., sliding window).

Aggregating alerts

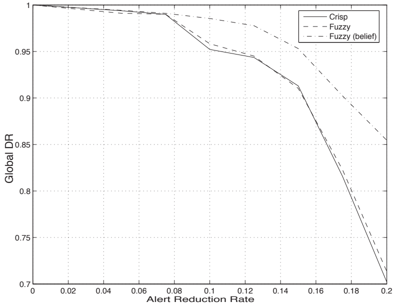
Alert coming from different tools should be aggregated to avoid duplicates.

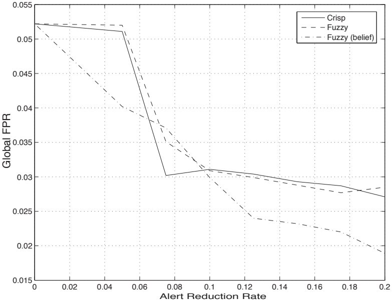
Time-based alert matching can be inaccurate, thus:

- model and alert as a fuzzy set (actually, interval),
- model intrinsic measurement errors,
- more robust than classic methods (e.g., sliding window).

Tested on about two weeks of detection resulting in about 2,000 alerts overall. Information Fusion, Elsevier [5].

Overall detection capabilities





Detecting related alerts

Malicious network behavior is reflected onto malicious kernel behavior.

Detecting related alerts

Malicious network behavior is reflected onto malicious kernel behavior.

How to detect relationships?

Detecting related alerts

Malicious network behavior is reflected onto malicious kernel behavior.

How to detect relationships?

- model alerts as stochastic processes,

Tested on about two weeks of detection resulting in about 1,000 alerts per system. RAID 2007 [7].

Detecting related alerts

Malicious network behavior is reflected onto malicious kernel behavior.

How to detect relationships?

- model alerts as stochastic processes,
- use statistical hypothesis tests (e.g., KS' goodness of fit),

Tested on about two weeks of detection resulting in about 1,000 alerts per system. RAID 2007 [7].

Detecting related alerts

Malicious network behavior is reflected onto malicious kernel behavior.

How to detect relationships?

- model alerts as stochastic processes,
- use statistical hypothesis tests (e.g., KS' goodness of fit),
- matching series $\rightarrow$ related alerts.

Tested on about two weeks of detection resulting in about 1,000 alerts per system. RAID 2007 [7].

Conclusions and lesson learned during my PhD

- some of our systems require refactoring because performance was not our primary focus,
- the most difficult task ever, in our research area, is gathering **enough experimental data**,
- often, scientifically sound experiments are very difficult to prepare because data is also non-labeled,
- in our future research we really want to spend a considerable amount of time and efforts at designing **public data collection infrastructure**.

Obligatory Slide

Thanks!
Questions?

References I

Privacy Rights Clearinghouse.
A chronology of data breaches.
Technical report, Privacy Rights Clearinghouse, July 2009.

Claudio Criscione, Federico Maggi, Guido Salvaneschi, and Stefano Zanero.
Integrated Detection of Attacks Against Browsers, Web Applications and Databases.
In *European Conference on Computer Network Defence, EC2ND*, 2009.

Alessandro Frossi, Federico Maggi, Gian Luigi Rizzo, and Stefano Zanero.
Selecting and Improving System Call Models for Anomaly Detection.
In Ulrich Flegel and Michael Meier, editors, *DIMVA*, Lecture Notes in Computer Science. Springer, 2009.

References II

Federico Maggi, Matteo Matteucci, and Stefano Zanero.
Detecting Intrusions through System Call Sequence and Argument Analysis.
IEEE Transactions on Dependable and Secure Computing (preprint), 99(1), 2009.

Federico Maggi, Matteo Matteucci, and Stefano Zanero.
Reducing False Positives In Anomaly Detectors Through Fuzzy Alert Aggregation.
Information Fusion, 10:300–311, October 2009.

Federico Maggi, William Robertson, Christopher Kruegel, and Giovanni Vigna.
Protecting a Moving Target: Addressing Web Application Concept Drift.
In Engin Kirda and Davide Balzarotti, editors, *RAID*, Lecture Notes in Computer Science. Springer, 2009.

References III

- Federico Maggi and Stefano Zanero.
  On the Use of Different Statistical Tests for Alert Correlation.
  In Christopher Kruegel, Richard Lippmann, and Andrew Clark, editors, *RAID*, volume 4637 of *Lecture Notes in Computer Science*, pages 167–177. Springer, 2007.
- Federico Maggi, Stefano Zanero, and Vincenzo Iozzo.
  Seeing the Invisible - Forensic Uses of Anomaly Detection and Machine Learning.
  ACM Operating Systems Review, April 2008.
- Ofer Shezaf and Jeremiah Grossman and Robert Auger.
  Web Hacking Incidents Database.
  http://www.xiom.com/whid-about, January 2009.

References IV

- William Robertson, Federico Maggi, Christopher Kruegel, and Giovanni Vigna.
  Effective Anomaly Detection with Scarce Training Data.
  In *Annual Network & Distributed System Security Symposium (accepted for publication)*, March 2010.
- Dean Turner, Marc Fossi, Eric Johnson, Trevor Mark, Joseph Blackbird, Stephen Entwise, Mo King Low, David McKinney, and Candid Wueest.
  Symantec Global Internet Security Threat Report – Trends for 2008.
  Technical Report XIV, Symantec Corporation, April 2009.