Machine Learning for Intrusion Detection

Dr. Konrad Rieck

Technische Universität Berlin

May 30, 2011

Outline

Hackers target home users for cash

'It's war out there' says Symantec report

Zero-Day-Exploit für Internet Explorer breitet sich aus

Critical security hole in Google Chrome

Kampf gegen Botnetze weitgehend ineffizient

SPIONAGENETZ

Hacker fangen E-Mails des Dalai Lama ab

Mittwoch, 7. April 2010 03:40

Visa und MasterCard betroffen Kommentare (0)

Hacker knacken Millionen Kreditkarten-Konten

Security Today

The Internet — a risk factor?

- Omnipresence of security threats and attacks
- Severe economic damage due to Internet crime
- Emergence of new criminal “industries”

Example: Careless users may fall victim to ...

- Credit card, password and identity theft
- Remote control of personal computers
- Involvement in crime as “stepping stone”

Basics of Security

Principle goals of computer security

1. Protection of *confidentiality* of resources
2. Protection of *integrity* of resources
3. Protection of *availability* of resources

Example: Suppose you send an email ...

Security Measures

Preventive security: Prevention and protection

- Access control and security policies
- Encryption, authentication and verification of data
- Redundancy and distribution of data

Reactive Security: Detection and response

- Anti-virus scanners and malware removal tools
- Intrusion detection and prevention systems
- Incident management and computer forensics

Intrusion Detection Systems (IDS)

Attack

Attempt to comprise the confidentiality, integrity or availibility

Intrusion detection system (IDS)

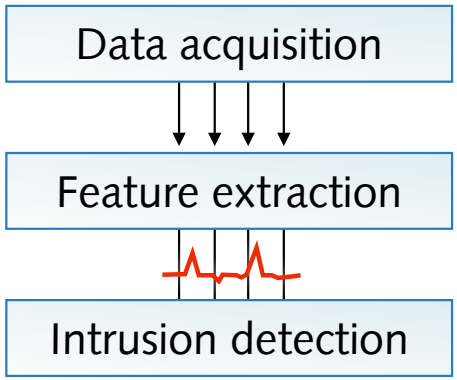
System monitoring a stream of events for attacks

Differentiation of IDS

- Event source $\longrightarrow$ host, network, application
- Analysis type $\longrightarrow$ signatures, rules, machine learning
- Response type $\longrightarrow$ messaging, blocking, sandboxing

Intrusion Detection in Detail

IDS Architecture



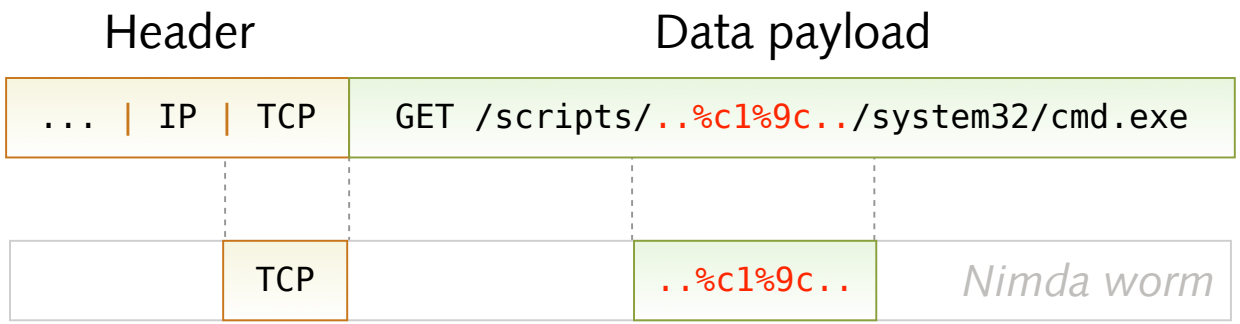
- **Data acquisition**
  Monitoring a stream of events, e.g. packets, system calls
- **Feature extraction**
  Extraction of features from events, e.g. addresses, users
- **Intrusion detection**
  Analysis of extracted features: misuse or anomaly detection

Classic Intrusion Detection

Identification of attacks using signatures

- Detection patterns, e.g., strings, regular expressions, rules
- Manual development of signatures from novel attacks
- Frequent update of signatures in intrusion detection systems

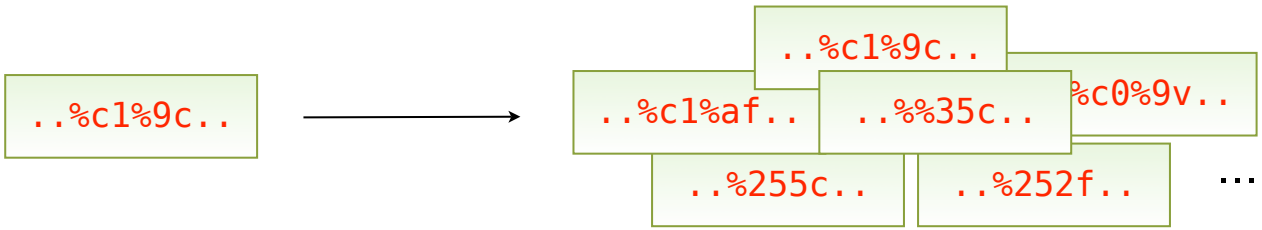
Example: Network packet and matching signature



Drawbacks of Signatures

Classic intrusion detection likely to fail in the future

- Inherent delay from discovery to availability of signature
- Unable to scale with diversity and amount of attacks
- Ineffective against novel and unknown attacks



Attack pattern Equivalent variants

→ *Need for automatic and adaptive detection technology*

Machine Learning for IDS

Application of machine learning for intrusion detection

- "Let computers learn to automatically detect attacks"
- Independent of signature generation and updates
- Complimentary to existing detection techniques

However: Not the average machine learning task

- Complex and structured data $\longrightarrow$ *Expressive features*
- Unknown and novel attacks $\longrightarrow$ *One-class learning*
- "Poisoned" training data $\longrightarrow$ *Robust learning*

Features for Intrusion Detection

Feature Extraction

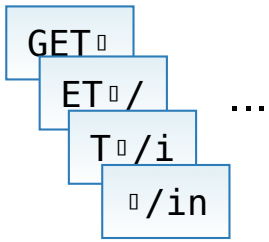
Event

$x =$ `GET index.html`

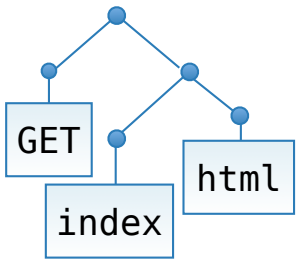
Extraction of features

Length	14
Entropy	3,4
Alpha.	12
Punct.	1

Numerical features (Vectors)



Sequential features (Sequences)



Structural features (Trees, graphs)

Numerical Features

Mapping of events to a vector space using numerical features

- Definition of a set F of numerical features
- Characterize event x by measuring numerical features
- Feature space $\equiv$ vector space of features F

Feature map

Function $\phi : \mathcal{X} \to \mathbb{R}^n$ mapping events $\mathcal{X}$ to $\mathbb{R}^n$ given by

$$x \longmapsto \begin{pmatrix} \phi_1(x) \\ \vdots \\ \phi_n(x) \end{pmatrix} \begin{matrix} \text{feature 1} \\ \vdots \\ \text{feature } n \end{matrix}$$

Numerical Features

Example: Numerical features for network payloads

```
GET vorlesung/ids.html HTTP/1.1
Host: www.ml.tu-berlin.de
User-Agent: Feuerfuchs 3.14
Connection: keep-alive
```

Simple numerical features

$\phi_1 =$	113	(Length)	$\phi_3 =$	105	(# Printable)
$\phi_2 =$	4.9	(Entropy)	$\phi_4 =$	8	(# Non-printable)

Often need for normalization of features (std-mean, min-max, ...)

Sequential Features

Mapping of events to a vector space using sequential features

- Event x is sequence of symbols from *alphabet* $\mathcal{A}$
- Characterize x using an *embedding language* $L \subseteq \mathcal{A}^*$
- Feature space spanned by frequencies of words $w \in L$

Feature map

Function $\phi : \mathcal{A}^* \to \mathbb{R}^{|L|}$ mapping sequences to $\mathbb{R}^{|L|}$ given by

$$x \longmapsto (\#_w(x))_{w \in L}$$

where $\#_w(x)$ returns the frequency of w in sequence x.

Sequential Features

Common embedding language: $L = \mathcal{A}^n$ (N-grams)
$\longrightarrow$ Independent of attack and protocol characteristics

Example: 2-grams extracted from network payloads

```
GET vorlesung/ids.html HTTP/1.1
Host: www.ml.tu-berlin.de
User-Agent: Feuerfuchs 3.14
Connection: keep-alive
```

$$\phi(x) = \underbrace{(\ldots, 0, 0, \overset{\texttt{er}}{3}, 0, 0, \ldots, 0, 0, \overset{\texttt{ml}}{2}, 0, 0, \ldots)}_{\text{Space of all 2-grams } (L = \mathcal{A}^2)}$$

Structural Features

Mapping of events to a vector space using structural features

- Event x is a structure (tree,graph) of labeled nodes
- Characterize event x using contained substructures
- Binary feature space spanned by substructures $s \in \mathcal{S}$

Feature map

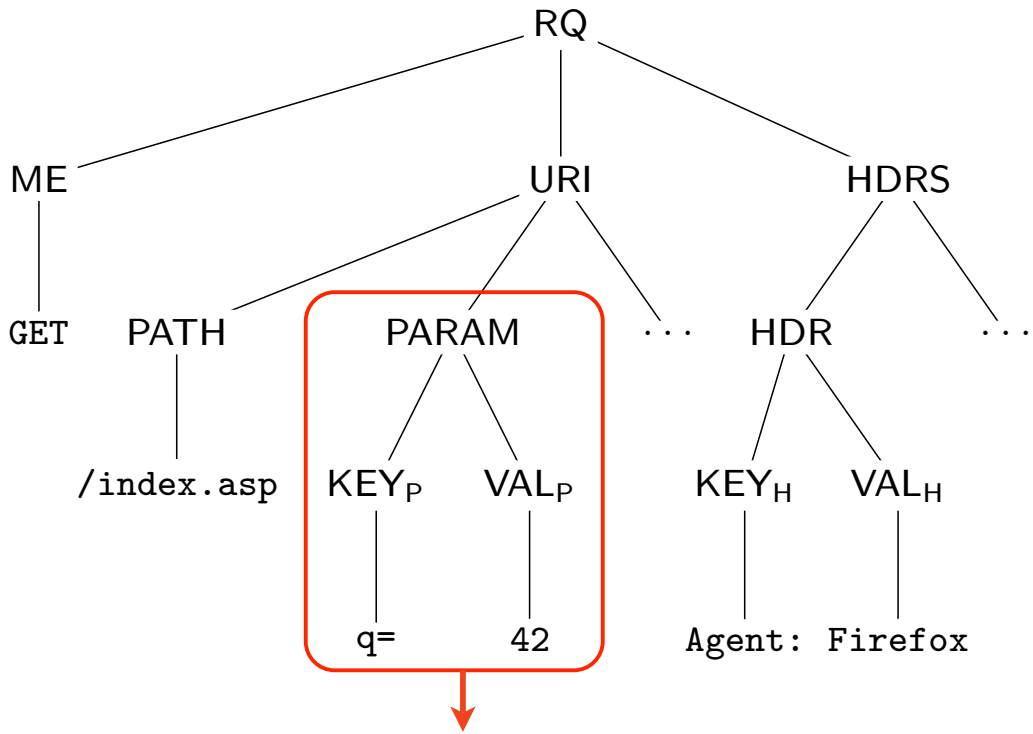
Function $\phi : \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{S}|}$ mapping structures to $\mathbb{R}^{|\mathcal{S}|}$ given by

$$x \longmapsto (\mathbb{I}_s(x))_{s \in \mathcal{S}}$$

where $\mathbb{I}_s(x)$ indicates if s is a substructure of x.

Structural Features

Example: Extraction of parse trees for the HTTP protocol



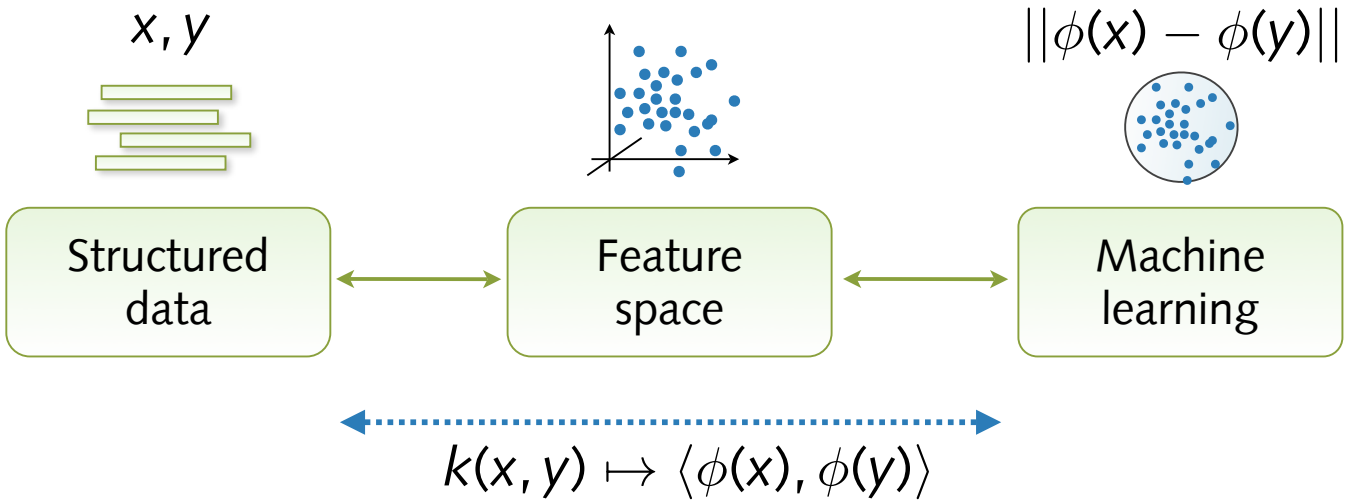
$$\phi(x) = \underbrace{(\ldots, 0, 1, 0, 0, 0, 0, 0, 0, 0, \ldots)}_{\text{Space of all subtrees}}$$

From Features to Kernels

The feature space dilemma: expressiveness $\leftrightarrow$ efficiency

Concept of kernel functions

- Implicit access to feature space using kernel functions
- Efficient kernels for vectors, sequences, trees, graphs, ...



Learning for Intrusion Detection

Learning Intrusion Detection?

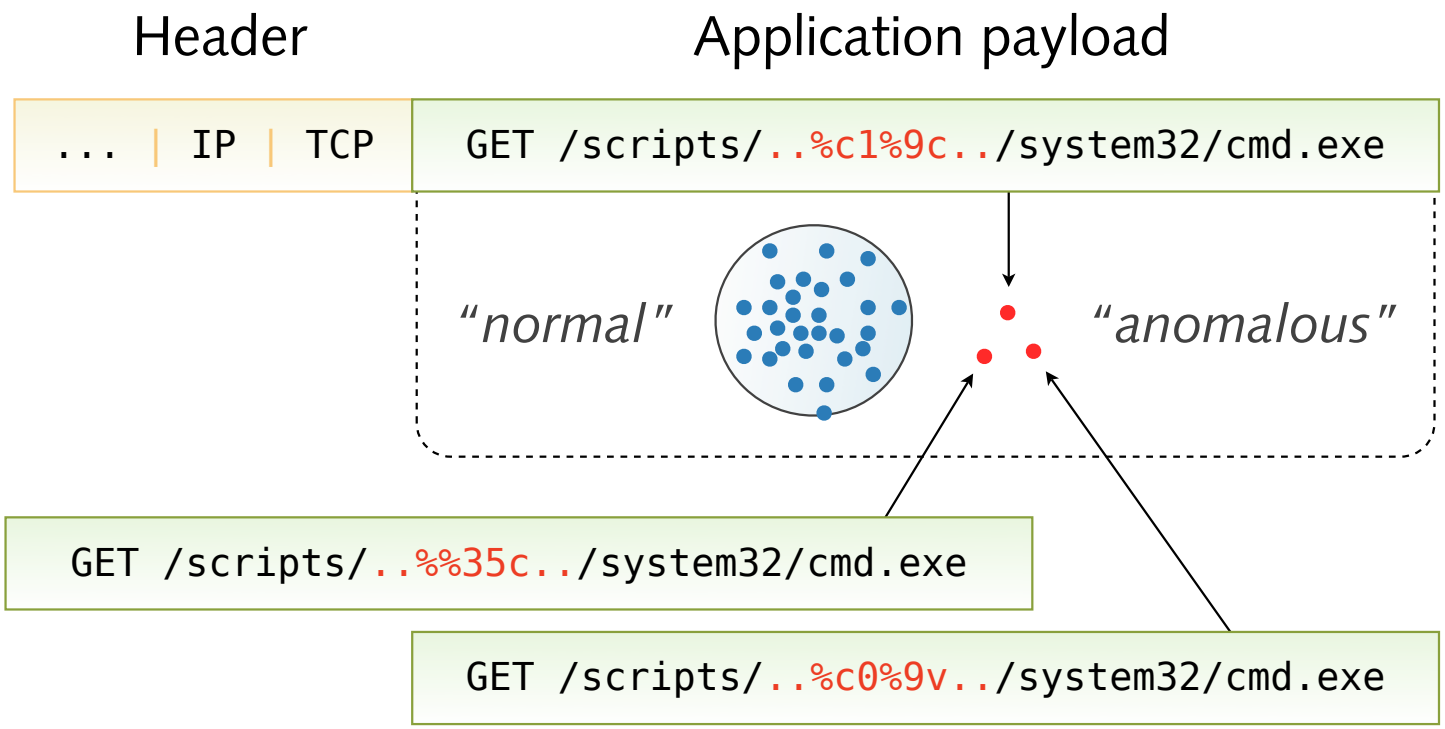
Learning setup

- No knowledge about future attacks
  Anomaly detection: *Learn model of normality*
- Labeling real data expensive
  Unsupervised learning: *Learn model on dirty data*

Underlying assumptions

- Attacks deviate from normal data
- Normal data is predominant

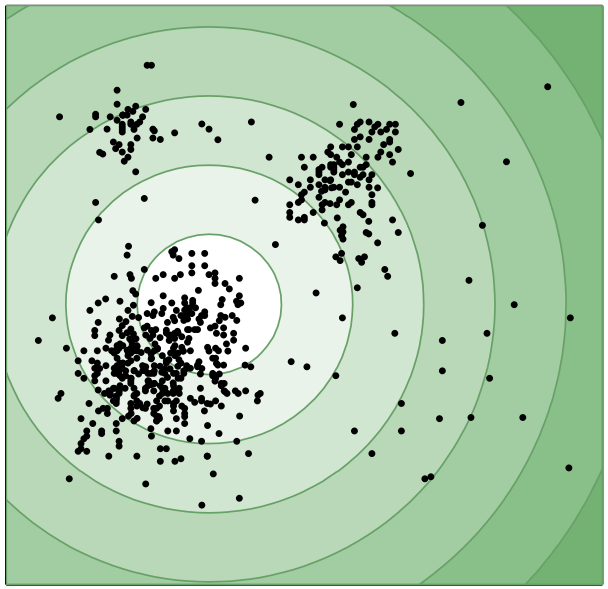
Anomaly Detection



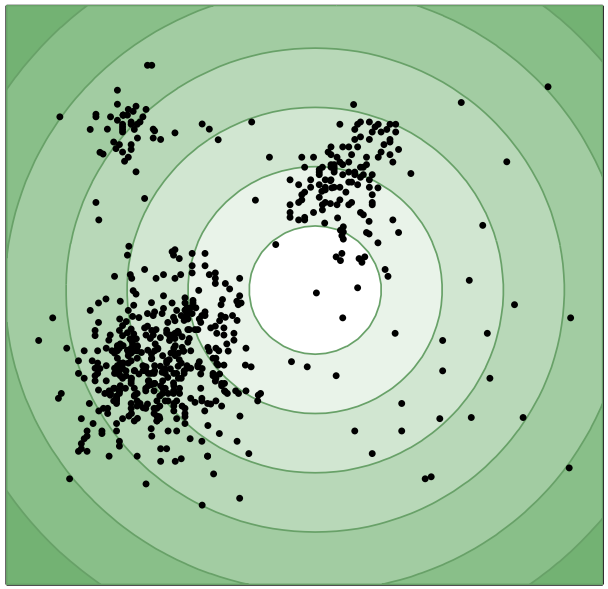
Hyperspheres for Anomaly Detection

Concept: Model data using hypersphere in feature space

Deviation of normality = Distance from center of hypersphere



(a) Center of Mass



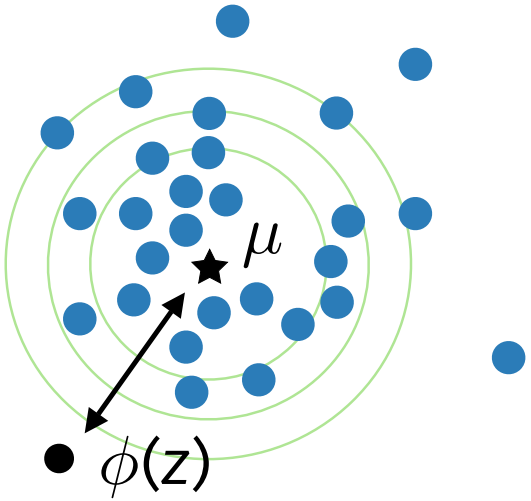
(b) One-Class SVM

Center of Mass

Model normality using hypersphere at center of mass

- For $\{x_1, \ldots x_n\}$ the center of mass is

$$\mu = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i)$$



- Anomaly score $a(z)$ of new point z

$$a(z) = ||\phi(z) - \mu||^2$$

$$= k(z,z) - \frac{2}{n} \sum_{i=1}^{n} k(z, x_i) + \frac{1}{n^2} \sum_{i,j=1} k(x_i, x_j)$$

One-class SVM

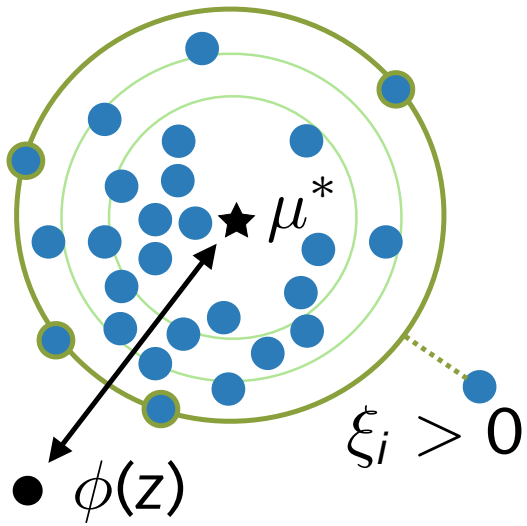
Model normality using hypersphere with mimimum volume

- Seek hypersphere with center μ^*

$$\min_{\mu,r,\xi} \; r^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{subject to} \; ||\phi(x_i) - \mu||^2 \leq r^2 + \xi_i$$

$$\xi_i \geq 0 \;\; \text{for } i = 1, \ldots, n$$



- Anomaly score $a(z)$ of new point z

$$a(z) = ||\phi(z) - \mu^*||$$

$$= k(z,z) - 2\sum_{i=1}^{n} \alpha_i k(x_i, z) + \sum_{i,j=1}^{n} \alpha_i \alpha_j k(x_i, x_j).$$

One-Class SVM in Dual

Dual formulation of One-class SVM

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i k(x_i, x_i) - \sum_{i,j=1}^{n} \alpha_i \alpha_j k(x_i, x_j)$$

$$\text{subject to} \quad \sum_{i=1}^{n} \alpha_i = 1 \quad \text{and} \quad 0 \leq \alpha_i \leq C \quad \text{for } i = 1, \ldots, n$$

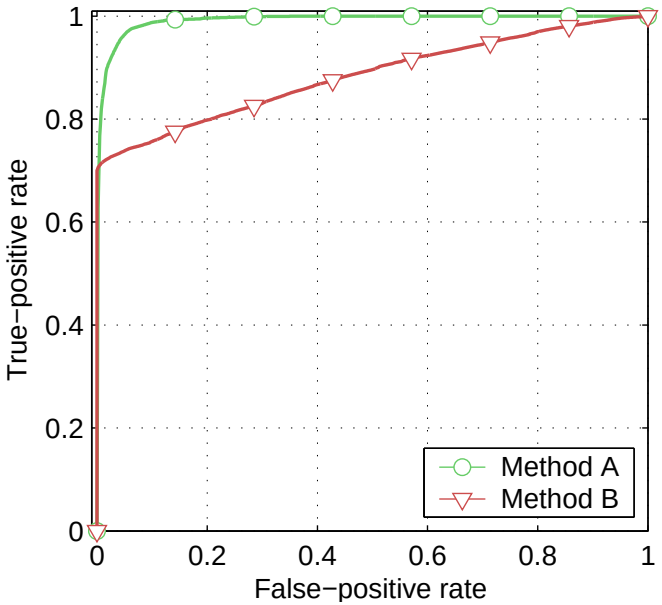
One-class SVM = Quadratic program with linear constraints

- Very similar to two-class formulation of SVM
- Efficient computation with standard SVM/QP solvers

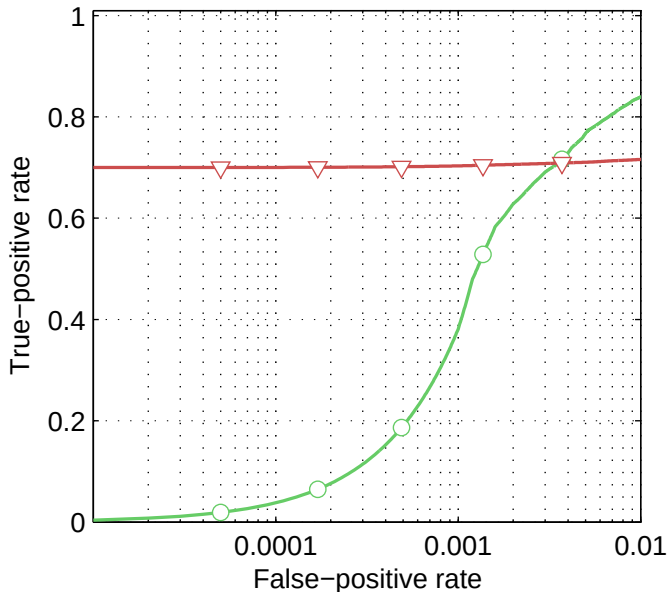
Results and Perspectives

ROC Curves

Receiver Operating Characteristic (ROC) Curves



(c) Full ROC curve



(d) Bounded ROC curve

- Depiction of true-positive and false-positive rates
- AUC_x = Area under ROC curve with false-positive rate $\leq x$

Comparison of Features

Evaluation of features for network intrusion detection

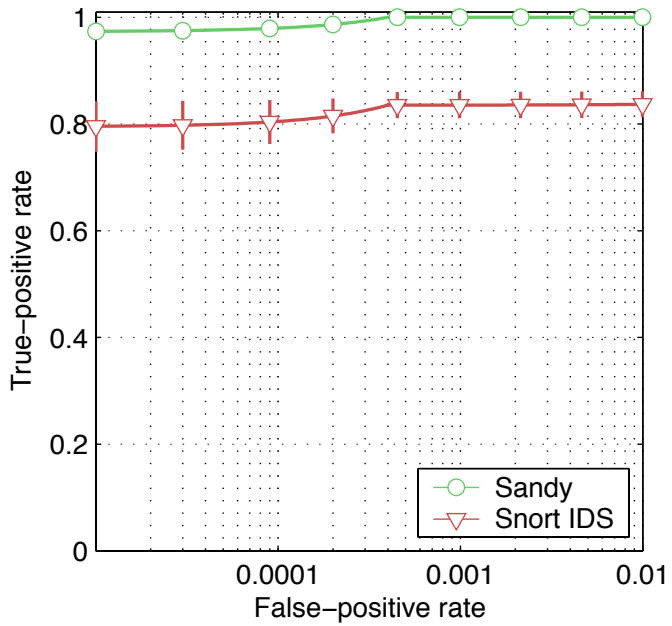
- 10 days of network traffic for HTTP and FTP protocol
- Artificial injection of recent network attacks
- One-class SVM as anomaly detection method

	Numerical	Sequential	Structural
HTTP	0.77 ± 0.03	0.99 ± 0.00	0.87 ± 0.03
FTP	0.99 ± 0.00	0.99 ± 0.00	0.28 ± 0.03

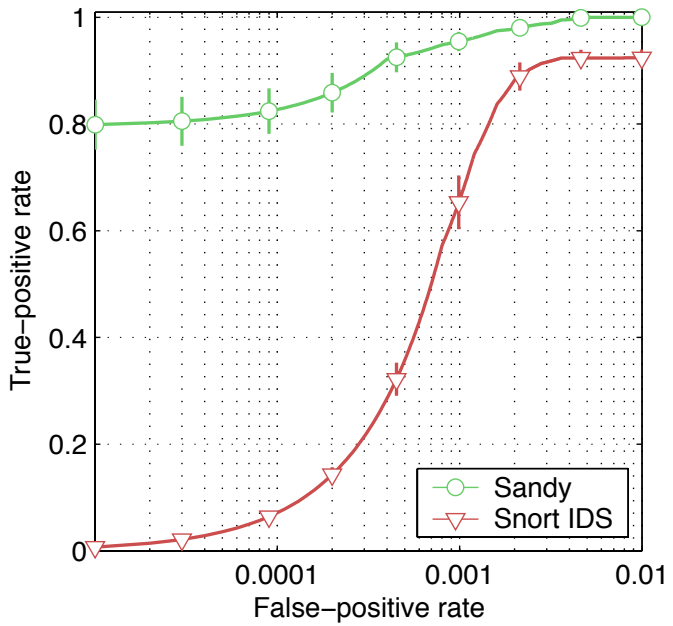
- Detection performance as $AUC_{0.01}$ with standard error
- Sequential features provide almost perfect detection!

Comparison with signature-based IDS

Comparative evaluation with popular IDS "Snort"



(e) HTTP protocol



(f) FTP protocol

- Prototype "Sandy" significantly outperforms Snort
- Detection of 80-97% attacks with >0.001% false-positives

Conclusions and Outlook

Machine learning for intrusion detection

- Intelligent detection of attacks using machine learning
- Incorporation of numerical, sequential and structural features
- Effective anomaly detection using hyperspheres

Futher applications of machine learning for security

1. Analysis of malicious software (Worms, Botnets, ...)
2. Detection of Trojan horses and their communication
3. Automatic generation of attack signatures

Interesting topics for bachelor, master and diploma thesis...

References

Laskov, P., Rieck, K., and Müller, K.-R. (2008).
Machine learning for intrusion detection.
In *Mining Massive Data Sets for Security*, pages 366–373. IOS press.

Rieck, K. (2009).
Machine Learning for Application-Layer Intrusion Detection.
Lulu Enterprises, Inc (lulu.com).

Schölkopf, B. and Smola, A. (2002).
Learning with Kernels.
MIT Press, Cambridge, MA.

Shawe-Taylor, J. and Cristianini, N. (2004).
Kernel methods for pattern analysis.
Cambridge University Press.