Data Mining for Computer Security 1
Konrad Rieck
Technische Universität Braunschweig, Germany
About me

- **Professor of Computer Science at TU Braunschweig**
  - Fun with security and machine learning for 15 years
  - Head of Institute of System Security (~10 people)

- More on our website: http://www.tu-bs.de/sec
Computer Security Today

- **Classic security cycle**
  - Prevention, e.g. authentication
  - Detection, e.g. virus scanners
  - Analysis, e.g. digital forensics

- **Security cycle out of balance**
  - Increasing amount and diversity of attacks
  - Larger attack surfaces due to system complexity
  - **Bottleneck**: manual analysis of security data
Our Research

- **Security systems with more “intelligence”**
  - Application of data mining and machine learning
  - Assistance during prevention, detection and analysis
  - Human out of the loop — but not without control

Automatisation of attacks
➡️ Automatisation of defenses?
Some of Our Work

- **Prevention:** Discovery of vulnerabilities in software
  - Graph mining for finding vulnerable code patterns *(S&P ’14, ’15)*
  - Identification of missing security checks *(CCS ’13)*

- **Detection:** Identification of attacks and malicious code
  - Detection of malicious Android applications *(NDSS ’14)*
  - Detection of malicious Flash animations *(DIMVA ’16, Best Paper Award)*

- **Analysis:** Understanding malicious code
  - Analysis of ultrasonic side channels in Android *(Euro S&P ’17)*
  - Authorship attribution of native program code *(?)*
Some of Our Work

- **Prevention:** Discovery of vulnerabilities in software
  - Graph mining for finding vulnerable code patterns *(S&P ’14, ’15)*
  - Identification of missing security checks *(CCS ’13)*

- **Detection:** Identification of attacks and malicious code
  - Detection of malicious Android applications *(NDSS ’14)*
  - Detection of malicious Flash animations *(DIMVA ’16, Best Paper Award)*

- **Analysis:** Understanding malicious code
  - Analysis of ultrasonic side channels in Android *(Euro S&P ’17)*
  - Authorship attribution of native program code *(?)*
Some of Our Work

**Prevention:** Discovery of vulnerabilities in software
- Graph mining for finding vulnerable code patterns *(S&P ’14, ’15)* ✓
- Identification of missing security checks *(CCS ’13)* ✓

**Detection:** Identification of attacks and malicious code
- Detection of malicious Android applications *(NDSS ’14)* ✓
- Detection of malicious Flash animations *(DIMVA ’16, Best Paper Award)* ✓

**Analysis:** Understanding malicious code
- Analysis of ultrasonic side channels in Android *(Euro S&P ’17)* ✓
- Authorship attribution of native program code (?) ✓
Let’s go ...

• **A generic view on learning**
  - How learning works in general (theoretically)

• **Types of machine learning**
  - Different types of machine learning techniques

• **Some learning algorithms**
  - Implementations of machine learning

• A complete lecture condensed into two sessions. Good luck! 😊
Machine Learning in a Nutshell
Machine Learning?

- **Machine learning** = branch of artificial intelligence
- Computer science intersecting with statistics
- No science fiction, please! We’re talking about algorithms.
Machine Learning

• **Theory and practice of making computers learn**
  - Automatic inference of dependencies from data
  - Generalization of dependencies; not simple memorization
  - Application of learned dependencies to unseen data

• **Example: Handwriting recognition**
  - Dependencies: written shapes $\leftrightarrow$ concrete letters
Influences

• Where does machine learning come from?
  • Interdisciplinary branch of computer science
  • Close relation to artificial intelligence and data mining

• Different inspirations for learning, e.g. neurology, physics, ...
• Large diversity of approaches, concepts and algorithms
Intrusion Detection

- **Network intrusion detection**
  - Detection of attacks in network payloads
  - Classic approach: signature-based detection
  - Running example in this talk

- **Network packet and matching signature**

<table>
<thead>
<tr>
<th>Headers</th>
<th>Packet payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>GET /scripts/..%c1%9c../system32/cmd.exe</td>
</tr>
<tr>
<td>IP</td>
<td>TCP</td>
</tr>
<tr>
<td>TCP</td>
<td>Nimda worm</td>
</tr>
</tbody>
</table>
Feature Spaces

- Machine learning usually defined over vector spaces
  - Security data almost never in form of vectors
  - Key for learning in security → a map to a feature space

- Representation of real objects using features
Feature Extraction

Network payload

\[ x = \text{GET index.html} \]

Feature extraction

Numerical features (Vectors)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>14</td>
</tr>
<tr>
<td>Entropy</td>
<td>3.4</td>
</tr>
<tr>
<td>Alpha</td>
<td>12</td>
</tr>
<tr>
<td>Punct.</td>
<td>1</td>
</tr>
</tbody>
</table>

Sequential features (Strings)

GET /index.html
GET /
GET /in

Structural features (Trees, Graphs)

Running example

GET index.html
A Learning Model

- **What can we learn?**
  - Inference of functional dependencies from data \((X \leftrightarrow Y)\)
  - Dependencies described by a learning model \(\theta\)
  - Model \(\theta\) parameterizes a prediction function \(f_\theta : X \rightarrow Y\)

- **A simple example**
  - \(X = \text{color } \times \text{height} \text{ of fruits}\)
  - \(Y = \{\text{apple, pear}\}\)
  - \(\theta = (\text{color, height})\) and bias
Examples: **Learning Models**

- Quadratic functions
- Other non-linear functions
- Decision stumps
Learning Function

• **Learning process**
  • Searching the space $\Theta$ for good models (functions $f_\theta$)

• **Supervised learning** *(with labels)*
  • Learning function $g : X \times Y \rightarrow \Theta$
  • “You know what you are looking for”

• **Unsupervised learning** *(without labels)*
  • Learning function $g : X \rightarrow \Theta$
  • “You don’t know what you are looking for”
Learning and Errors

- Learning process guided by errors
  - Minimal error of learning model $\theta$ desirable
  - Quantification of disagreement between predictions and truth
  - Different strategies for reducing errors

3 errors
Learning and Errors

- Learning process guided by errors
  - Minimal error of learning model $\theta$ desirable
  - Quantification of disagreement between predictions and truth
  - Different strategies for reducing errors
Learning and Errors

- **Learning process guided by errors**
  - Minimal error of learning model $\theta$ desirable
  - Quantification of disagreement between predictions and truth
  - Different strategies for reducing errors

![Diagram showing learning process with 3 errors initially, reducing to 3 errors, and finally to no errors.](image-url)
Test Data and Overfitting

- **Training and test data**
  - Model learned on *training data*; prediction on unseen *test data*
  - Optimizing the error on training data dangerous

![Diagram showing training data and model fit.]
Test Data and Overfitting

- **Training and test data**
  - Model learned on training data; prediction on unseen test data
  - Optimizing the error on training data dangerous
Regularization

- Regularization key to effective learning
  - Danger of adapting learning model to training data only
  - Balancing of training error and model complexity
  - Examples: Costs of SVMs, pruning in decision trees, ...

```
Risk (error)  

Complexity term

Training error

Complexity of learning model

Underfitting  Overfitting

Test error
```
Types of Machine Learning
Supervised: **Classification**

- **Learning to categorize objects into known classes**
  - Discrimination of objects using learning model
  - Output domain often $Y = \{-1, +1\}$ or $\{1,2,3,\ldots\}$

- **Examples**
  - Handwriting recognition
  - Spam filtering in emails

- **Common algorithms**
  - SVM, KNN, Neural Networks, ...
Classification

- Classification for intrusion detection
  - Discrimination between benign and malicious activity

```
... | IP | TCP
```

```
GET /scripts/..%35c../system32/cmd.exe
```

“benign”  “malicious”
Classification

- **Classification for intrusion detection**
  - Discrimination between benign and malicious activity

<table>
<thead>
<tr>
<th>Header</th>
<th>Data payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>GET /scripts/..%35c../system32/cmd.exe</td>
</tr>
<tr>
<td>IP</td>
<td>“benign”</td>
</tr>
<tr>
<td>TCP</td>
<td>“malicious”</td>
</tr>
</tbody>
</table>

| GET /scripts/..%c1%af../system32/cmd.exe |

Running example
Classification

- Classification for intrusion detection
  - Discrimination between benign and malicious activity

<table>
<thead>
<tr>
<th>Header</th>
<th>Data payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>GET /scripts/..%35c../system32/cmd.exe</td>
</tr>
<tr>
<td>IP</td>
<td>IP</td>
</tr>
<tr>
<td>“benign”</td>
<td>“malicious”</td>
</tr>
<tr>
<td>GET /scripts/..%c1%af../system32/cmd.exe</td>
<td></td>
</tr>
<tr>
<td>GET /scripts/..%255c../system32/cmd.exe</td>
<td></td>
</tr>
</tbody>
</table>
Unsupervised: Clustering

- **Grouping of similar objects into clusters**
  - Contrast to classification: clusters not known at start
  - Output domain $Y = \{1, 2, 3, \ldots\}$ (~ permutations)

- **Examples**
  - Comparison of species
  - Malware analysis

- **Common learning algorithms**
  - K-means, linkage clustering, ...
Clustering

- Clustering of network payloads for later analysis
  - Unsupervised grouping of similar payloads into clusters

<table>
<thead>
<tr>
<th>Header</th>
<th>Data payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>IP</td>
</tr>
</tbody>
</table>

Running example

GET /scripts/..%35c../system32/cmd.exe
Clustering

- Clustering of network payloads for later analysis
  - Unsupervised grouping of similar payloads into clusters

<table>
<thead>
<tr>
<th>Header</th>
<th>Data payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>IP</td>
</tr>
</tbody>
</table>

Attack Z
Clustering

- Clustering of network payloads for later analysis
  - Unsupervised grouping of similar payloads into clusters

```
... | IP | TCP
---|----|----
GET /scripts/..%35c../system32/cmd.exe
```

```
GET /php-fun/?-s
```

Running example

- Attack Y
- Attack Z
Clustering

- Clustering of network payloads for later analysis
  - Unsupervised grouping of similar payloads into clusters

Running example

| GET /scripts/%35c/system32/cmd.exe |
| GET /IISWebAgentIF.dll?overflow |
| GET /php-fun/?-s |

Header | Data payload

... | IP | TCP
Unsupervised: Anomaly Detection

• Detection of deviations from learned model of normality
  • Generative or discriminative models of normality
  • Output domain often $Y = [0,1]$ (anomaly score)

• Examples
  • Engine failure detection
  • Intrusion detection

• Common approaches
  • Statistics, one-class SVM, ...
Anomaly Detection

- **Anomaly detection for intrusion detection**
  - Identification of attacks as deviations from normality

<table>
<thead>
<tr>
<th>Header</th>
<th>Data payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>GET /scripts/..%35c../system32/cmd.exe</td>
</tr>
<tr>
<td>IP</td>
<td>TCP</td>
</tr>
</tbody>
</table>

“normal” “anomalous”
Anomaly Detection

- Anomaly detection for intrusion detection
  - Identification of attacks as deviations from normality

```
   | IP | TCP
---|----|----
GET /scripts/..%35c../system32/cmd.exe
```

```
GET /scripts/..%c1%af../system32/cmd.exe
```

“normal”  “anomalous”
Anomaly Detection

- Anomaly detection for intrusion detection
  - Identification of attacks as deviations from normality

```
GET /scripts/..%35c../system32/cmd.exe
```

```
GET /scripts/..%c1%af../system32/cmd.exe
```

```
GET /scripts/..%255c../system32/cmd.exe
```

Running example
Supervised: **Regression**

- **Learning to predict a numerical property (score)**
  - Approximation of observed function by learning model
  - Output domain usually $Y = \mathbb{R}$

- **Examples**
  - Temperature forecasting
  - Stock market prediction

- **Common algorithms**
  - Logistic & ridge regression, ...
Dimension Reduction

- Supervised or unsupervised reduction of dimensionality
  - Extraction of more informative features for objects
  - $X = \mathbb{R}^N$ and $Y = \mathbb{R}^M$ with $N \gg M$

- Examples
  - Visualisation and denoising
  - Vulnerability discovery

- Common learning algorithms
  - PCA, LLE, NMF, ...
Some Learning Algorithms
K-Nearest Neighbors

- **Learning using the local neighborhood of data**
  - Most intuitive and oldest learning algorithm
  - Learning = not really ...training data is just stored
  - Regularization = size of considered neighborhood
  - Prediction = labels of neighborhood

![Neighborhood k = 4](image1)

![Neighborhood k = 11](image2)
K-Nearest Neighbors

- Learning using the local neighborhood of data
  - Most intuitive and oldest learning algorithm
  - Learning = not really ...training data is just stored
  - Regularization = size of considered neighborhood
  - Prediction = labels of neighborhood

Neighborhood $k = 4$

Neighborhood $k = 11$
Neural Networks

- **Learning using a network of artificial neurons**
  - Classic method inspired by biological neural networks (~1940)
  - Learning = adaption of weights of neural network
  - Regularization = brain damage or weight decay
  - Prediction = forward pass through neural network

[Diagram of neural network with input, hidden layers, and output]

**Deep Learning:** Recent revival of neural networks with several different hidden layers
Decision Trees

- **Learning by composition of simple logic predicates**
  - Classic method inspired by decision making (~1960)
  - Learning = inductive composition of tree nodes
  - Regularization = pruning of subtrees
  - Prediction = top-down pass through tree

Random Forests: Ensemble of decision trees, each learned on randomly selected features
Support Vector Machines

- **Learning using a hyperplane in a kernel feature space**
  - Modern method inspired by learning theory (~1990)
  - Learning = convex problem for determining hyperplane
  - Regularization = softening of hyperplane for outliers
  - Prediction = orientation to hyperplane

![Diagram showing input space and feature space transformation](image-url)
Support Vector Machines

- **Learning using a hyperplane in a kernel feature space**
  - Modern method inspired by learning theory (~1990)
  - Learning = convex problem for determining hyperplane
  - Regularization = softening of hyperplane for outliers
  - Prediction = orientation to hyperplane

![Diagram of input space, kernel map, feature space, and hyperplane with maximum margin]
Several Other Methods

- Several other learning methods
  - Probabilistic models
  - Boosting and bagging
  - Genetic algorithms
  - ...

- Several other learning concepts
  - Reinforcement learning
  - ...

Duda, Hart and Stork: Pattern Classification Wiley & Sons 2001


The Standard

Kernel Methods
Summary
Summary

• **Current problems of computer security**
  - Increasing automatization of attacks and malware
  - Large amounts of novel malicious code
  - Defenses involving manual analysis often ineffective

• **Machine learning in computer security**
  - Adaptive defenses using learning algorithms
  - Automatic detection and analysis of threats
  - Assisted analysis of threats, e.g. vulnerabilities
Thank you! Questions?