Moduldokumentation

Modul Applikationssicherheit (apsi)

Simon Wächter

2019

Inhalt

[1 Einleitung 4](#_Toc21376989)

[1.1 Einleitung 4](#_Toc21376990)

[1.2 Lernziele 4](#_Toc21376991)

[1.3 Prüfungen 4](#_Toc21376992)

[2 Woche 1 5](#_Toc21376993)

[2.1 Agenda 5](#_Toc21376994)

[2.2 Administrative Issues 5](#_Toc21376995)

[2.3 Who is Your Lecturer (Wagner) 5](#_Toc21376996)

[2.4 whoami (Schläpfer) 6](#_Toc21376997)

[2.5 Why is Application Security Important? 6](#_Toc21376998)

[2.6 Lecture Overview (1) 7](#_Toc21376999)

[2.7 Lecture Overview (2) 7](#_Toc21377000)

[2.8 Lecture Overview (3) 8](#_Toc21377001)

[2.9 Example: Buffer Overflow 8](#_Toc21377002)

[2.10 How does a Buffer Overflow Work? 8](#_Toc21377003)

[2.11 Buffer Overflow Demo 9](#_Toc21377004)

[2.12 Not Really a Fix for Buffer Overflow: Stack Protection 9](#_Toc21377005)

[2.13 Fix for Buffer Overflow: Checked Buffers 9](#_Toc21377006)

[2.14 Example: Taint Checking in Perl (1) 10](#_Toc21377007)

[2.15 Example: Taint Checking (2) 10](#_Toc21377008)

[2.16 Example: Taint Checking (3) 11](#_Toc21377009)

[2.17 Example: Taint Checking (4) 11](#_Toc21377010)

[2.18 Example: Data Leakage by Behavior (1) 11](#_Toc21377011)

[2.19 Example: Data Leakage by Behavior (2) 12](#_Toc21377012)

[2.20 Example: Data Leakage by Error Output 12](#_Toc21377013)

[2.21 Human Factors 13](#_Toc21377014)

[2.22 Exercises and Preparation for Next Week 13](#_Toc21377015)

[2.23 Reserve 13](#_Toc21377016)

[3 Woche 2 14](#_Toc21377017)

[3.1 Agenda 14](#_Toc21377018)

[3.2 Recap: What do Attackers Want? 14](#_Toc21377019)

[3.3 Buffer Overflow II 14](#_Toc21377020)

[3.4 Buffer Overflow II 15](#_Toc21377021)

[3.5 Buffer Overflow II 15](#_Toc21377022)

[3.6 Buffer Overflow II 16](#_Toc21377023)

[3.7 Buffer Overflow II 16](#_Toc21377024)

[3.8 Attack Technique: Fuzzing 17](#_Toc21377025)

[3.9 Error Handling 17](#_Toc21377026)

[3.10 Case study – Handling too long input 18](#_Toc21377027)

[3.11 Case-study: Compression Bomb 18](#_Toc21377028)

[3.12 Error Handling 19](#_Toc21377029)

[3.13 Algorithmic Complexity Attacks 20](#_Toc21377030)

[3.14 Algorithmic Complexity: Sorting 20](#_Toc21377031)

[3.15 Algorithmic Complexity: Hashing 21](#_Toc21377032)

[3.16 Algorithmic Complexity: Regular Expressions 21](#_Toc21377033)

[4 Woche 3 22](#_Toc21377034)

[4.1 Agenda 22](#_Toc21377035)

[4.2 Secure Coding Principle: Test-before-use 22](#_Toc21377036)

[4.3 Input Validation 22](#_Toc21377037)

[4.4 Input Normalization 23](#_Toc21377038)

[4.5 Unix Permission Model (simplified) 23](#_Toc21377039)

[4.6 Chroot 24](#_Toc21377040)

[4.7 Mandatory Access Control (MAC) 24](#_Toc21377041)

[4.8 MAC Example: SELinux 25](#_Toc21377042)

[4.9 MAC Example: AppArmor 25](#_Toc21377043)

[4.10 Secure Coding Principle: Defense in Depth 26](#_Toc21377044)

[4.11 Privilege Separation 26](#_Toc21377045)

[4.12 Overall Idea 27](#_Toc21377046)

[4.13 Form of the Modules 27](#_Toc21377047)

[4.14 Example: OpenSSH 28](#_Toc21377048)

[4.15 OpenSSH: Privilege Separation Approach 28](#_Toc21377049)

[4.16 Example: Postfix MTA (Email Server) 28](#_Toc21377050)

[4.17 Postfix System View 29](#_Toc21377051)

[4.18 Postfix: Example Data-Flow 29](#_Toc21377052)

[5 Woche 4 30](#_Toc21377053)

[5.1 Agenda 30](#_Toc21377054)

[5.2 Security Testing (for Software) 30](#_Toc21377055)

[5.3 Pen-Test 30](#_Toc21377056)

[5.4 How does a Pen-Test fail? 31](#_Toc21377057)

[5.5 Code Inspection and Review 31](#_Toc21377058)

[5.6 Code Scanners 32](#_Toc21377059)

[5.7 Execution Emulators 32](#_Toc21377060)

[5.8 Patching 33](#_Toc21377061)

[5.9 How to Patch 33](#_Toc21377062)

[5.10 Problems with Patching 34](#_Toc21377063)

[5.11 Problems with not Patching 34](#_Toc21377064)

[5.12 Problems with Unavailability of Patches 34](#_Toc21377065)

[5.13 Economic Aspects of Security 35](#_Toc21377066)

[5.14 Case Study: DDoS on Brian Krebs (20.9.2016) 35](#_Toc21377067)

[5.15 What can Perform Such an Attack? 35](#_Toc21377068)

[5.16 IoT: Internet of insecure Things 36](#_Toc21377069)

[5.17 Why are These IoT Devices so Insecure? 36](#_Toc21377070)

[5.18 Possible Solutions 37](#_Toc21377071)

[5.19 Does it Pay to make Software Secure? 37](#_Toc21377072)

[5.20 So, what to do? 38](#_Toc21377073)

[6 Woche 5 39](#_Toc21377074)

# Einleitung

## Einleitung

Dieses Dokument stellt die Moduldokumentation für das Modul apsi dar. Allfällige Unterlagen sind im Modulordner zu finden.

## Lernziele

Das Modul beinhaltet folgende Lernziele:

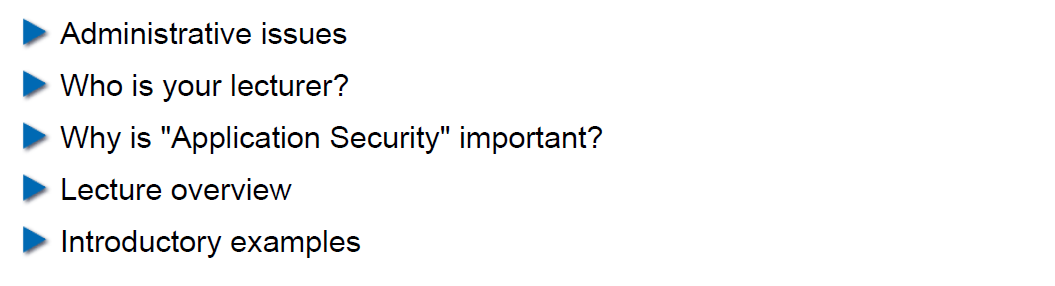
* Kennen typische Bedrohungen von Applikationen sowie ihren Daten und können abschätzen, wieweit diese für durch sie zu entwickelnde Software ein Sicherheitsrisiko darstellen
* Kennen typische Sicherheitsprobleme und können sie vermeiden
* Sind in der Lage, einschlägige Quellen zu konsultieren, um neu veröffentlichte Sicherheitsprobleme zu erfassen und zu verstehen, wie diese zu vermeiden sind
* Verfügen über Konzeptwissen, nicht nur Produktwissen, um das Gelernte auch in anderen Umfeldern und mit anderen Programmiersprachen anwenden zu können

## Prüfungen

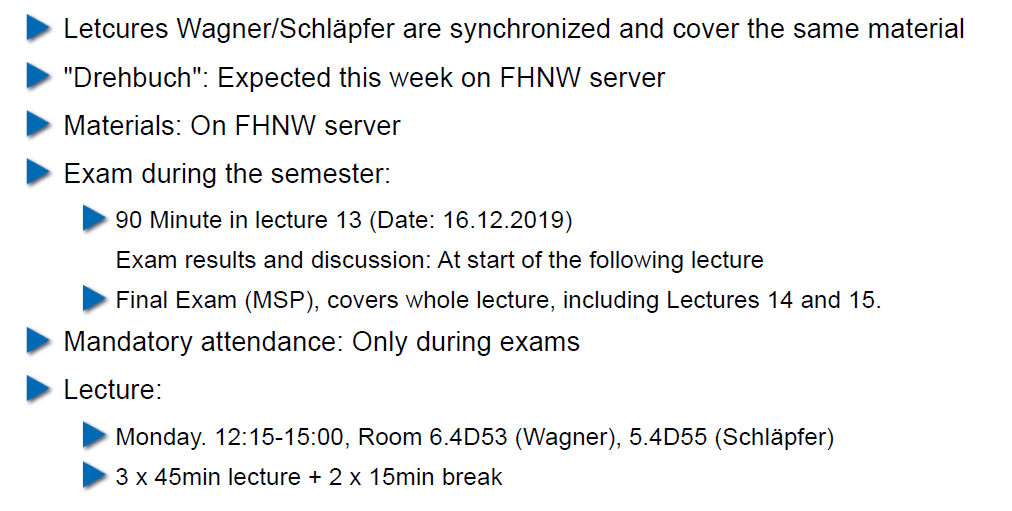
Die Modulnote setzt sich zu 100% aus einer Semesterprüfung und einer Modulschlussprüfung zu je 50% zusammen.

# Woche 1

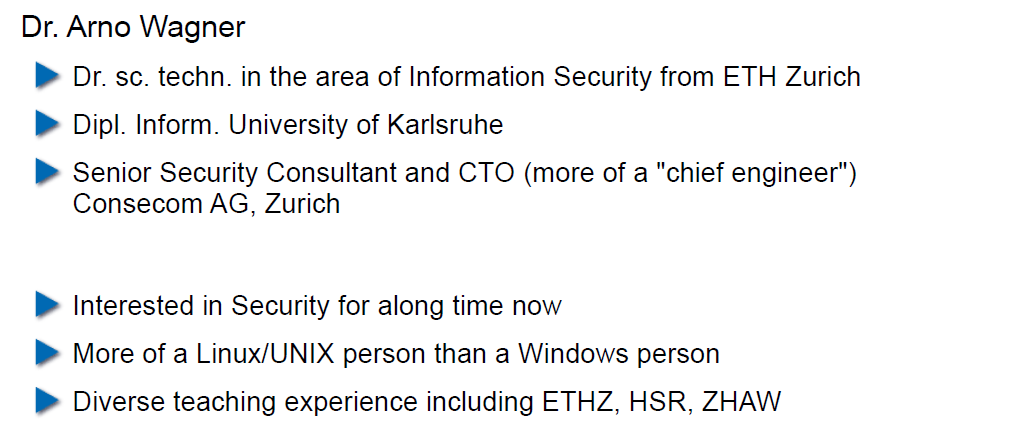
## Agenda



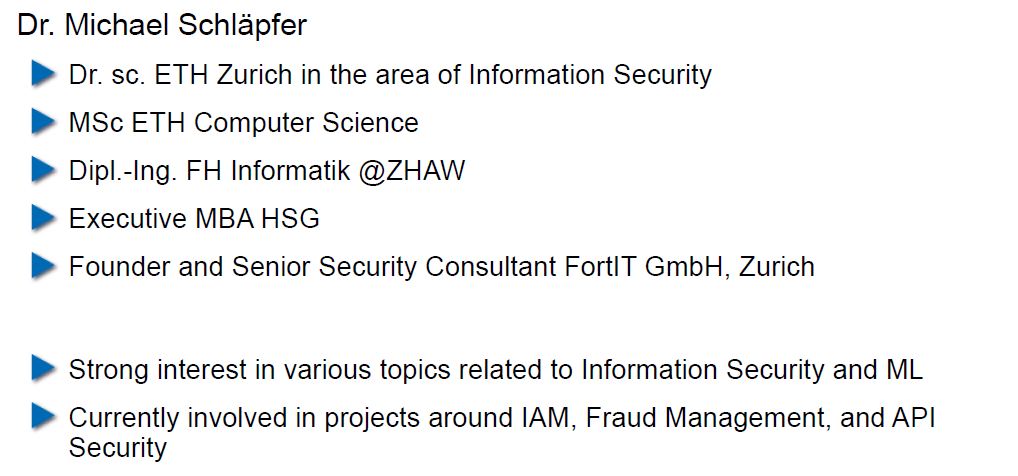
## Administrative Issues



## Who is Your Lecturer (Wagner)



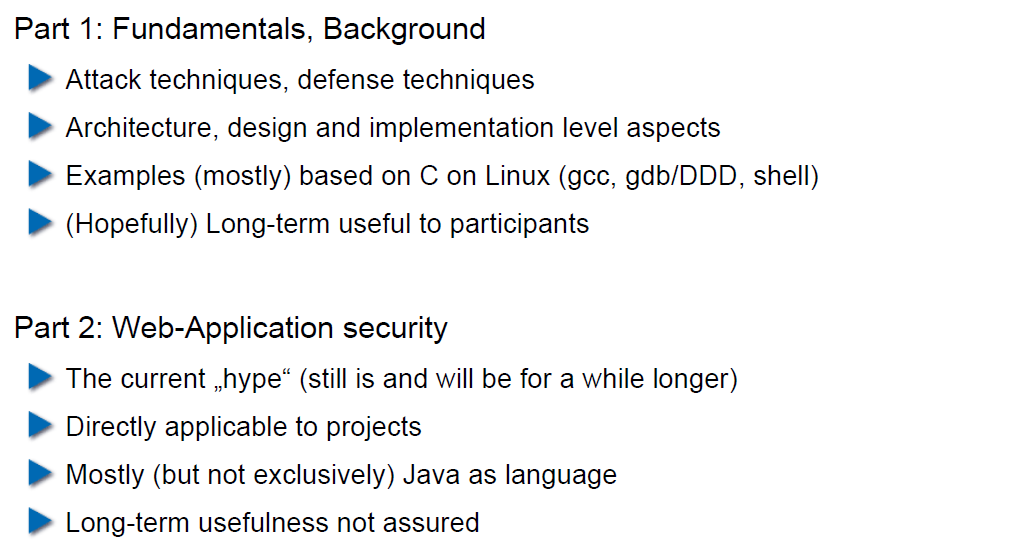
## whoami (Schläpfer)



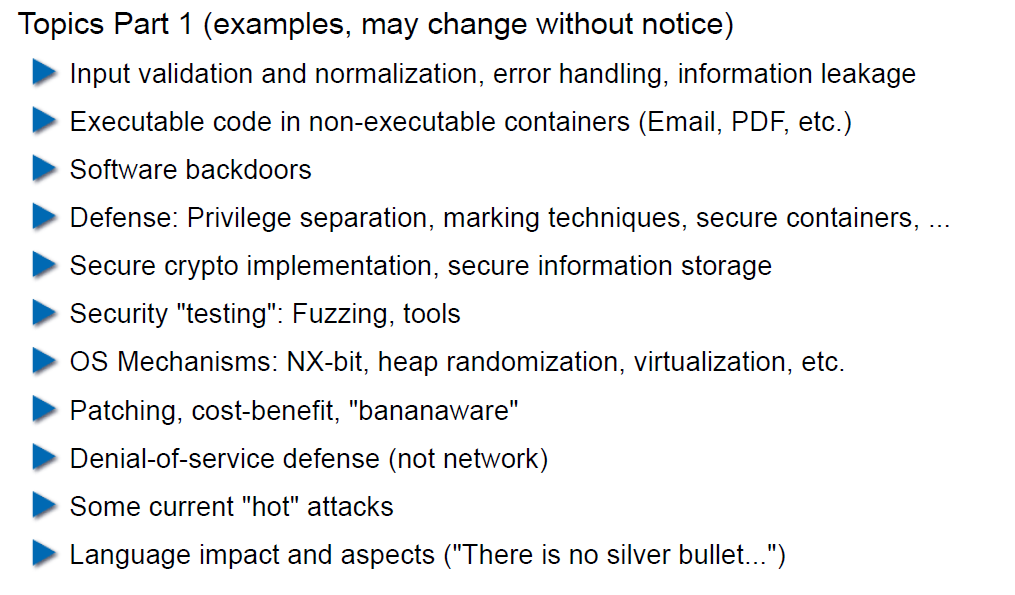
## Why is Application Security Important?



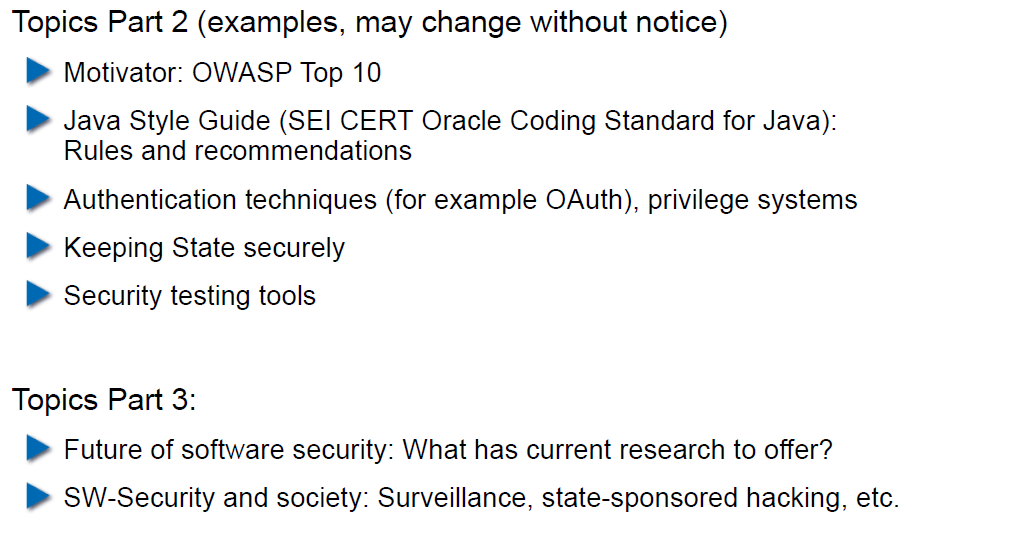
## Lecture Overview (1)



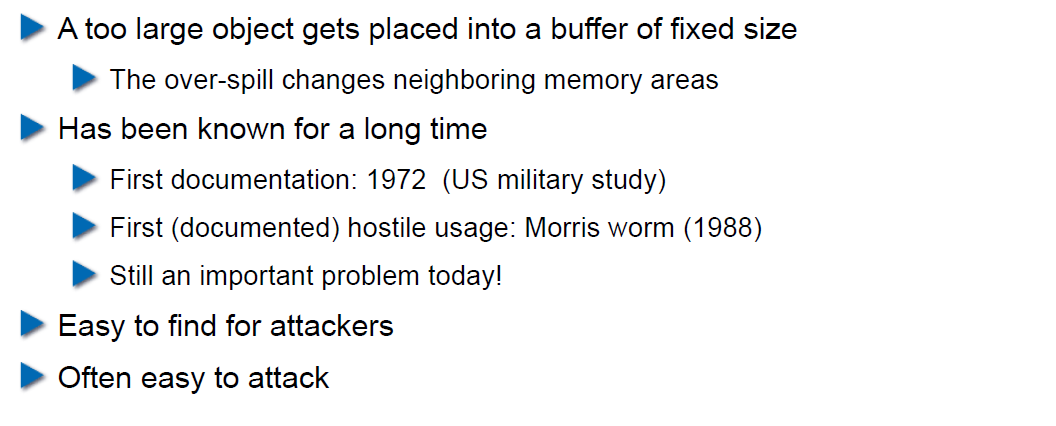
## Lecture Overview (2)



## Lecture Overview (3)



## Example: Buffer Overflow



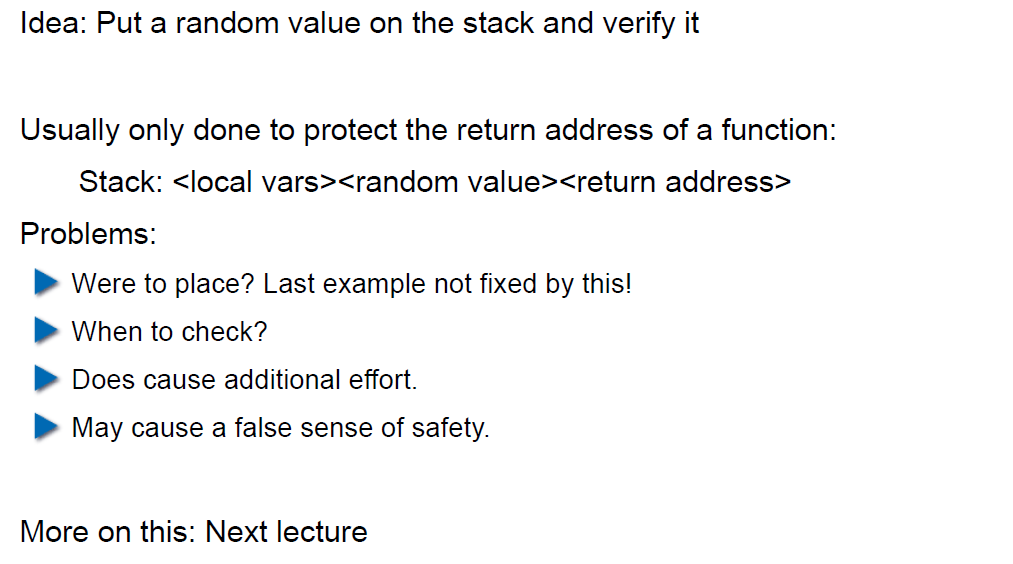
## How does a Buffer Overflow Work?



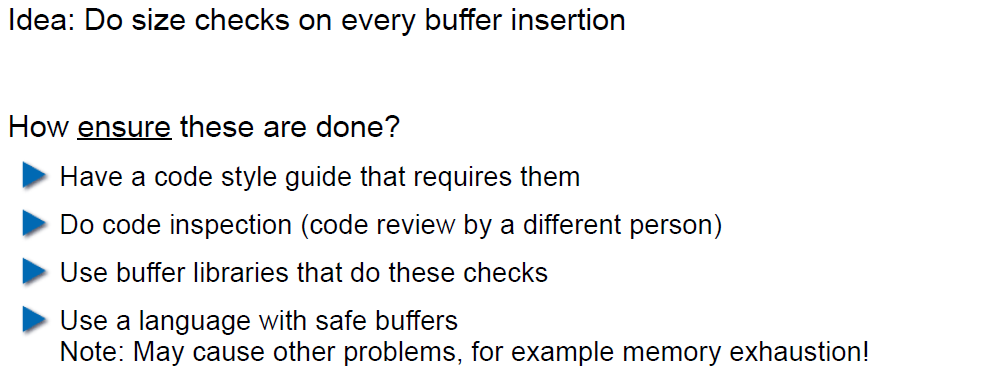
## Buffer Overflow Demo



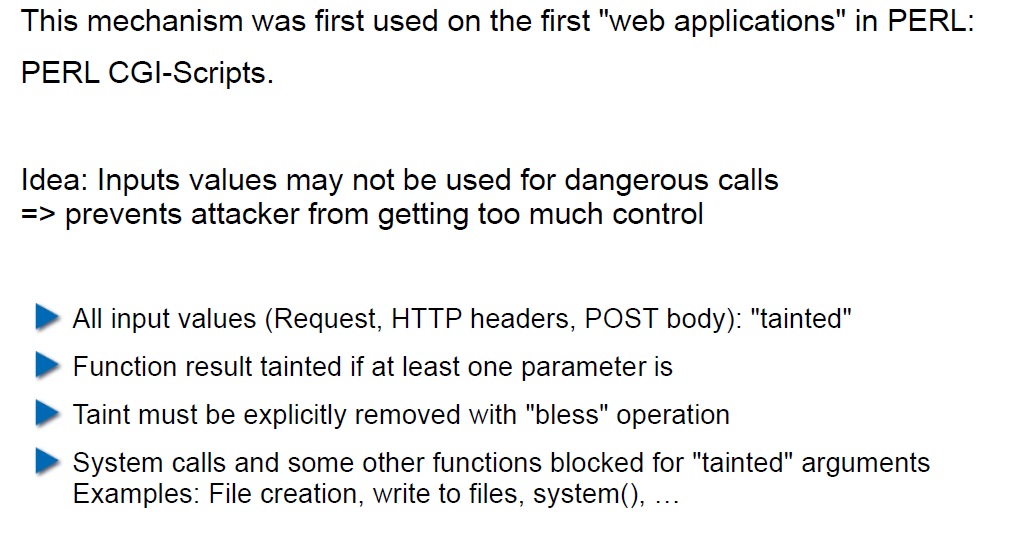
## Not Really a Fix for Buffer Overflow: Stack Protection



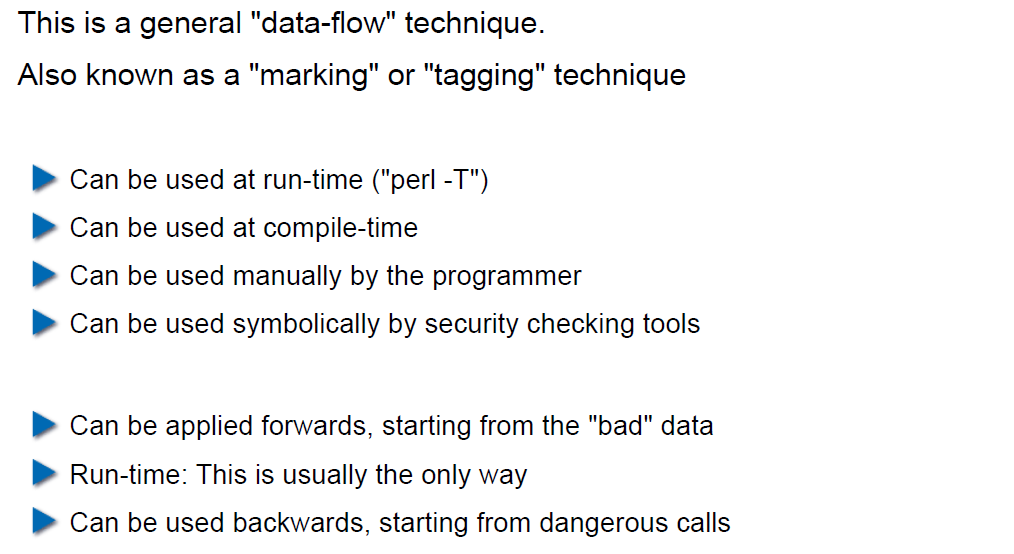
## Fix for Buffer Overflow: Checked Buffers



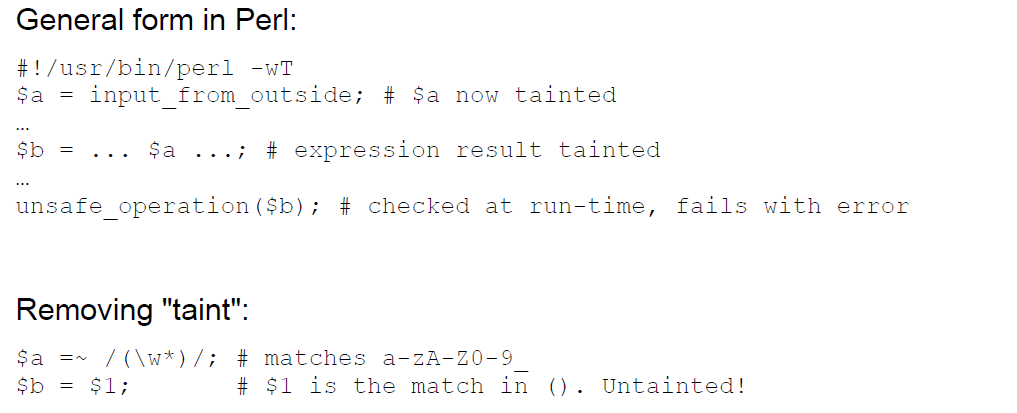
## Example: Taint Checking in Perl (1)



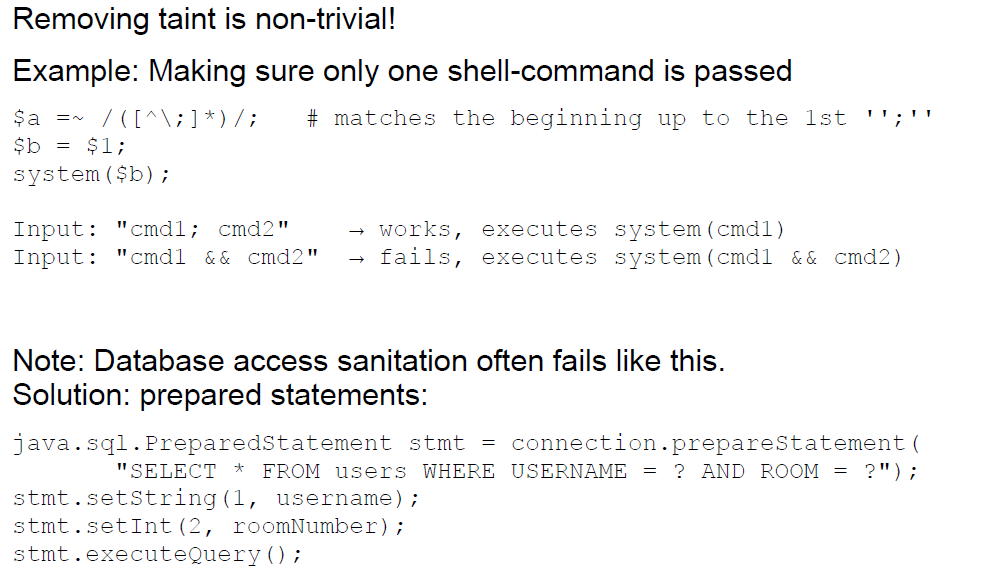
## Example: Taint Checking (2)



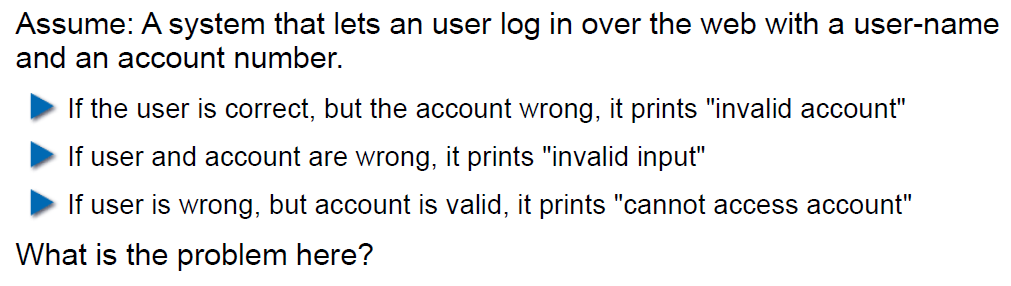
## Example: Taint Checking (3)



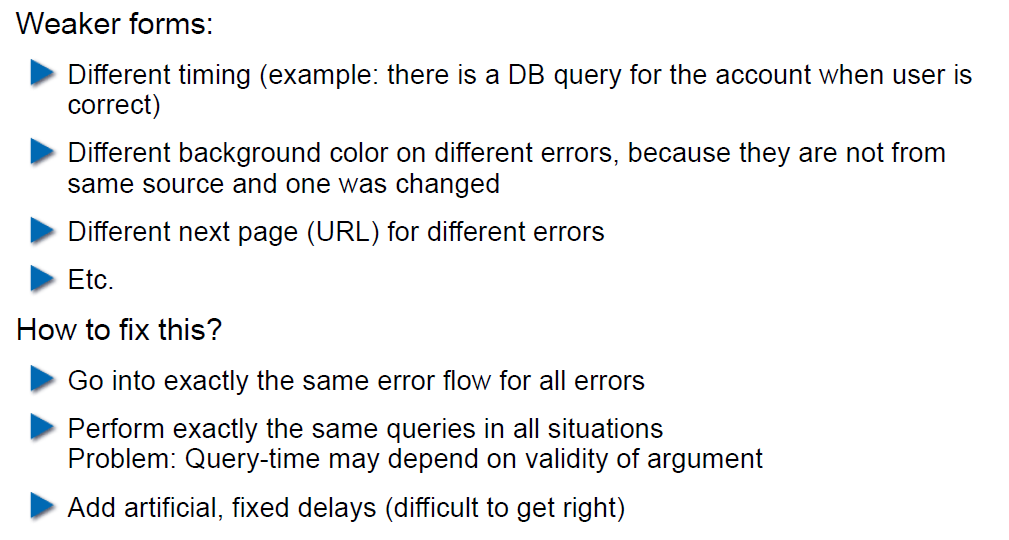
## Example: Taint Checking (4)



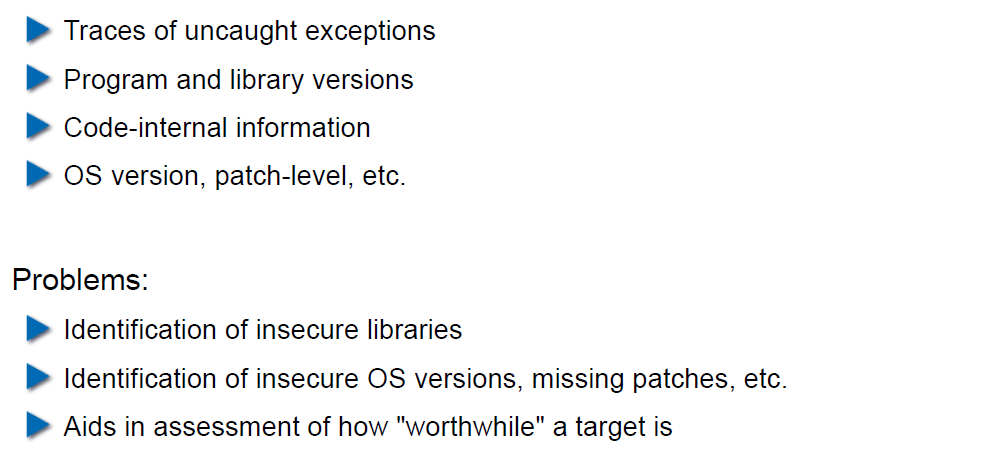
## Example: Data Leakage by Behavior (1)



## Example: Data Leakage by Behavior (2)



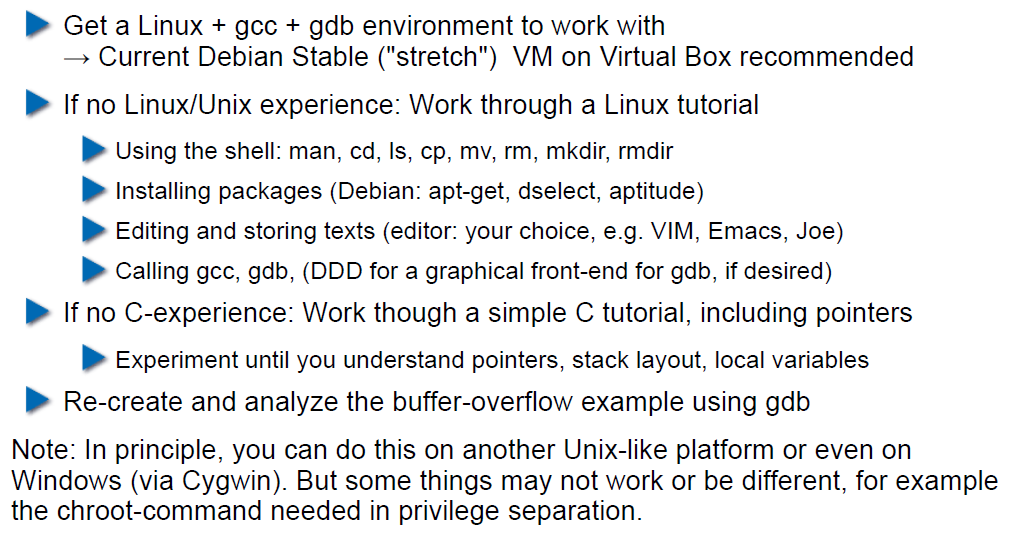
## Example: Data Leakage by Error Output



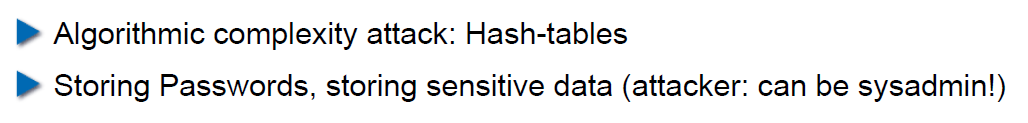
## Human Factors



## Exercises and Preparation for Next Week

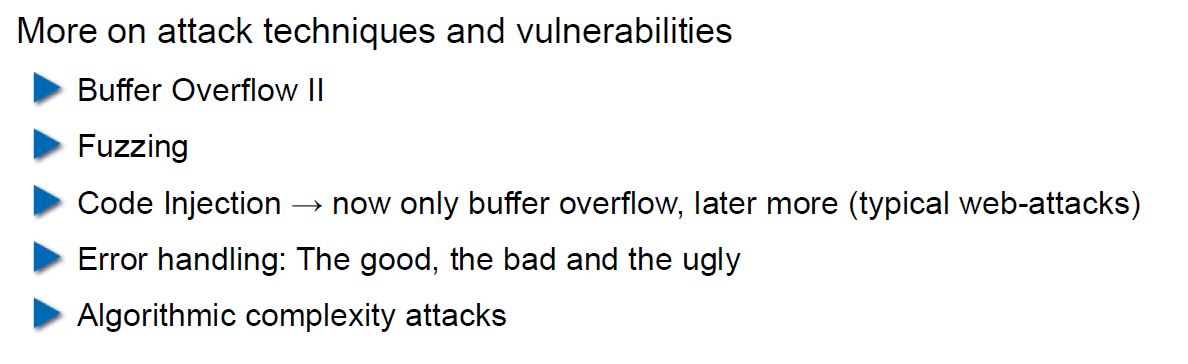


## Reserve

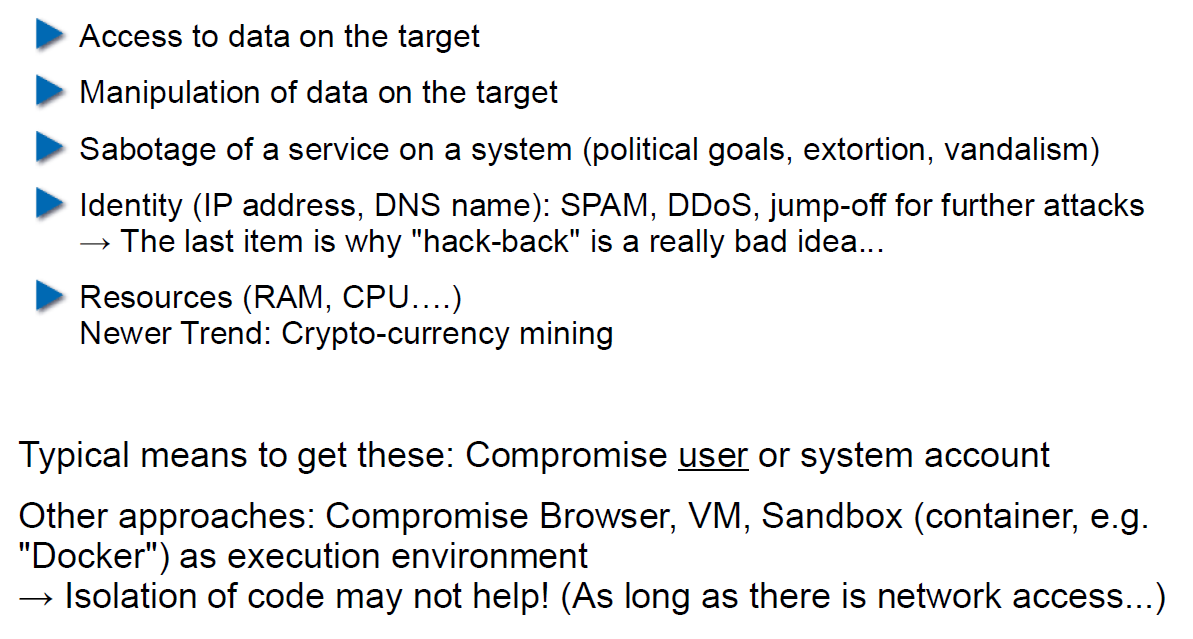


# Woche 2

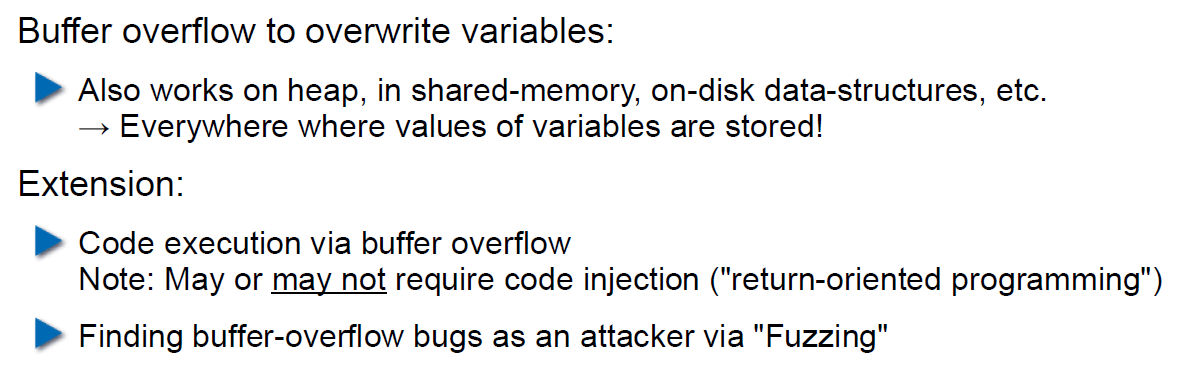
## Agenda



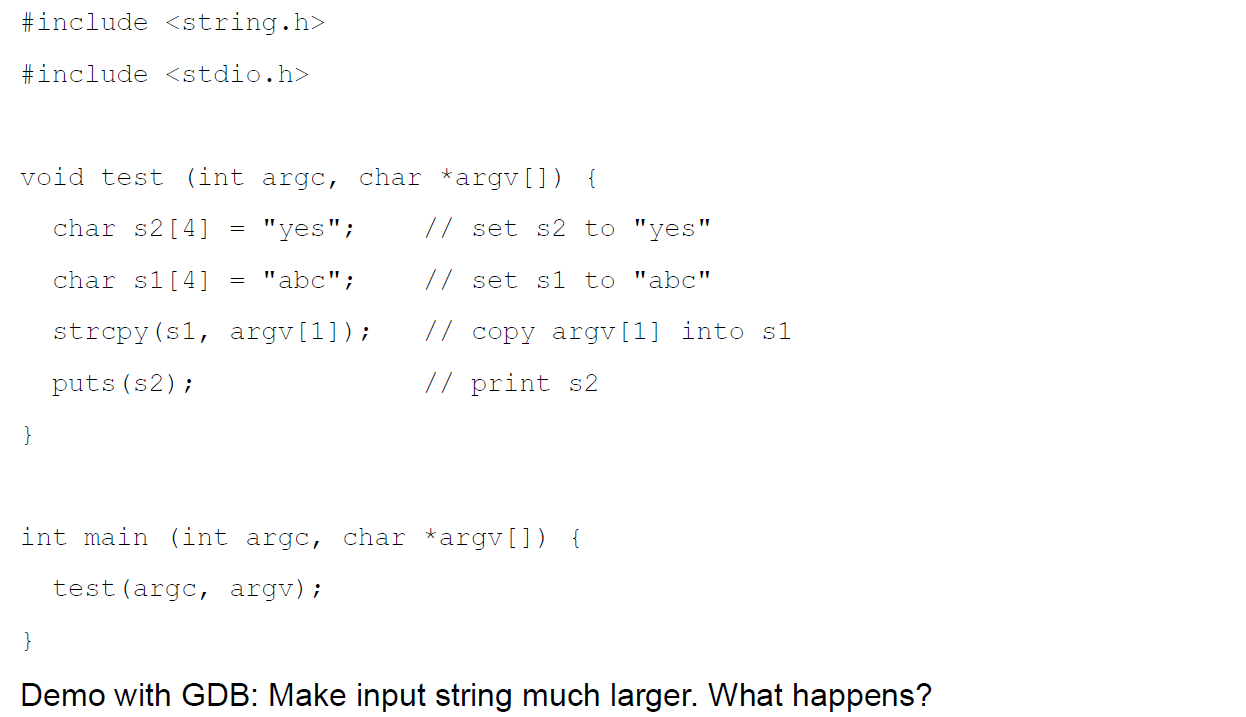
## Recap: What do Attackers Want?



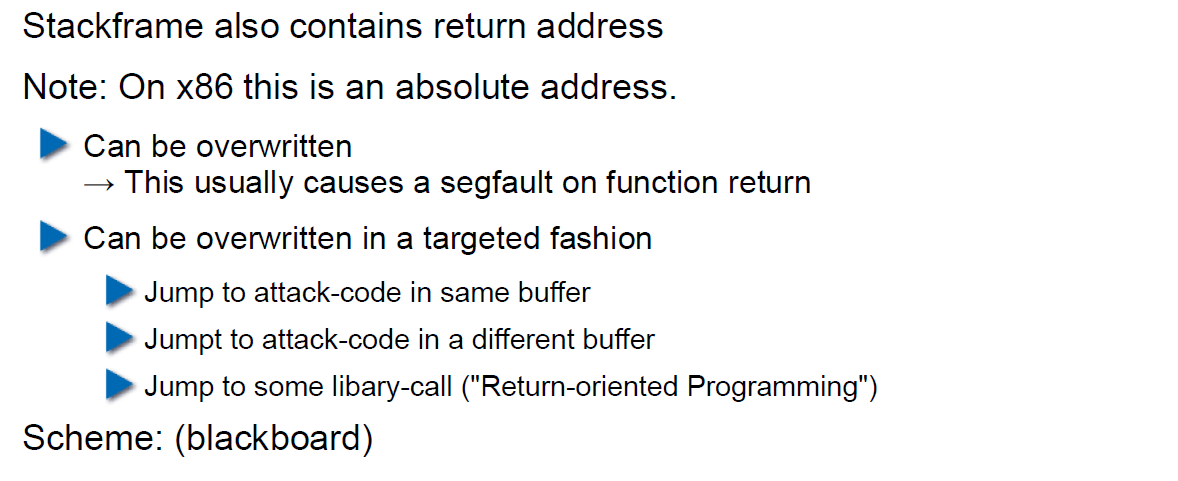
## Buffer Overflow II



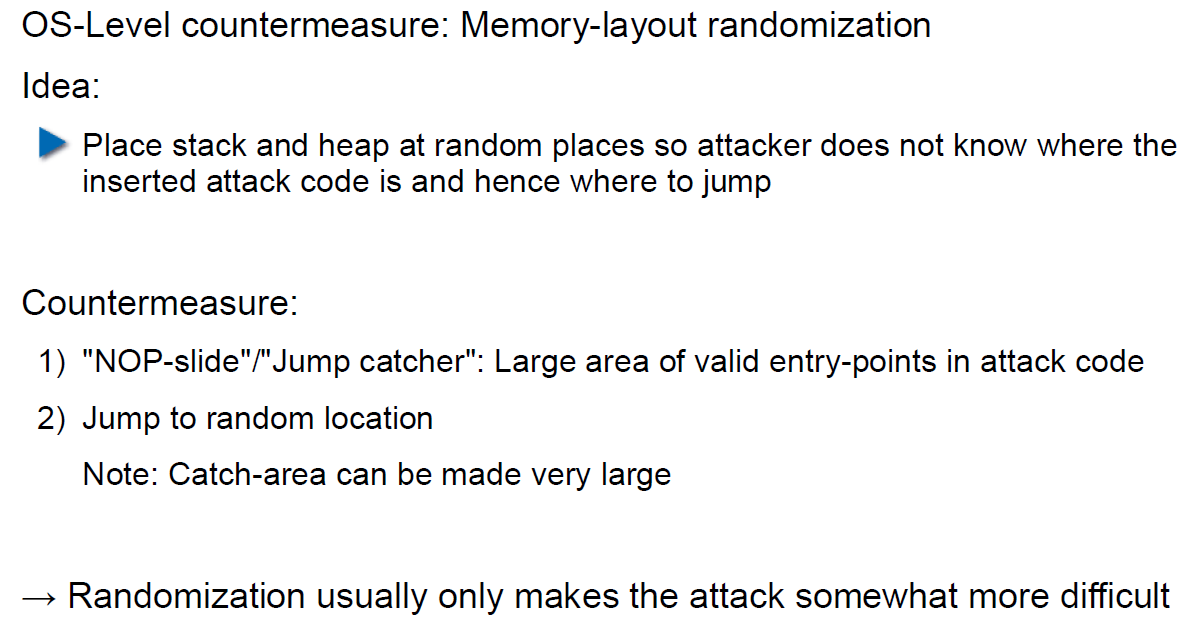
## Buffer Overflow II



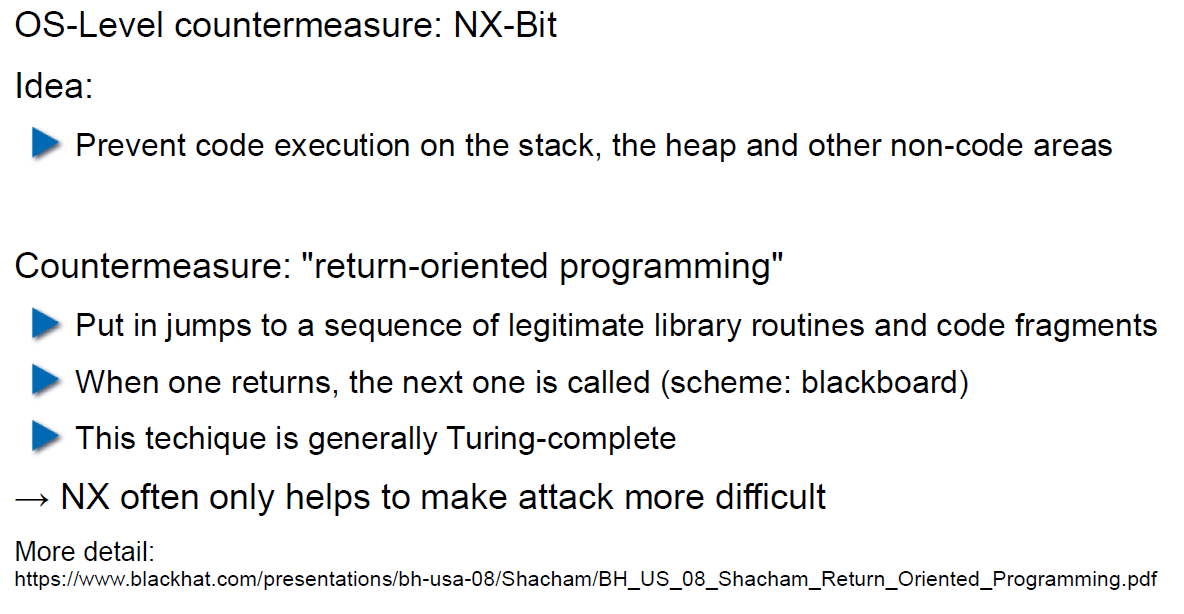
## Buffer Overflow II



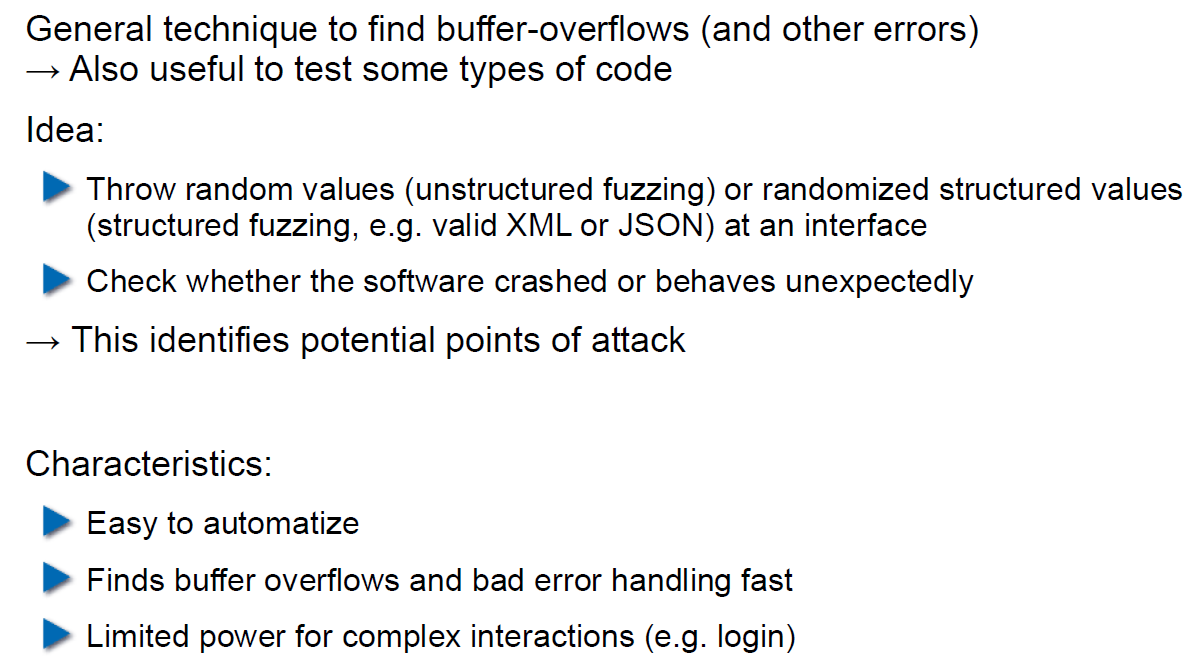
## Buffer Overflow II



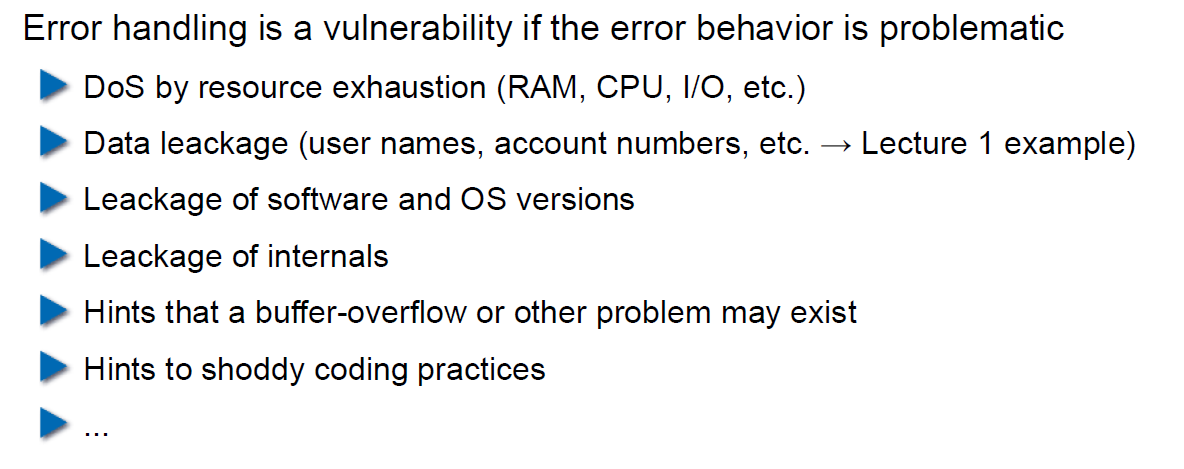
## Buffer Overflow II



## Attack Technique: Fuzzing



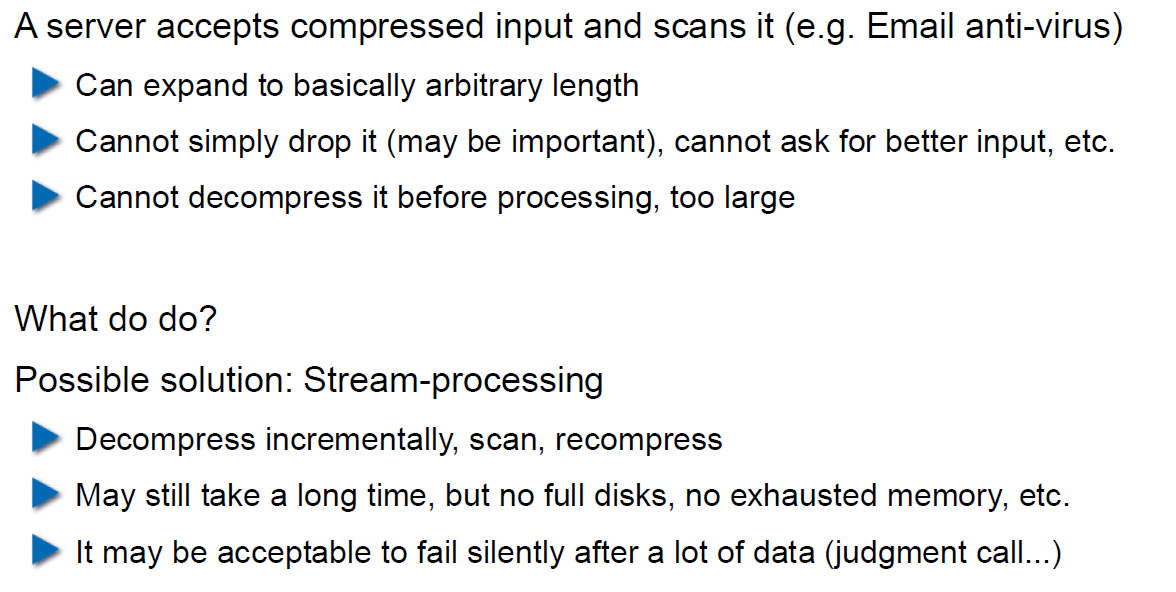
## Error Handling



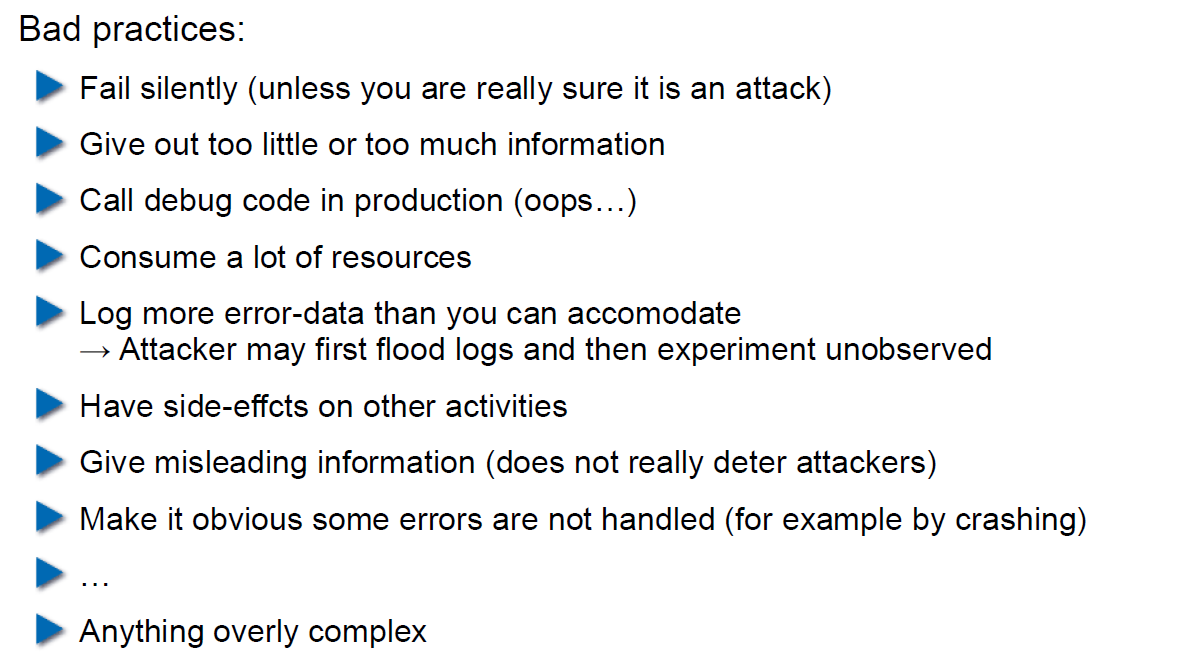
## Case study – Handling too long input



## Case-study: Compression Bomb

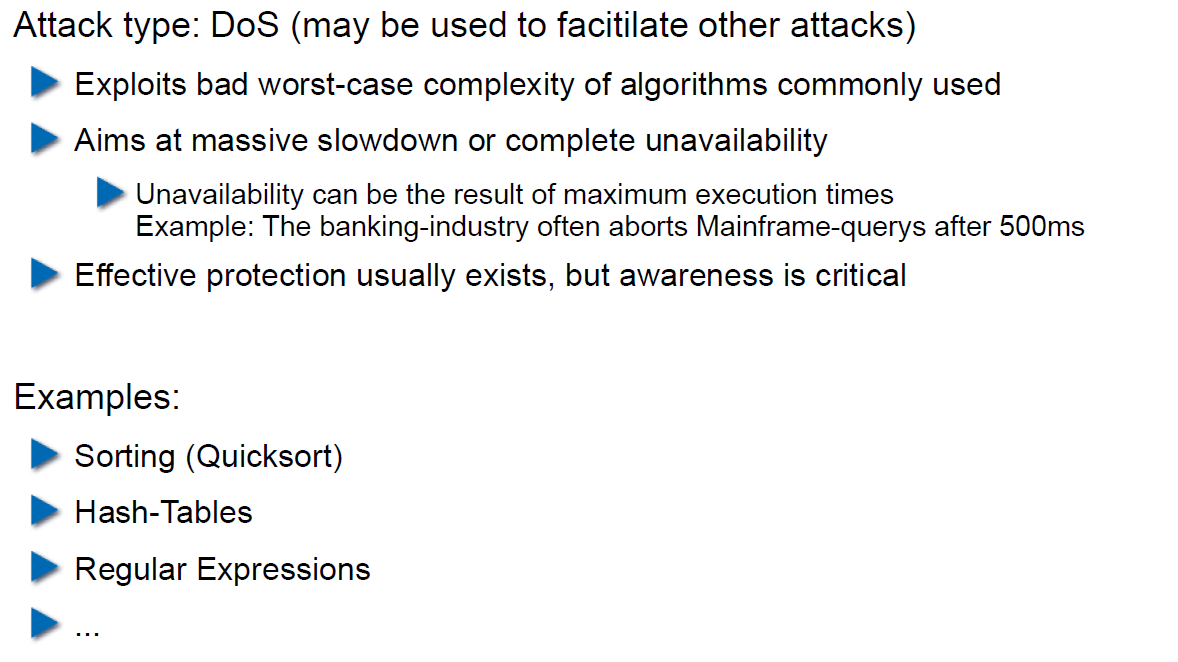


## Error Handling

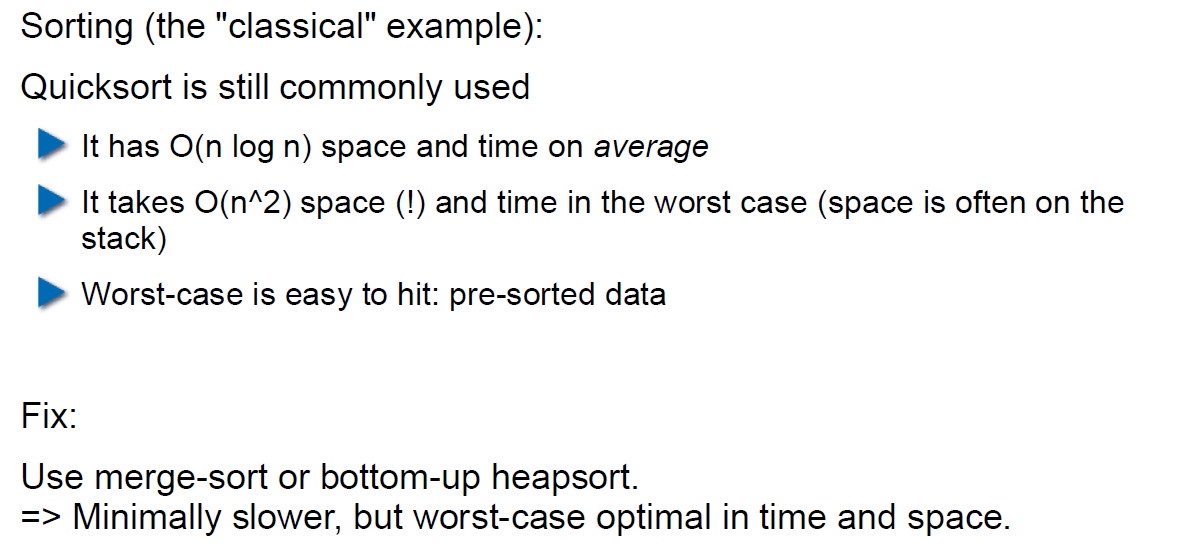




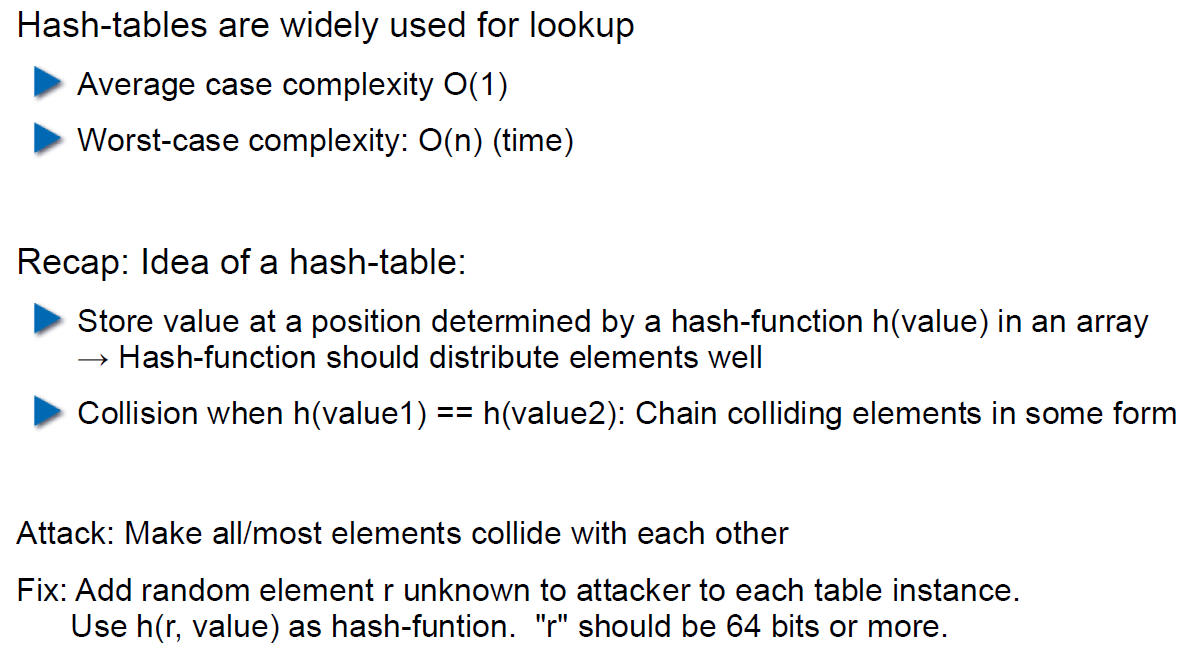
## Algorithmic Complexity Attacks



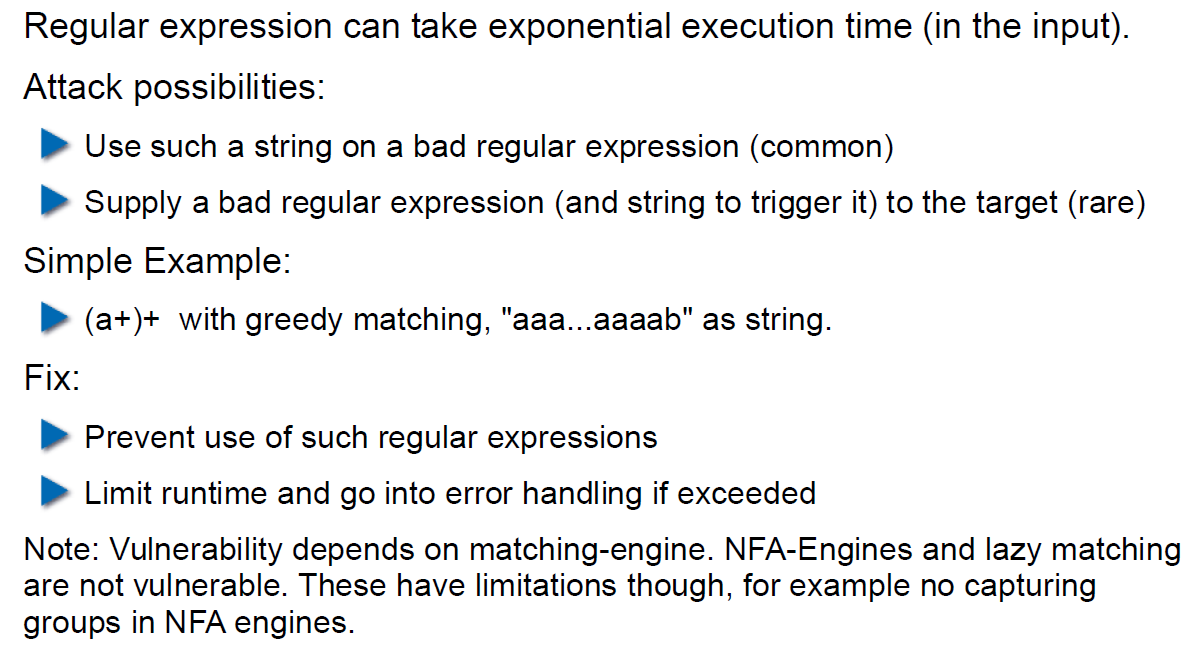
## Algorithmic Complexity: Sorting



## Algorithmic Complexity: Hashing

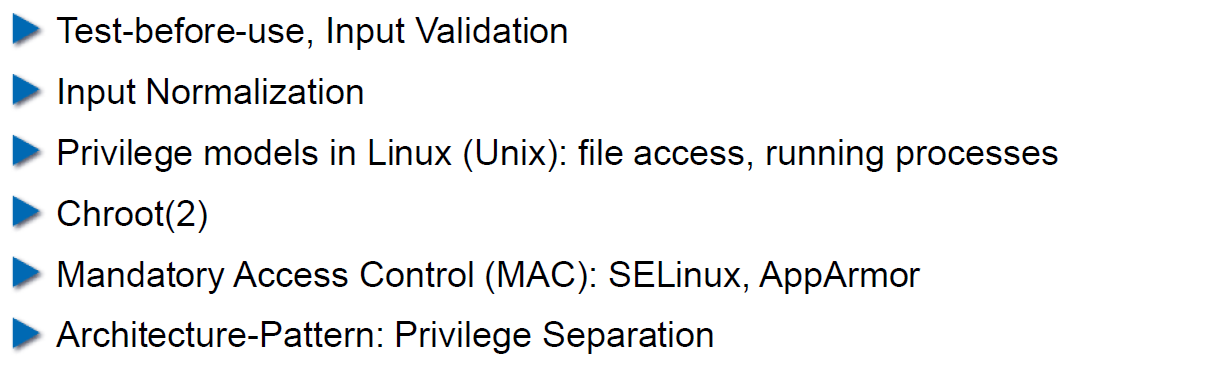


## Algorithmic Complexity: Regular Expressions

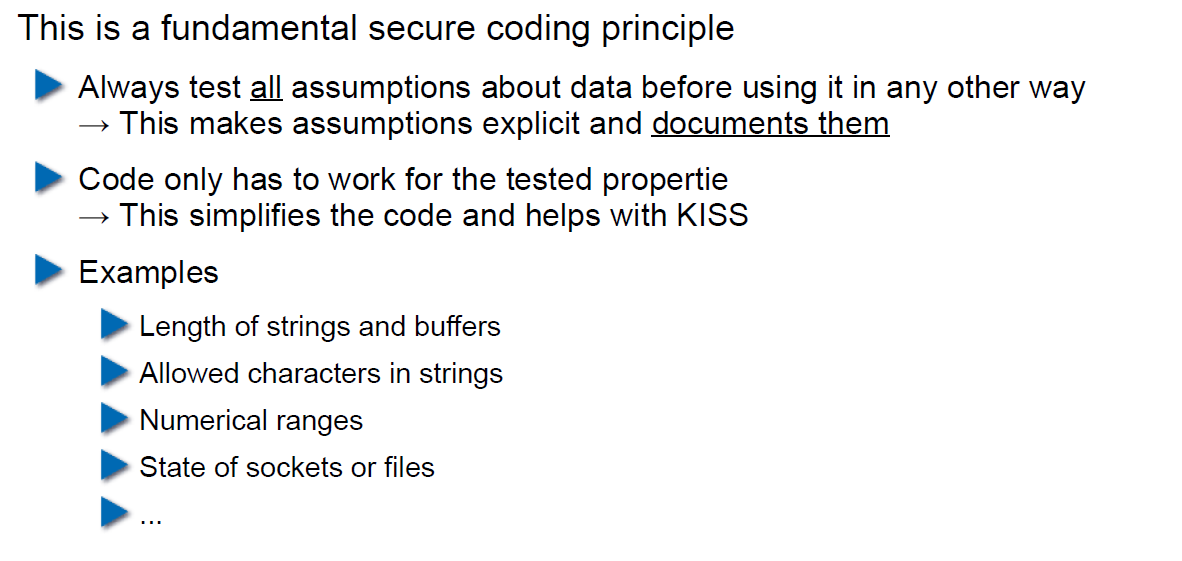


# Woche 3

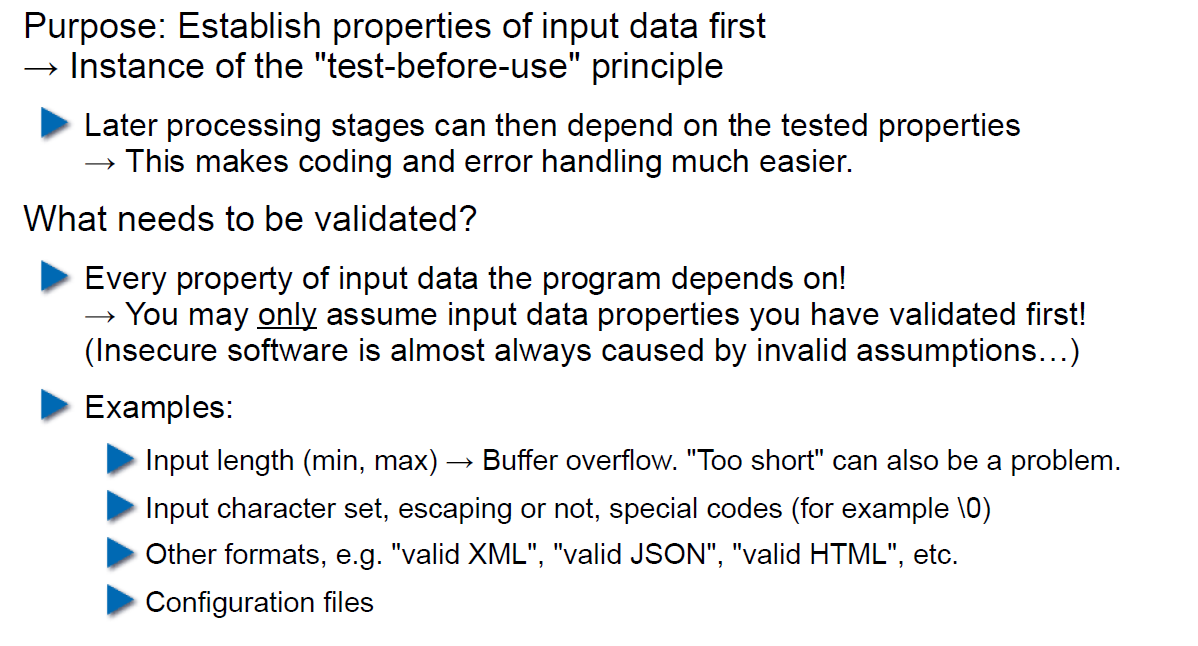
## Agenda



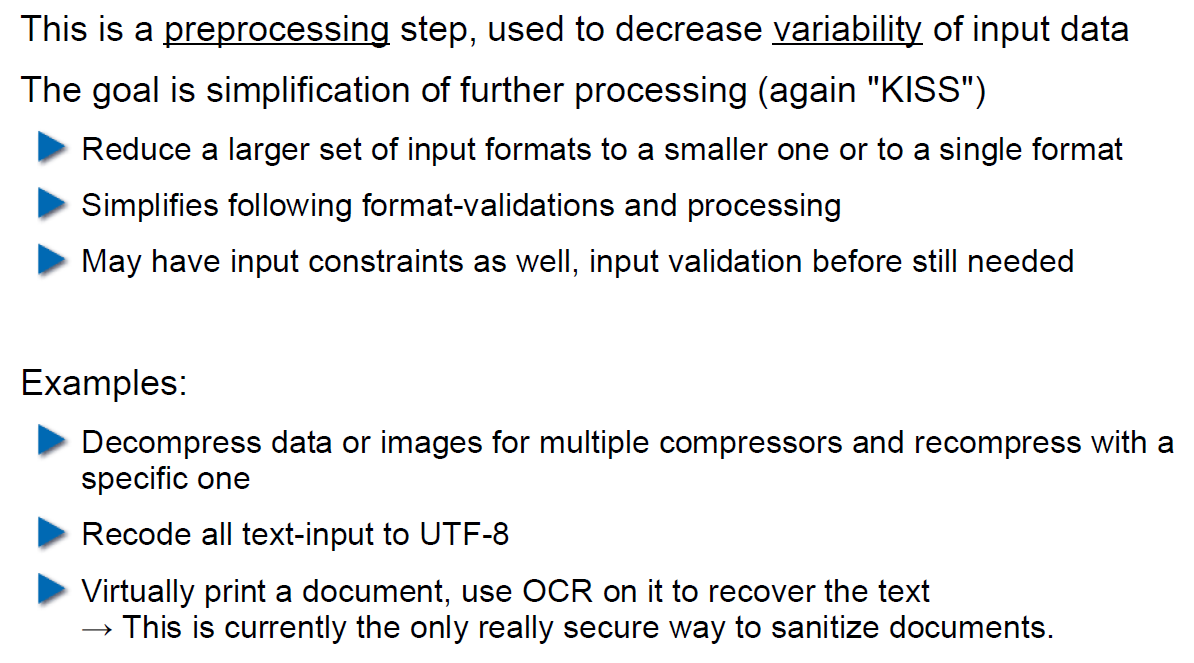
## Secure Coding Principle: Test-before-use



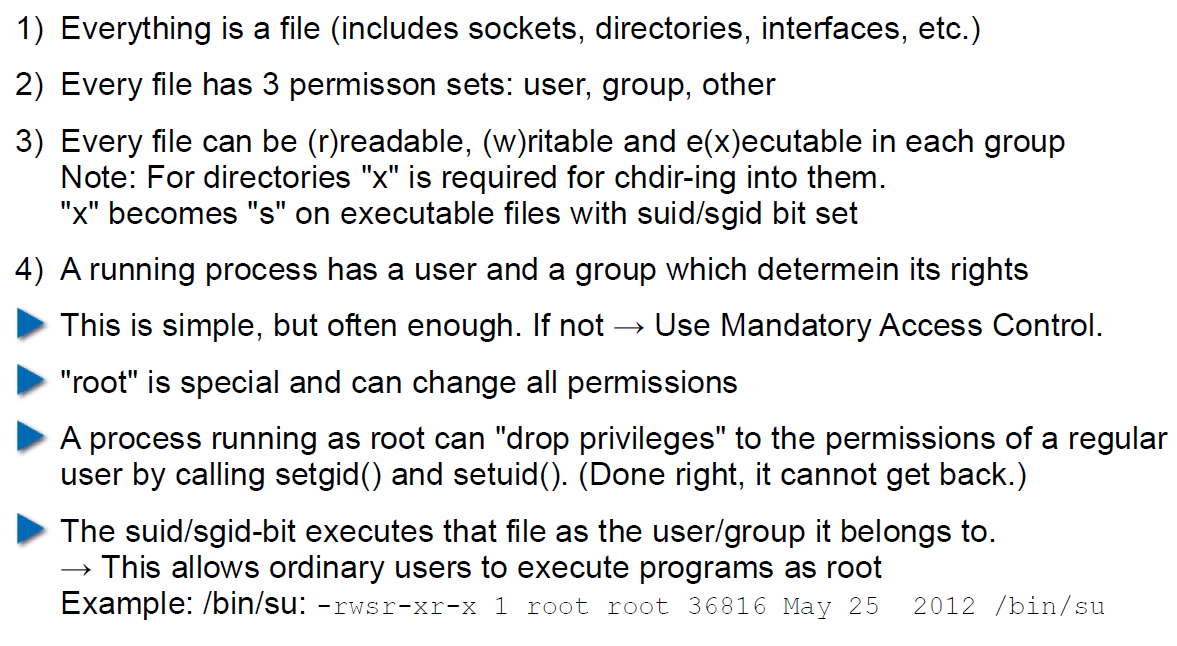
## Input Validation



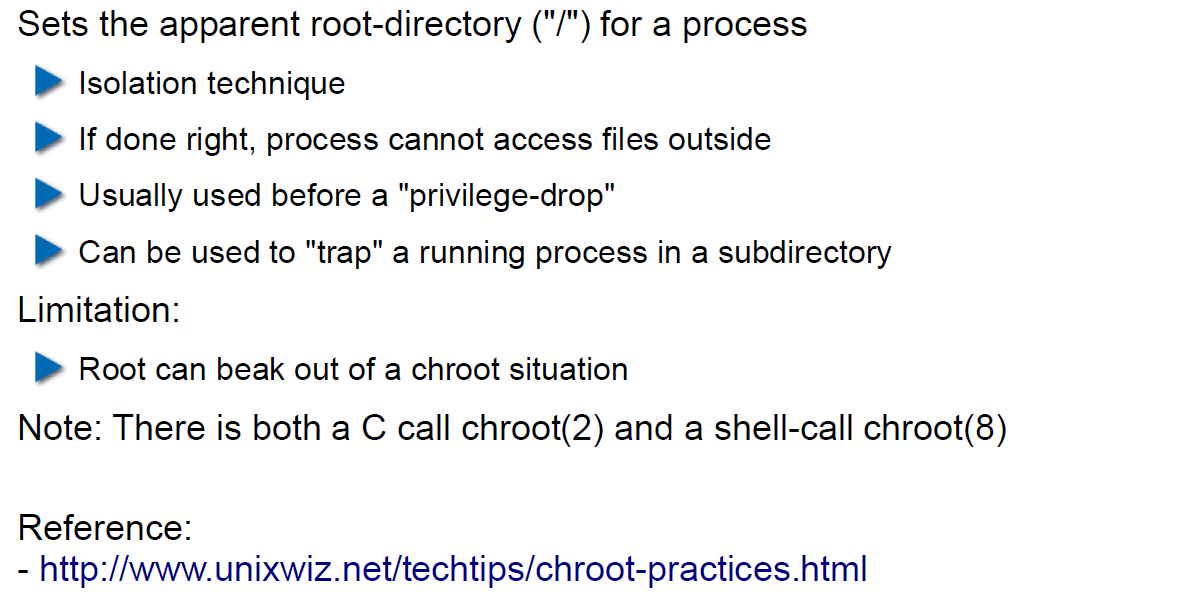
## Input Normalization



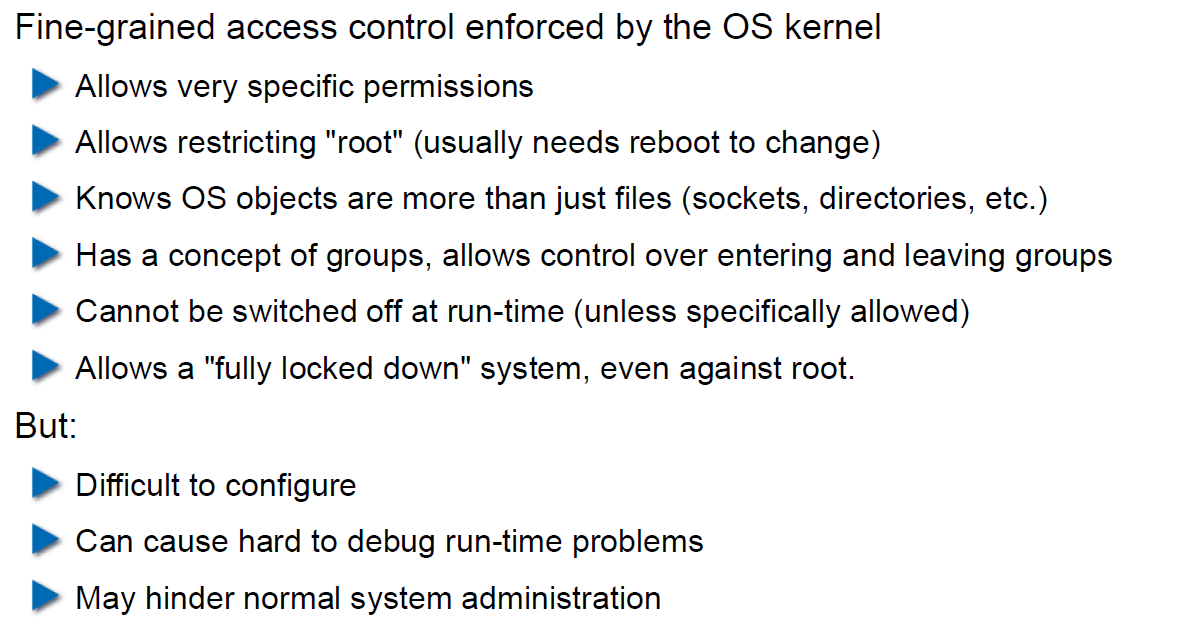
## Unix Permission Model (simplified)



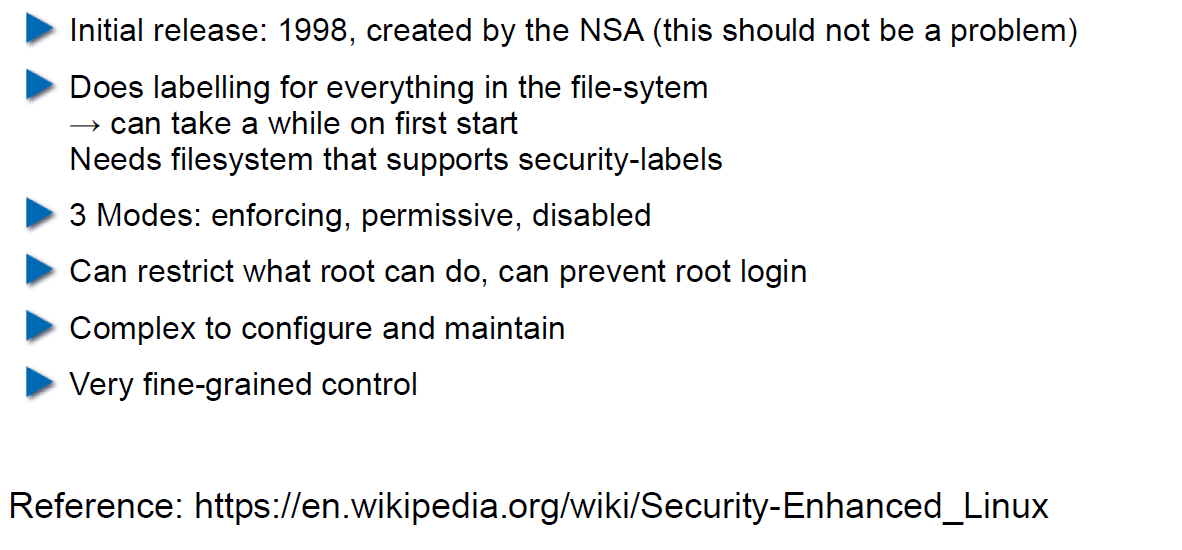
## Chroot



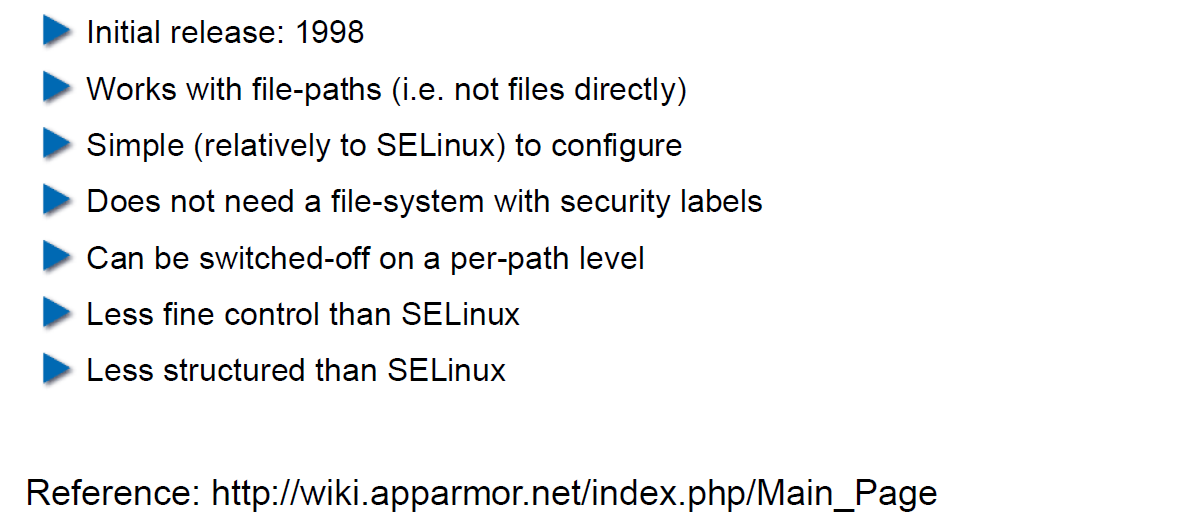
## Mandatory Access Control (MAC)



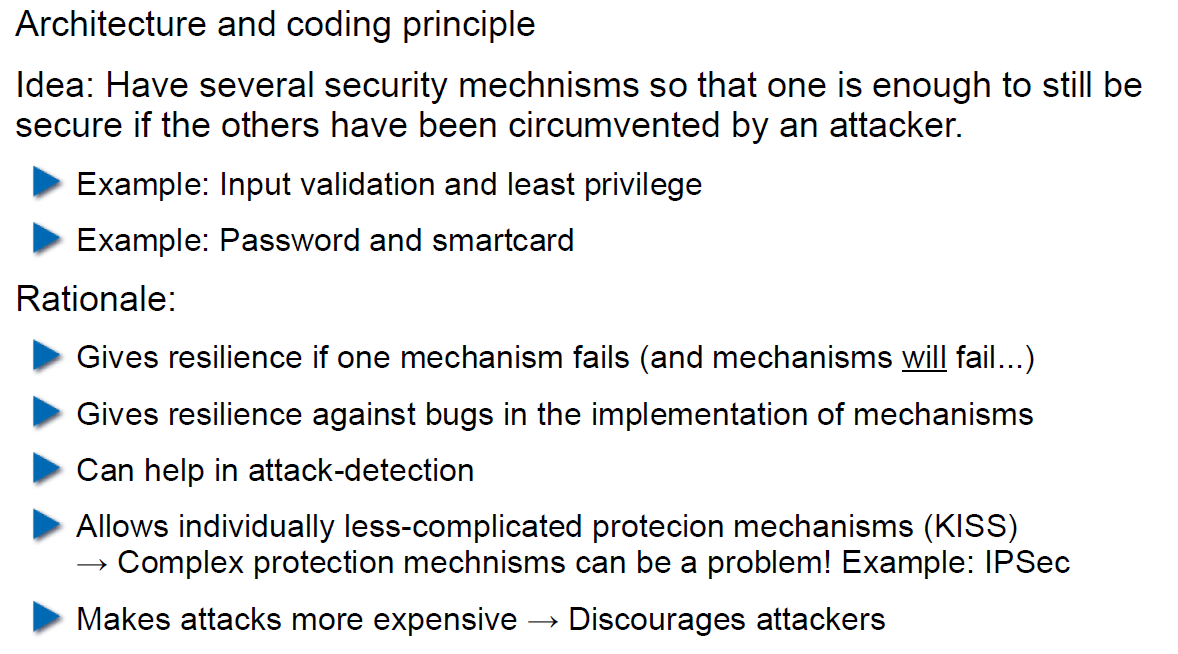
## MAC Example: SELinux



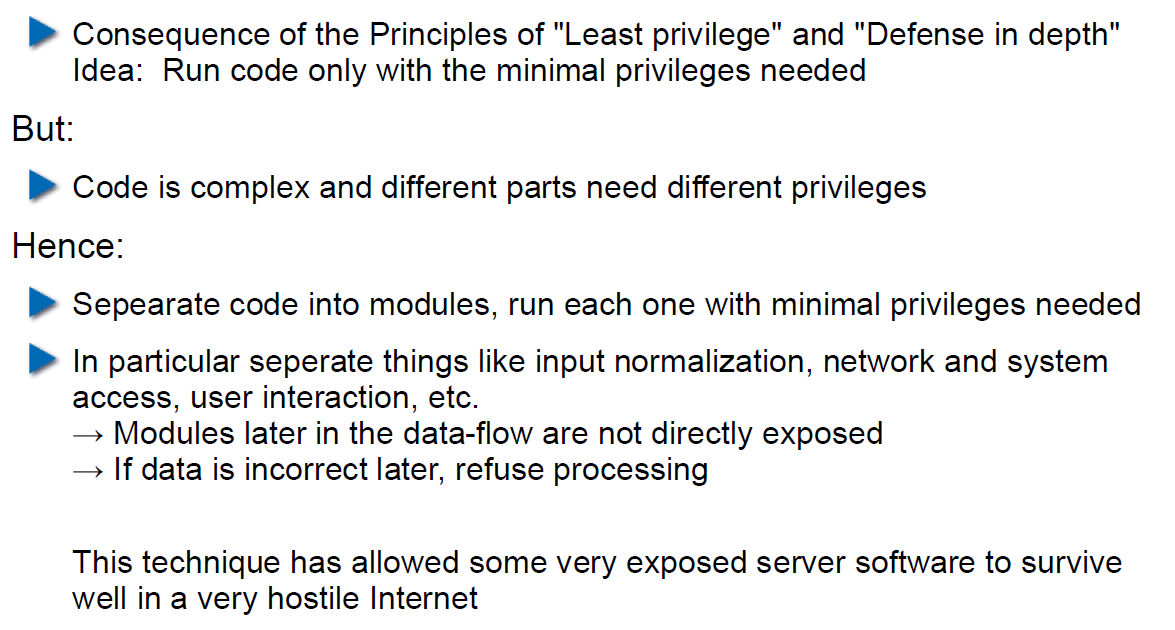
## MAC Example: AppArmor



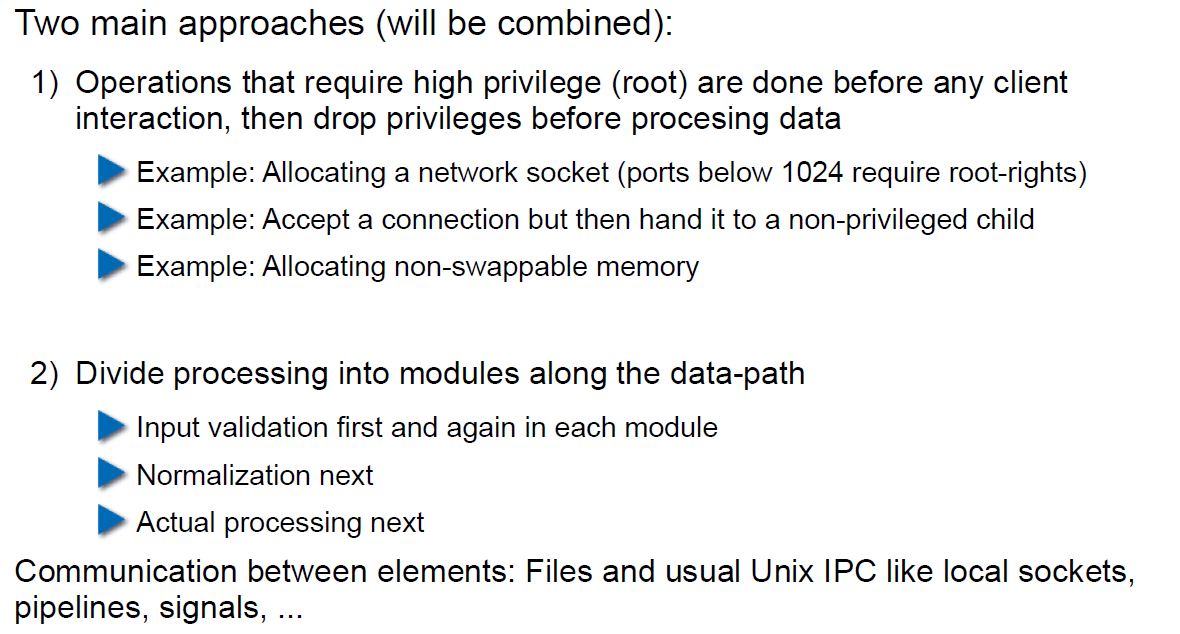
## Secure Coding Principle: Defense in Depth



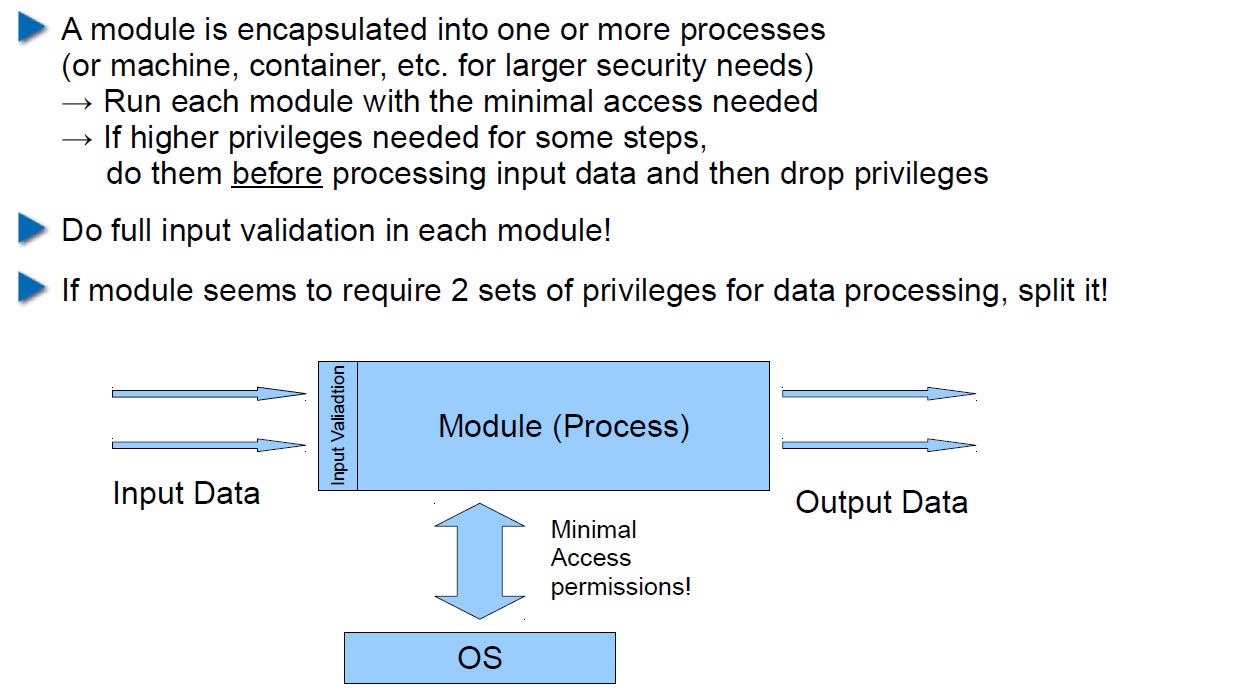
## Privilege Separation



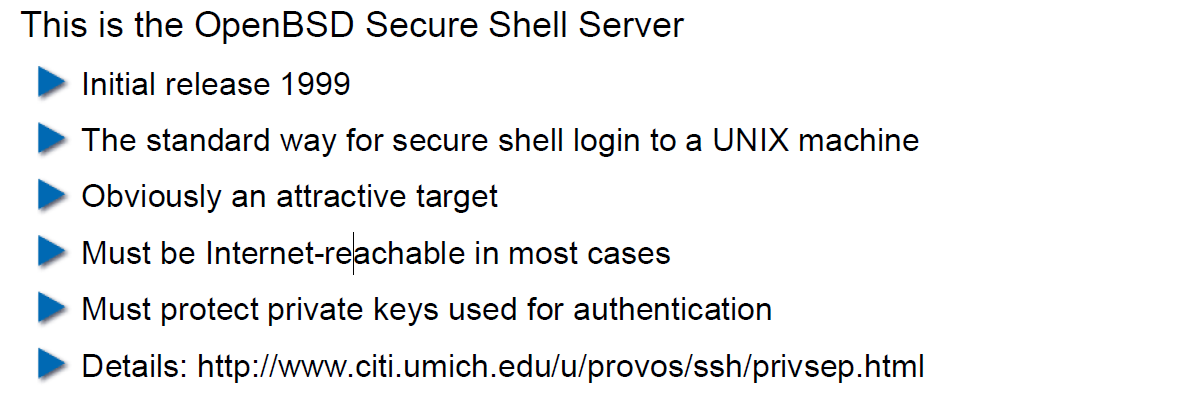
## Overall Idea



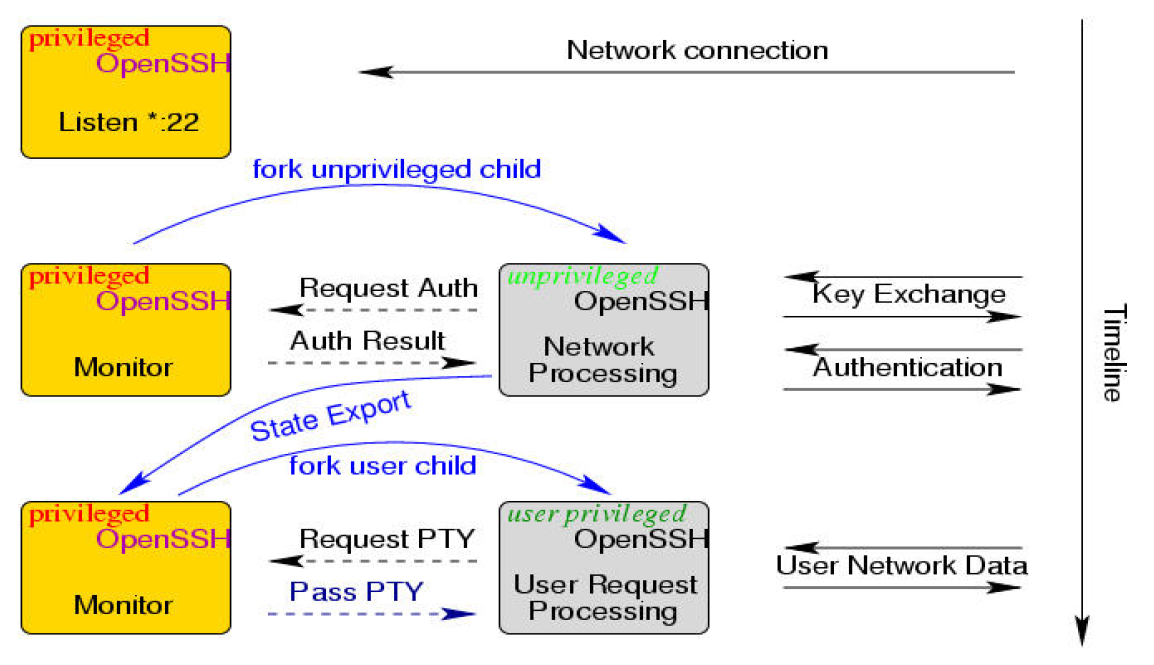
## Form of the Modules



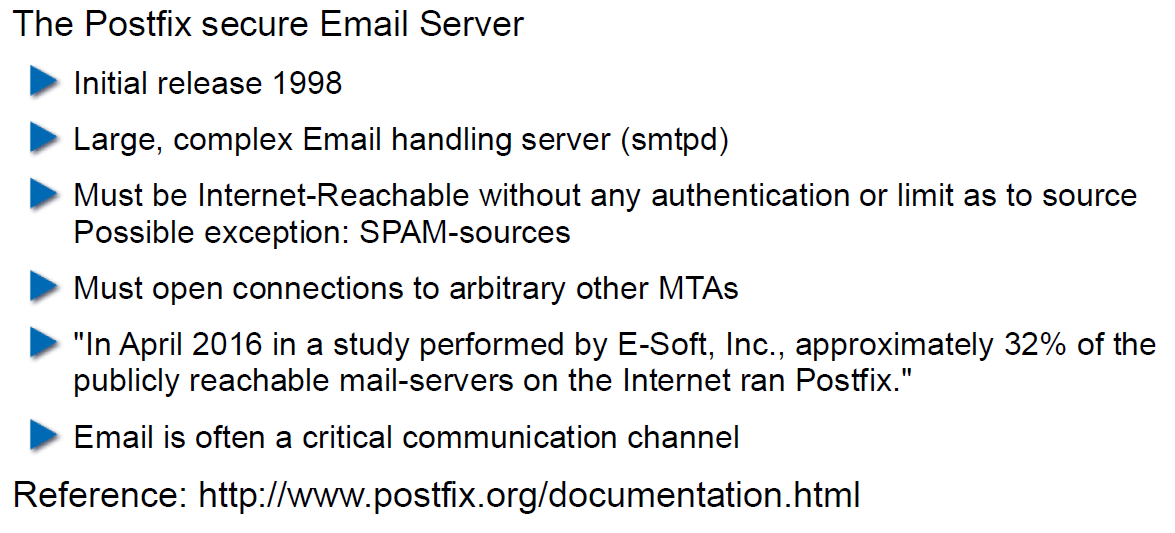
## Example: OpenSSH



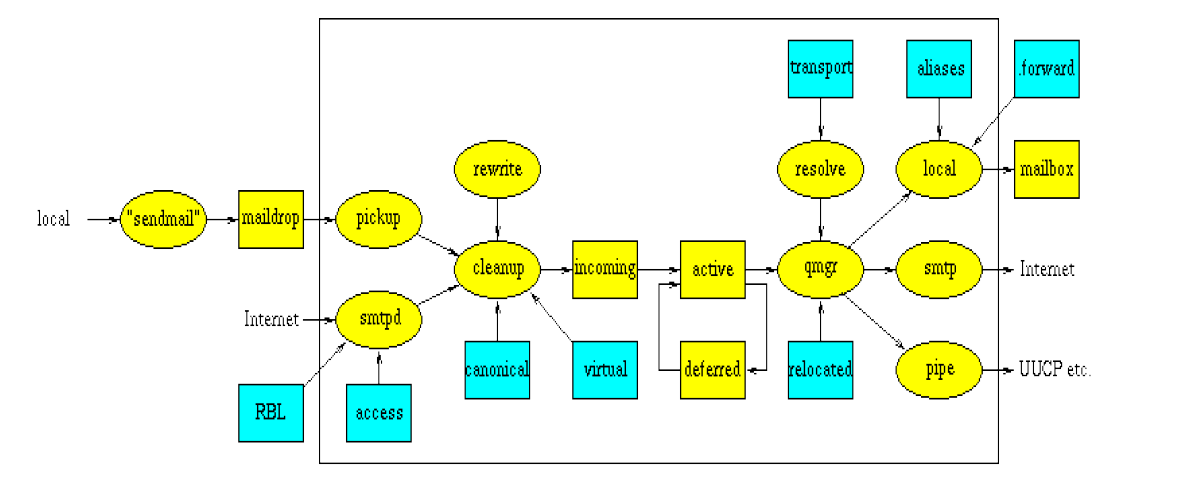
## OpenSSH: Privilege Separation Approach



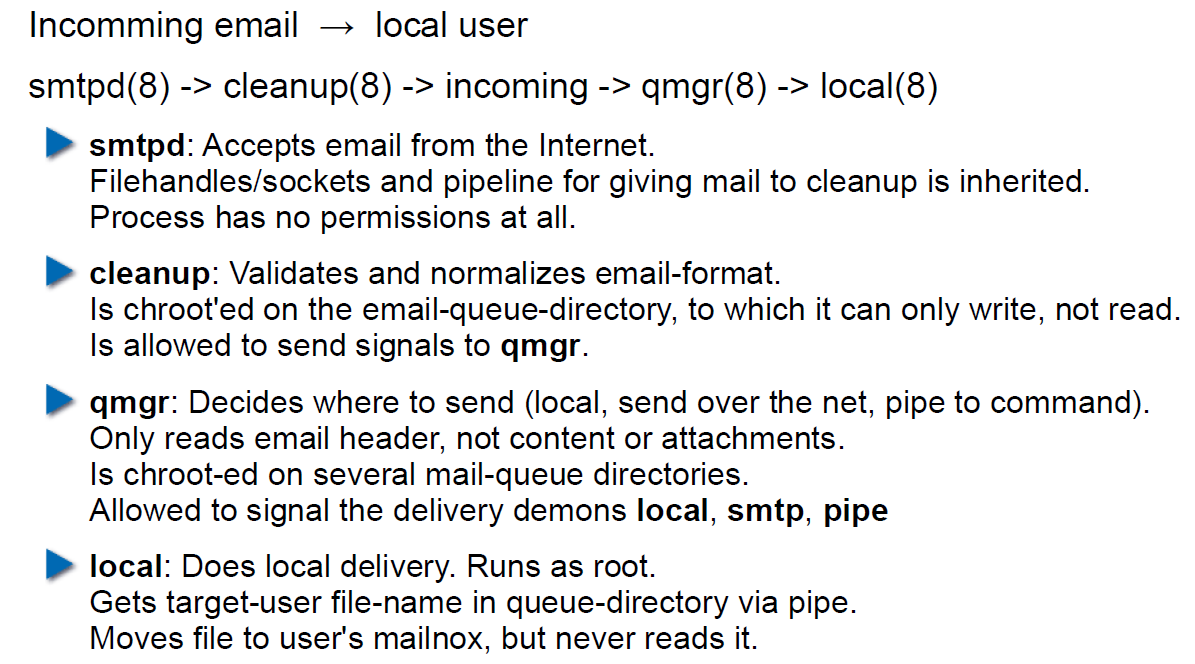
## Example: Postfix MTA (Email Server)



## Postfix System View

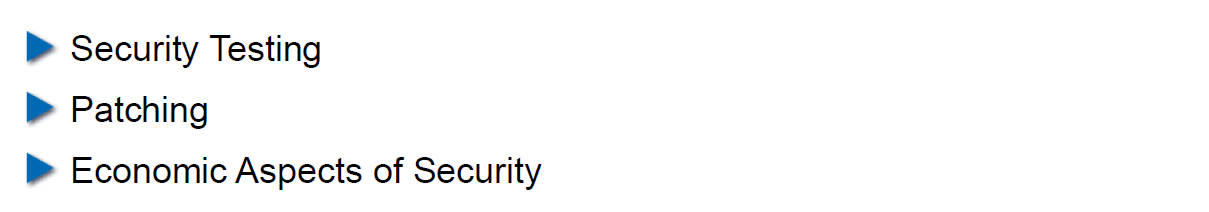


## Postfix: Example Data-Flow

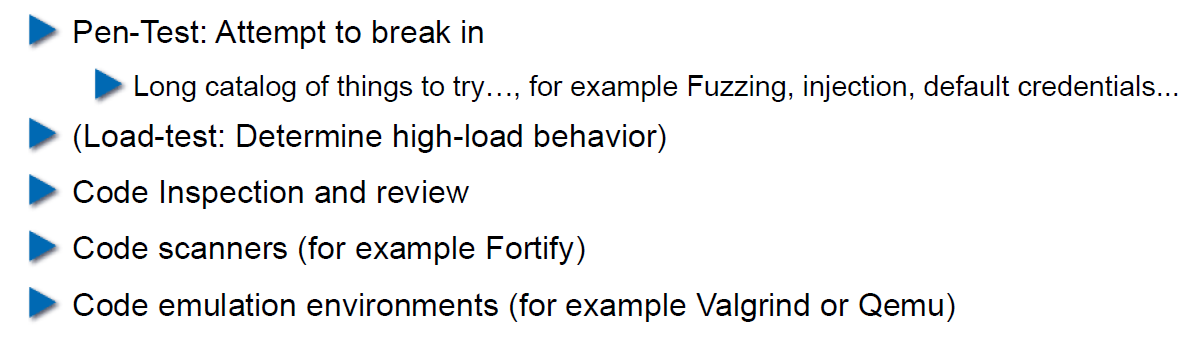


# Woche 4

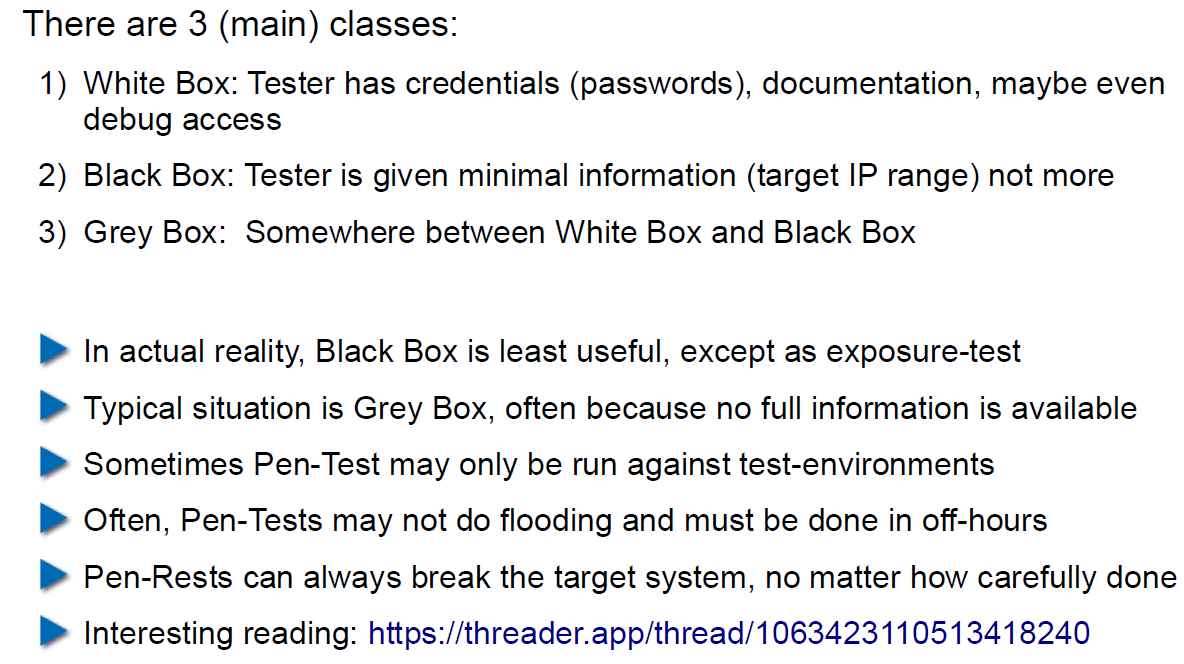
## Agenda



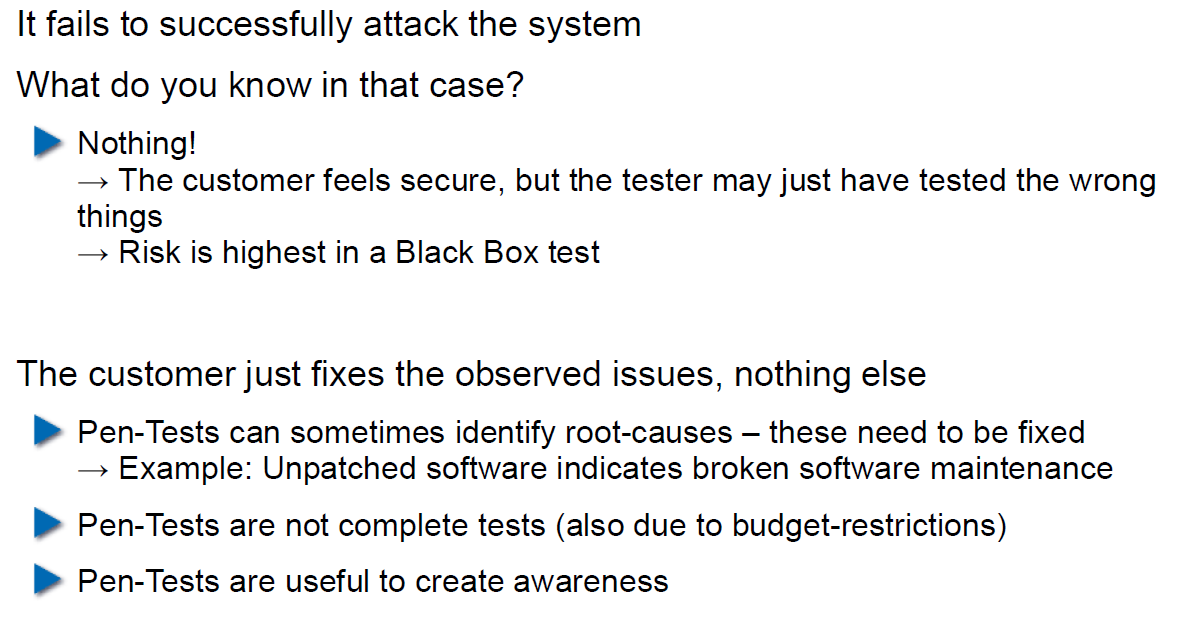
## Security Testing (for Software)



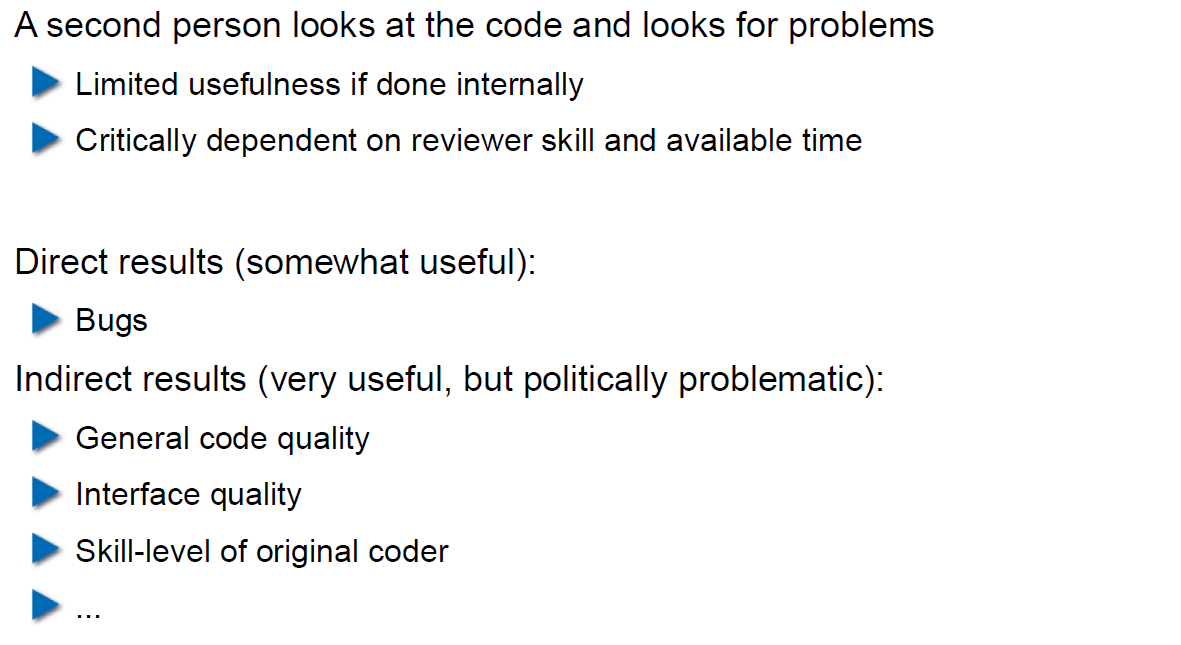
## Pen-Test



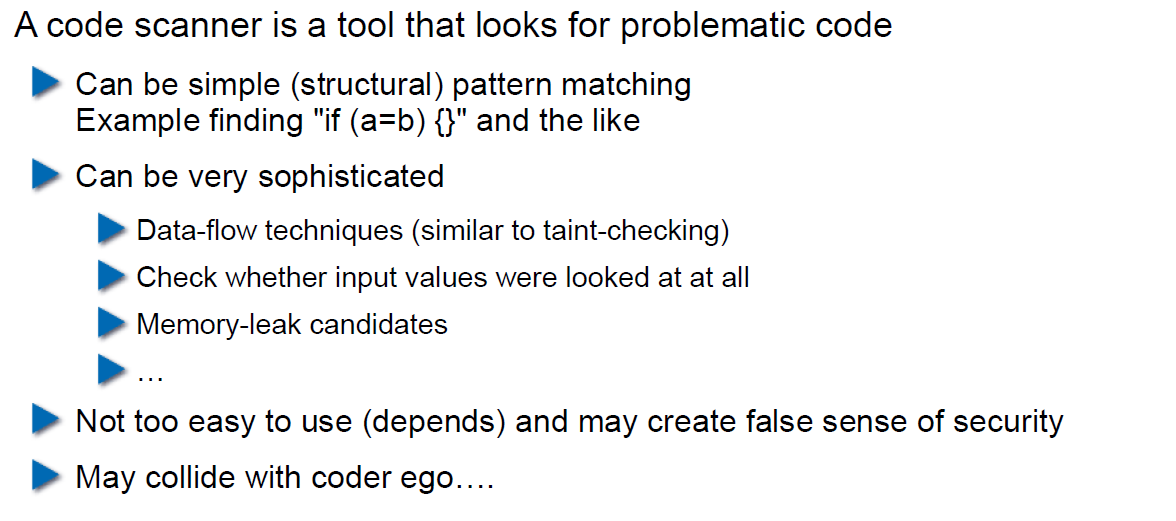
## How does a Pen-Test fail?



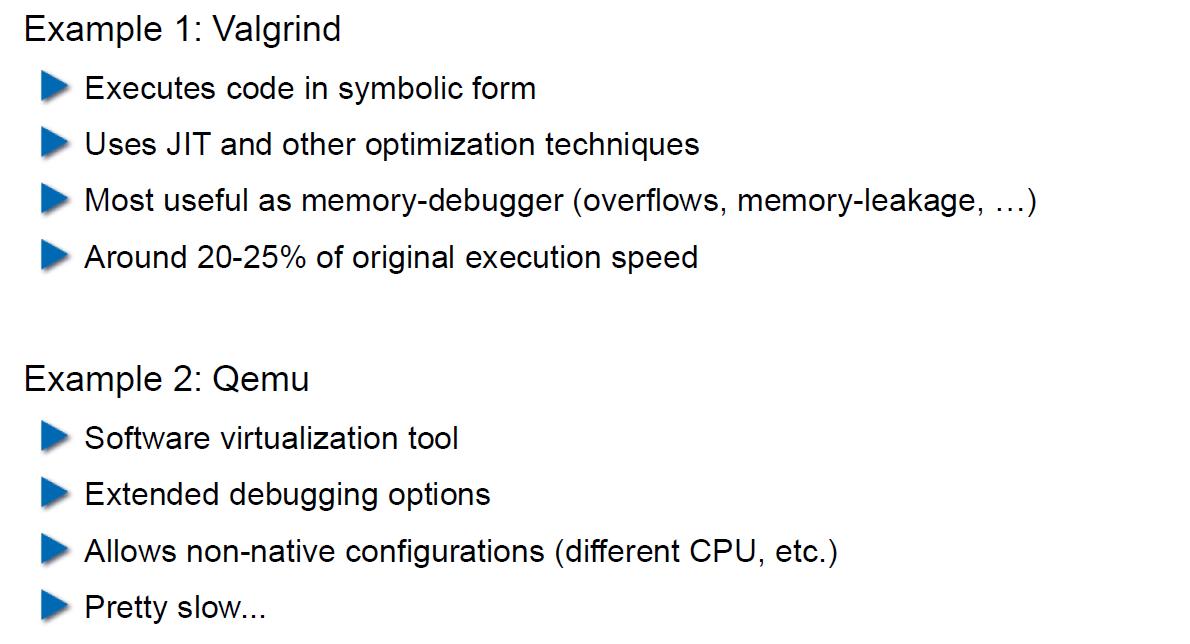
## Code Inspection and Review



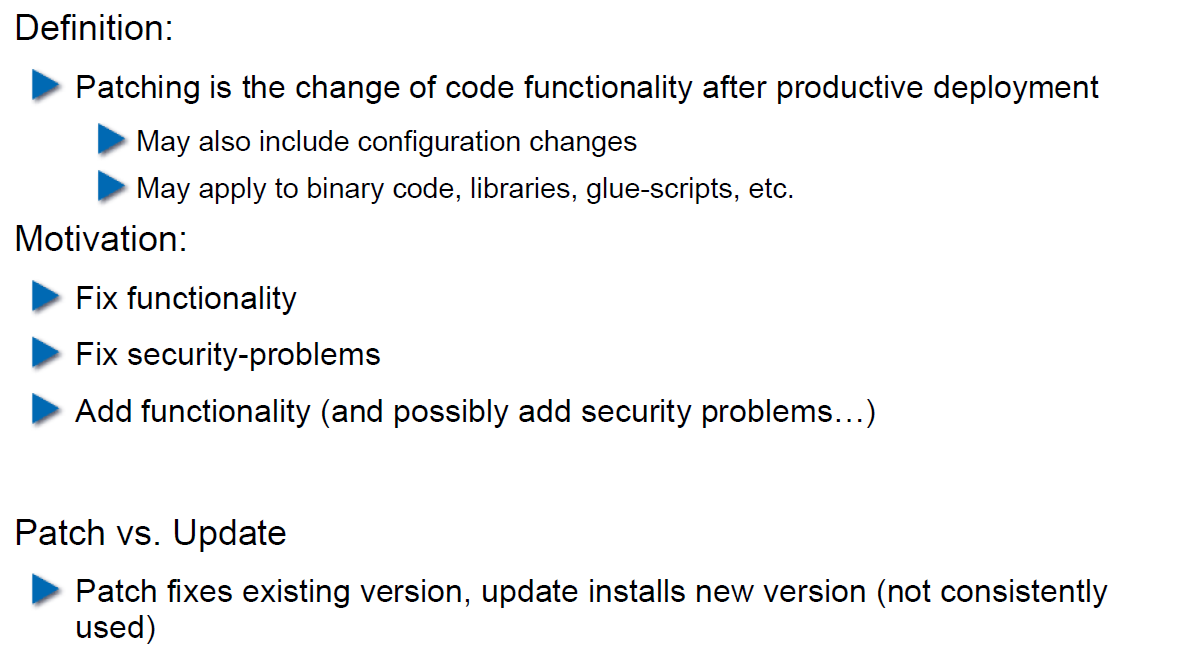
## Code Scanners



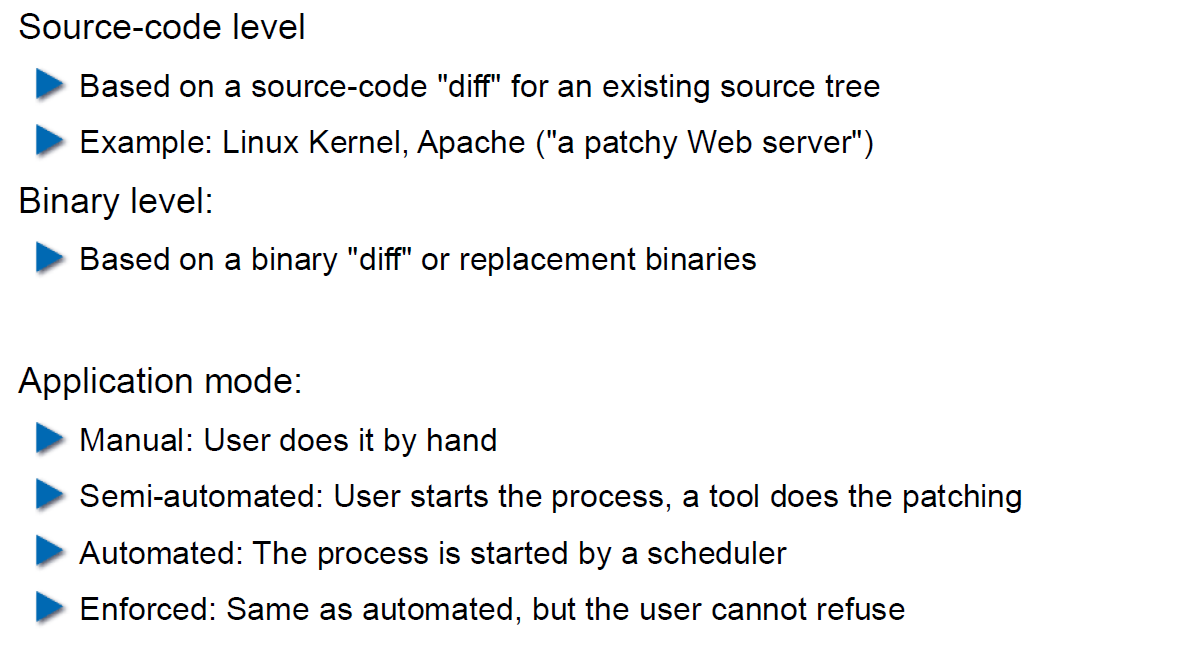
## Execution Emulators



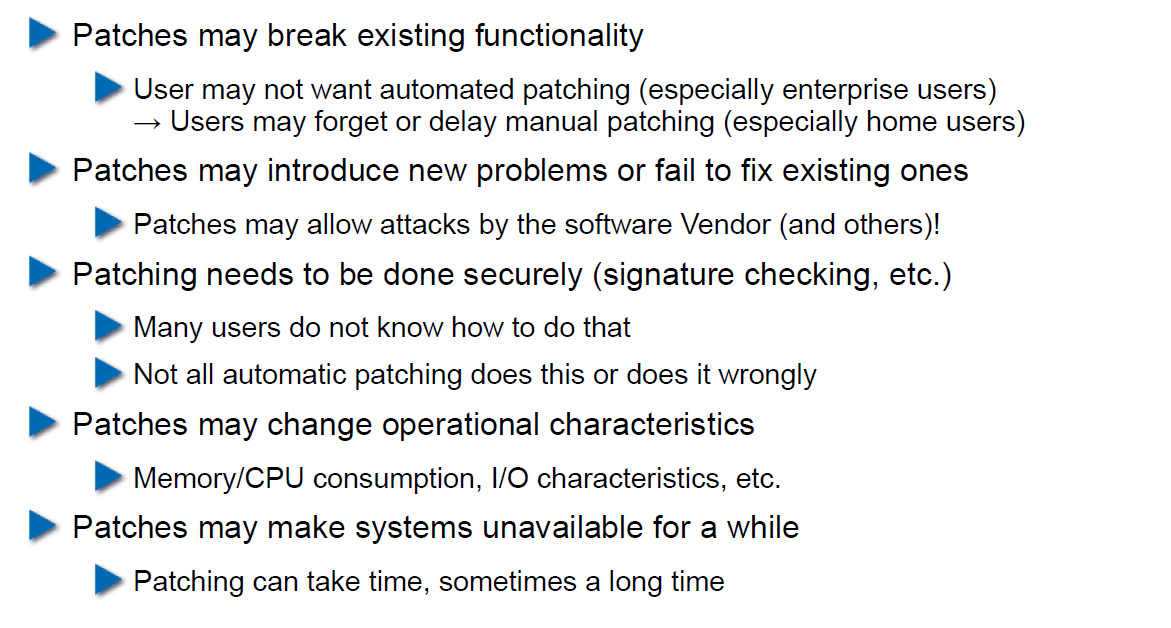
## Patching



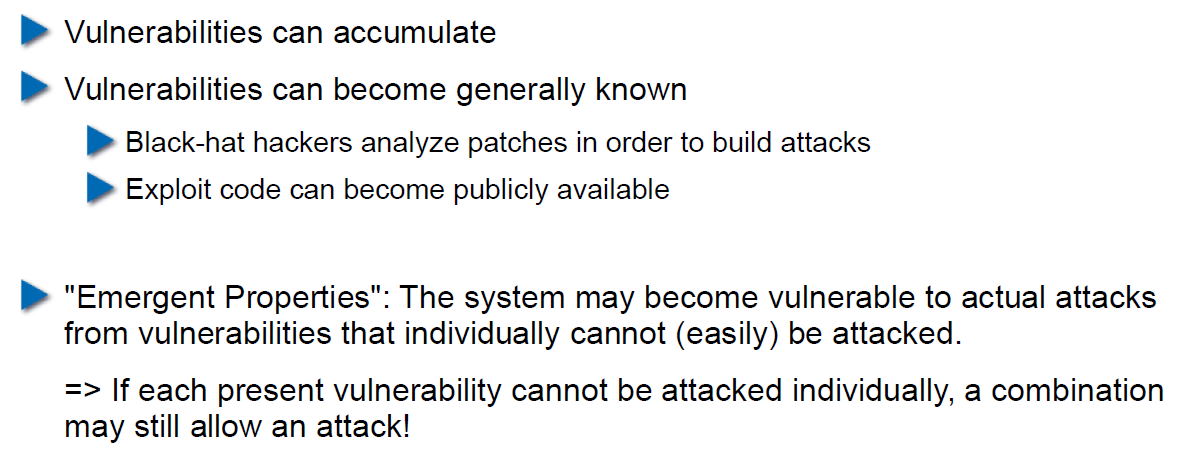
## How to Patch



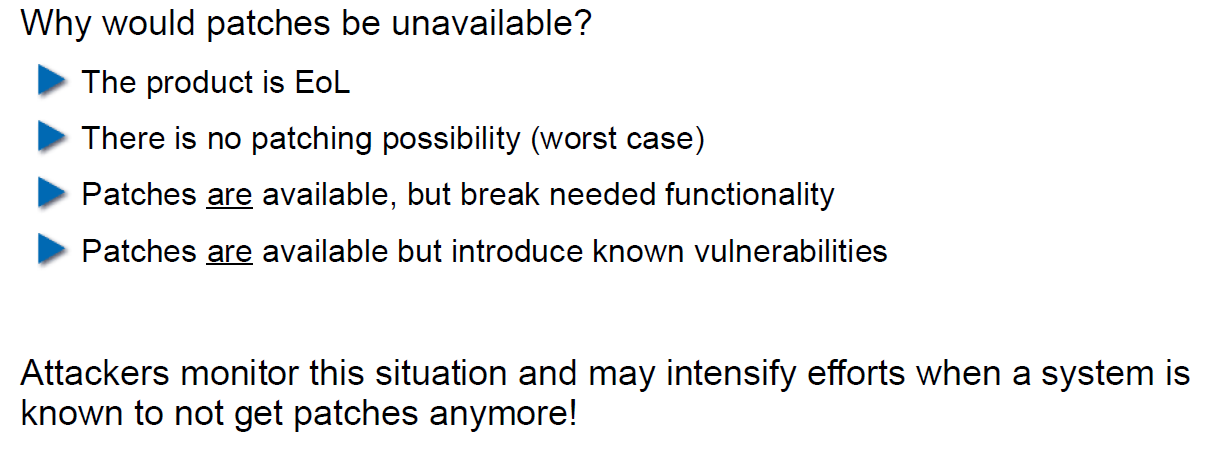
## Problems with Patching



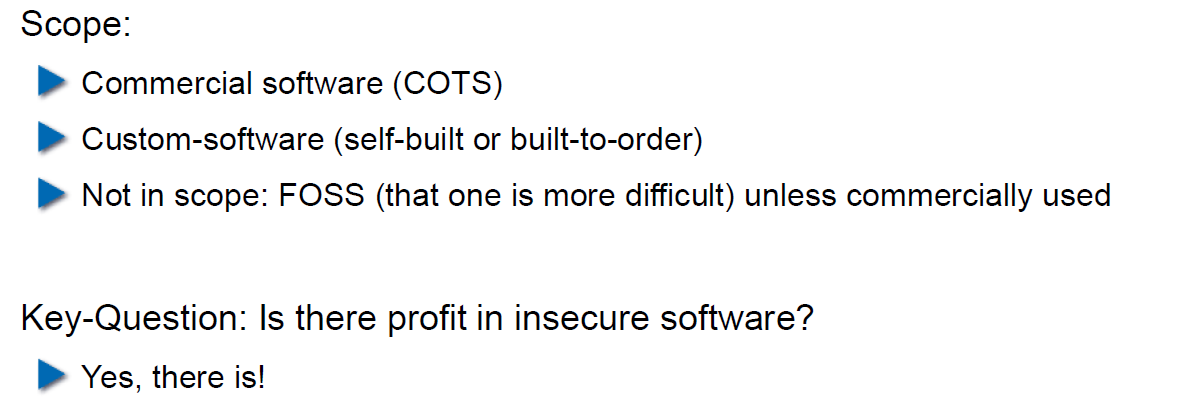
## Problems with not Patching



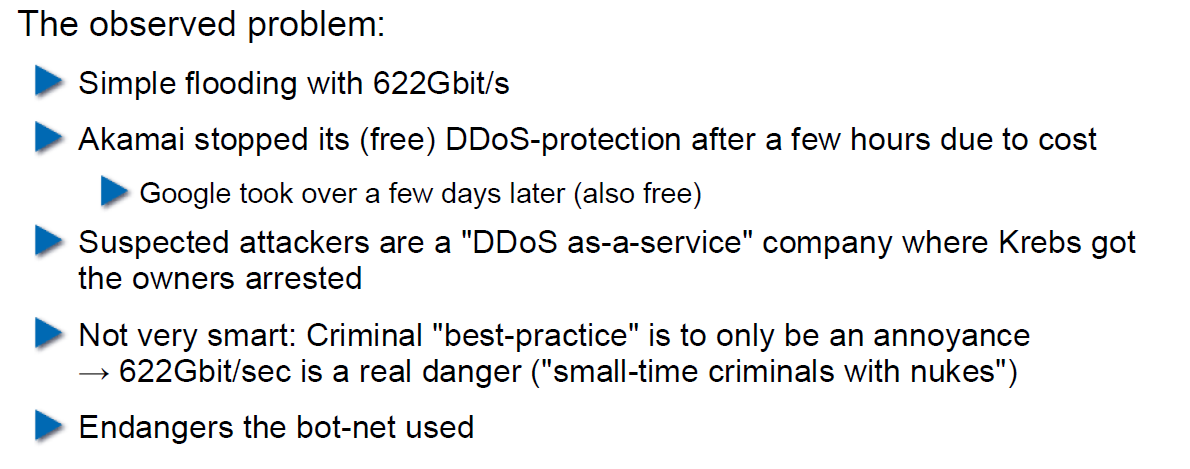
## Problems with Unavailability of Patches



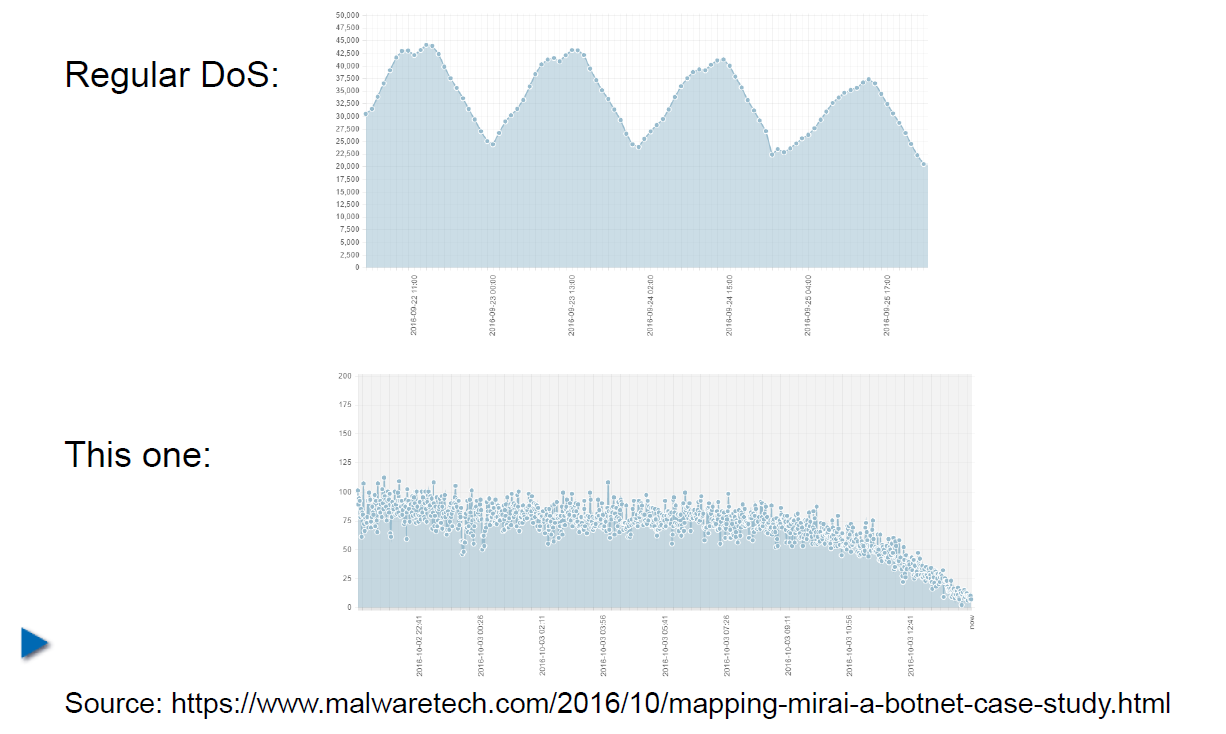
## Economic Aspects of Security



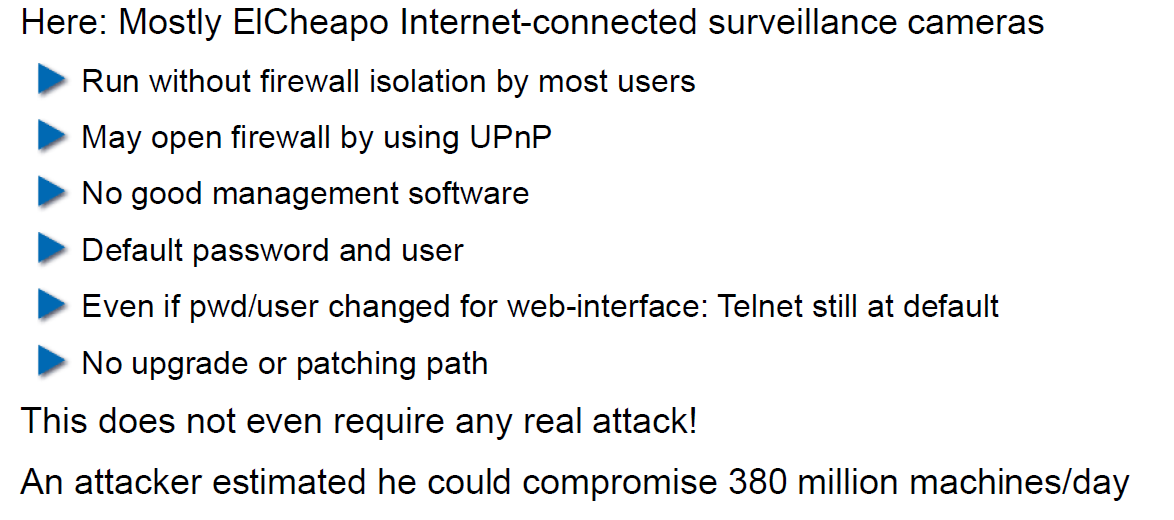
## Case Study: DDoS on Brian Krebs (20.9.2016)



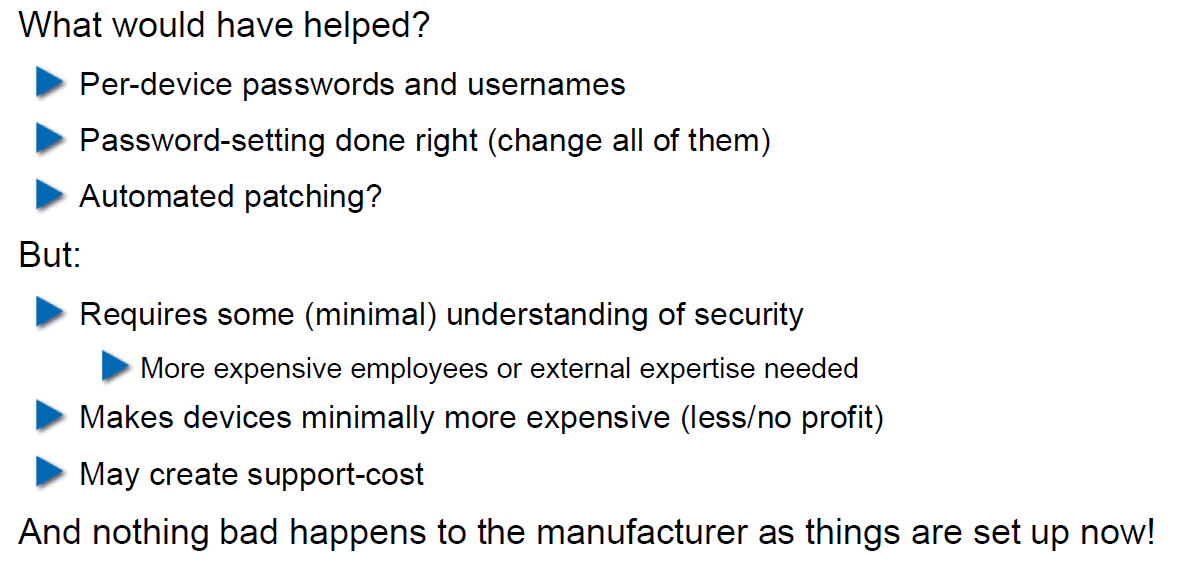
## What can Perform Such an Attack?



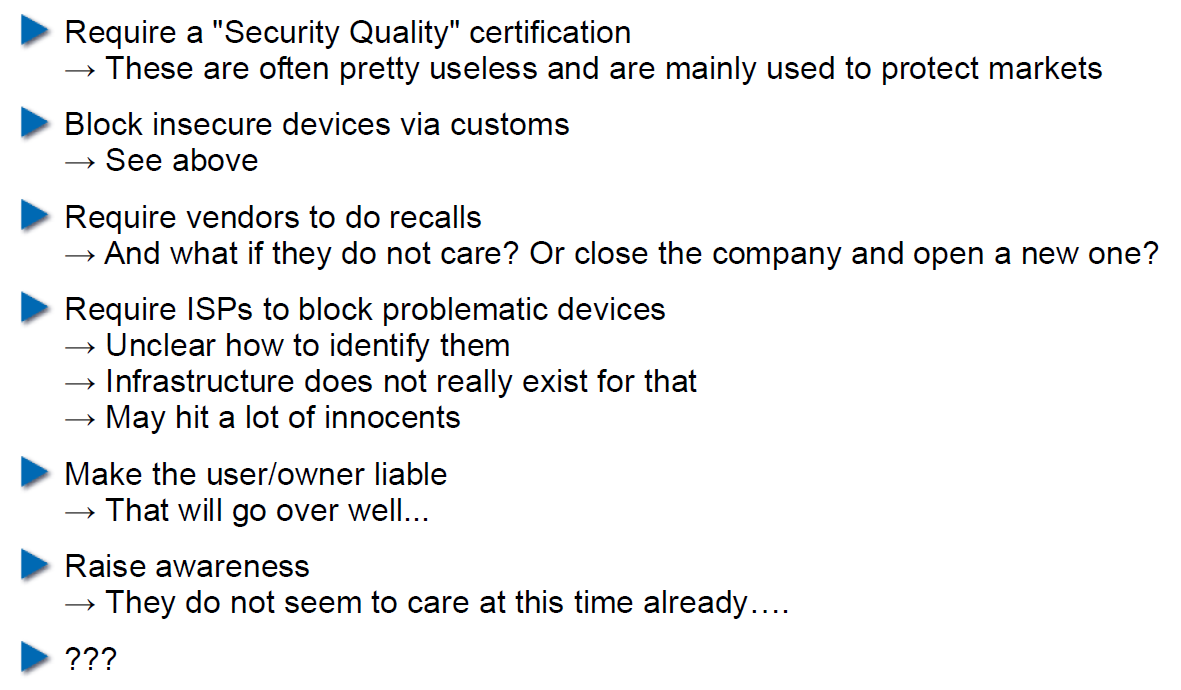
## IoT: Internet of insecure Things



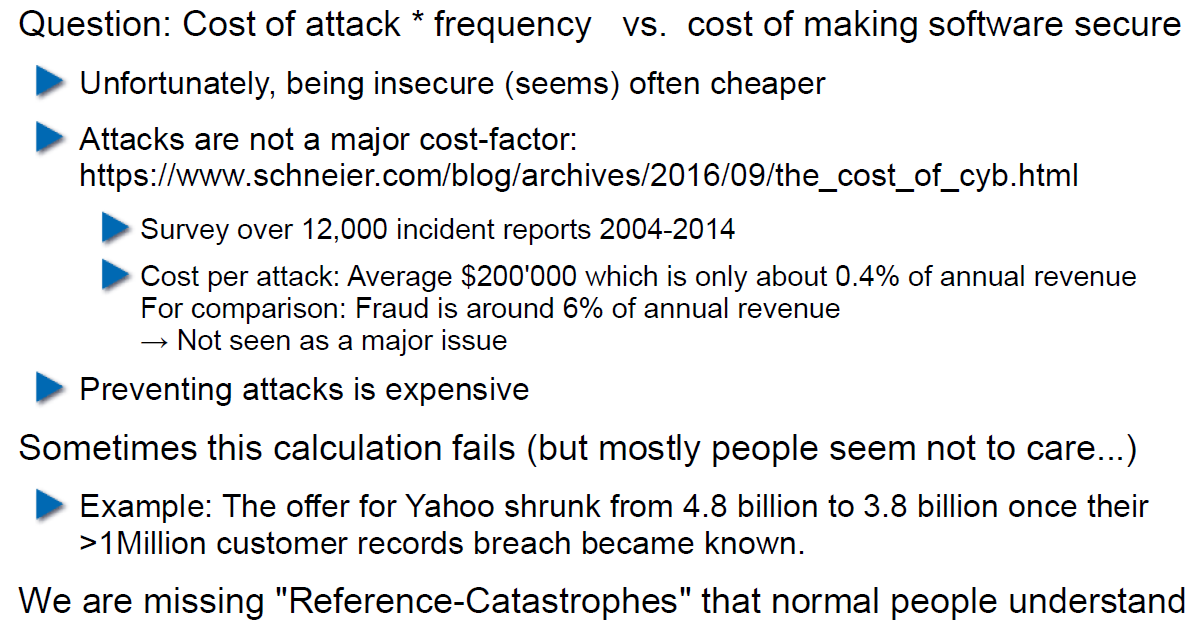
## Why are These IoT Devices so Insecure?



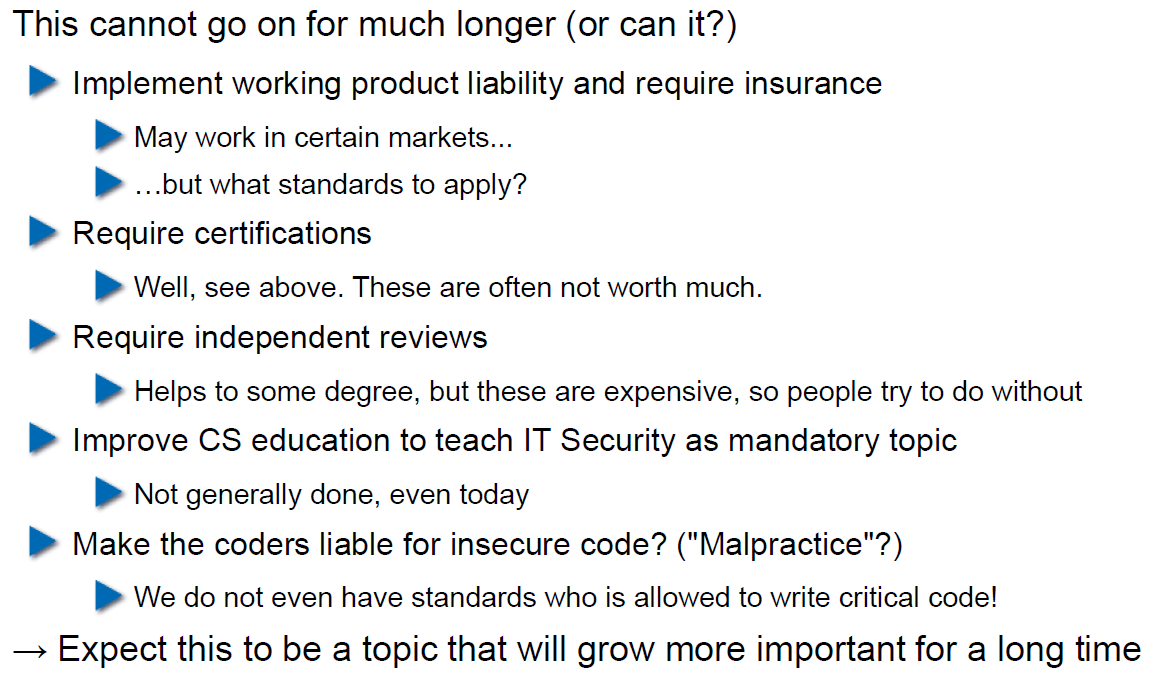
## Possible Solutions



## Does it Pay to make Software Secure?



## So, what to do?



# Woche 5