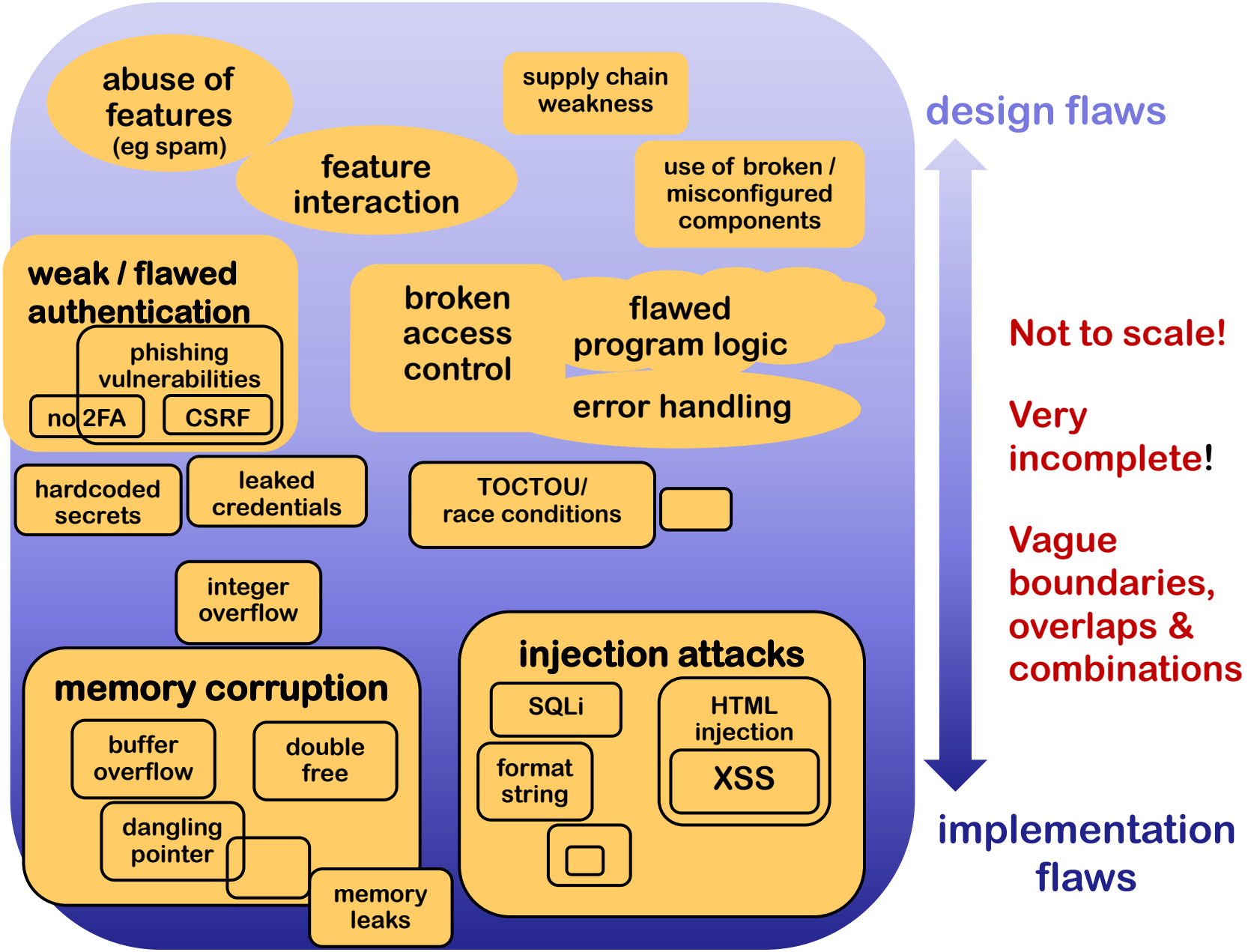
Software Security

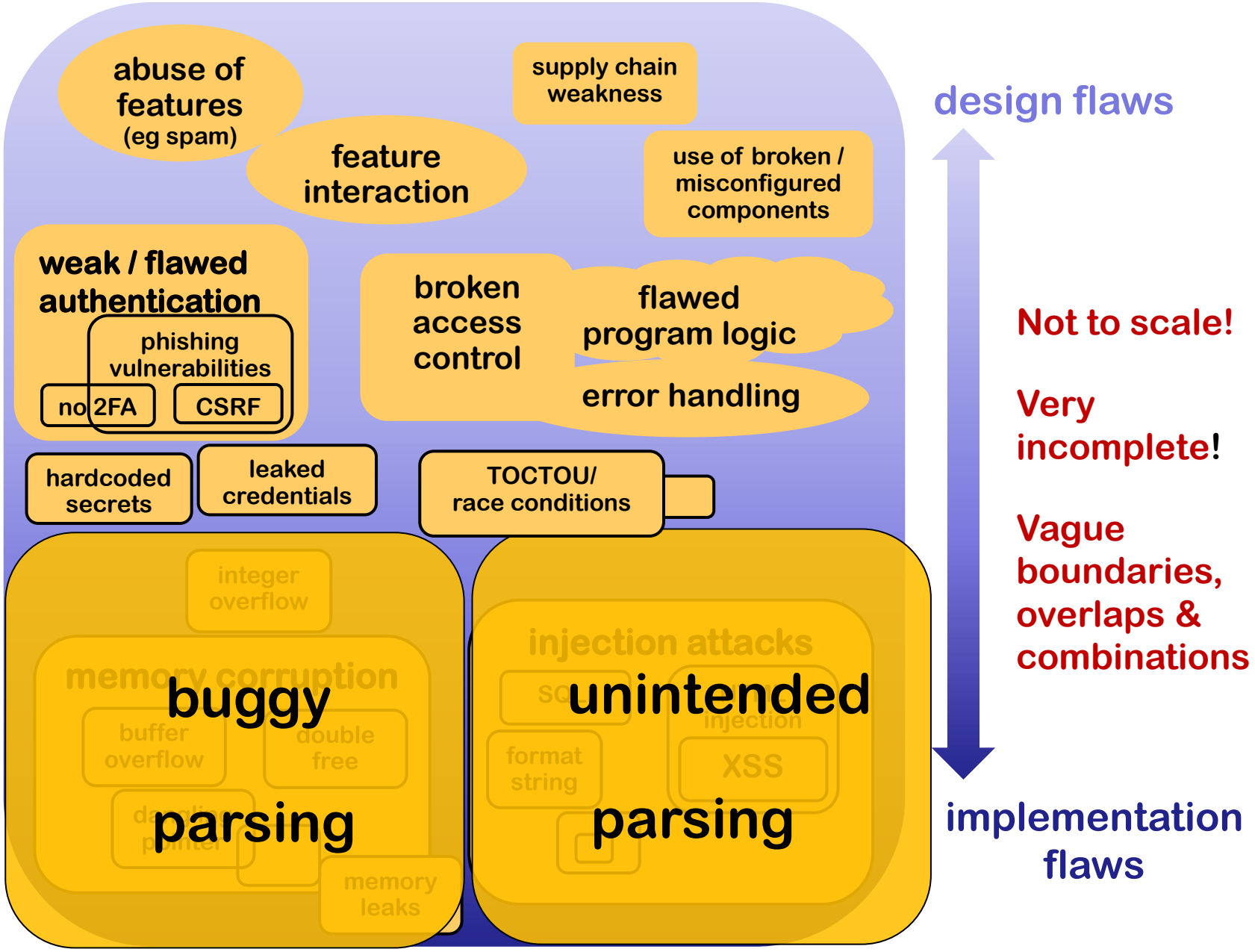
(All?) other kinds of problems

Erik Poll

Digital Security

Radboud University Nijmegen


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
leaked credentials
TOCTOU/ race conditions
integer overflow
memory corruption
buffer overflow
double free
dangling pointer
memory leaks
injection attacks
SQLi
HTML injection
XSS
format string
design flaws
implementation flaws
Not to scale!
Very incomplete!
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
leaked credentials
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!
Vague boundaries, overlaps & combinations

Access Control

Access control - basic concepts

- Access control consists of two steps
  1. authentication - checking who you are
  2. authorisation - enforcing what you are allowed to do

  Often when people say 'access control' they just mean 2.

- Access control, more specifically authorisation, is configured by means of a policy

Where can we expect problems?

Problems with access control

Where can we expect problems?

- Authentication can be weak or broken

  incl. stealing credentials

- Authorisation can be misconfigured

  esp. if policies are COMPLEX

- Access control may be missing / circumventable

Insecure Direct Object Reference (IDOR)

Typical scenario

1. User U requests some resource R (e.g. a file, webpage)
2. Access control system checks if user is allows to access
3. If so, user is given a reference to the resource

   eg https://brightspace.ru.nl/lms/dropbox/admin/folders_manage.d2l?ou=331358.pdf

What could be a potential access control problem?

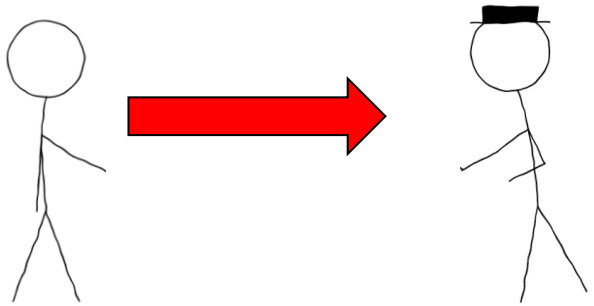
- *Is access control check repeated when the link is used?*
- *What if the user changes the link, to* 331359.pdf, 331360.pdf, ...
- *What if the user passes on this reference to someone else?*

Insecure direct object reference: users are provided a direct reference to some object after access control check, but access control check is not repeated when they use that reference.

Note: this looks a lot like path tranversal.

The confused deputy problem

A less privilege user/process A asks a more privileged user/process B to do something



If B does that with its access rights, A can by-pass access control.

This is an input handling problem,

but also an access control or privilege escalation problem

Many input handling & access control problems involve a confused deputy.

Phishing also involves a confused deputy!

CSRF (Cross Site Request Forgery)

Suppose a bank transfer on `bank.com` is done with a simple click of

```
<a href="transferMoney?amount=1000&toAccount=52.12.57.762">
```

Then attackers can create malicious links

```
<a href="https://bank.com/transferMoney?amount=1000
                          &toAccount=52.12.57.762">
```

and insert such links on `mafia.com` or in HTML-formatted email.

If browser follow such links, attaching its cookie for `bank.com`, we have an CSRF attack.

Note: CSRF is an instance of a confused deputy attack.

CSRF is uncommon these days, as most websites use more robust session management solutions than just an authentication cookie.

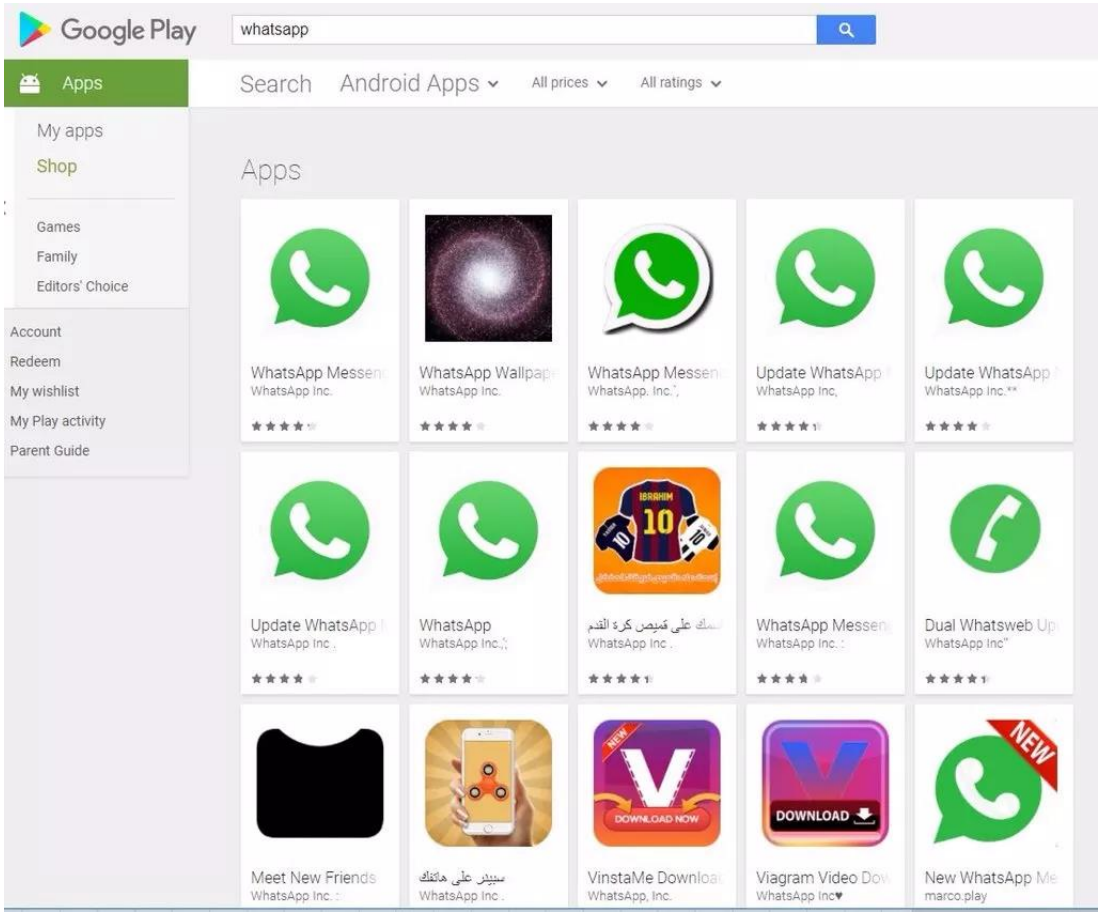
Authentication solutions

1. Authentication of humans by computers
   - passwords, PIN codes, biometrics (fingerprint, face), smartcards & other hardware tokens, MFA, …
2. Authentication of computers by computers (or *computer applications*)
   - cryptography (symmetric; or asymmetric, maybe using certificates & PKI), Kerberos tickets, SMB pass the hash, API keys, …
3. Authentication of computer applications by humans
   - People often forget about this!
   - URLs, info provided by browsers, applications, app stores, …
   - Phishing attacks this form of (usually weak) authentication
     - Phishing is a software design flaw !

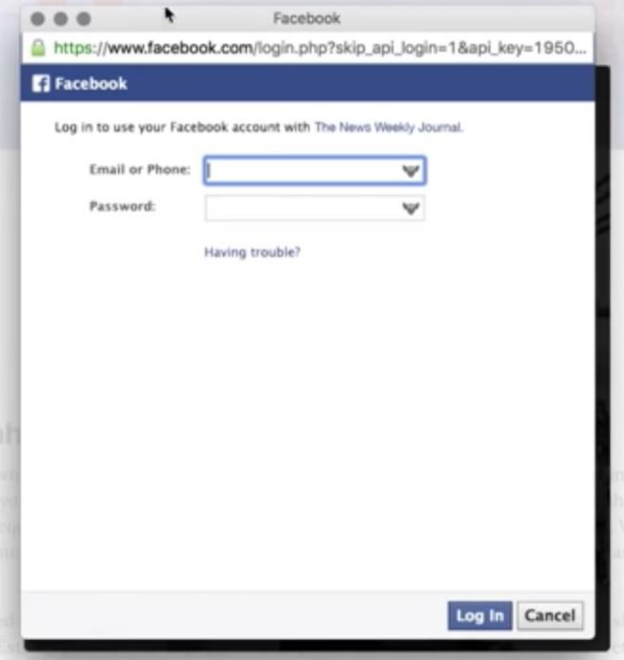
Problems in authenticating apps by humans

Fake WhatsApp App Downloaded Over 1 Million Times

By Don Reisinger last updated February 25, 2020

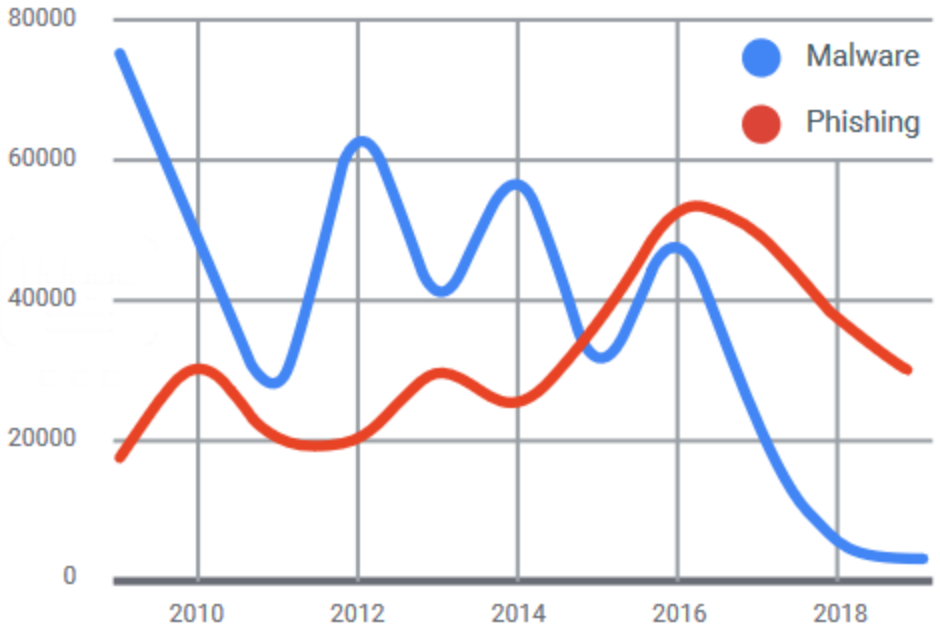


Is this a phishing website?



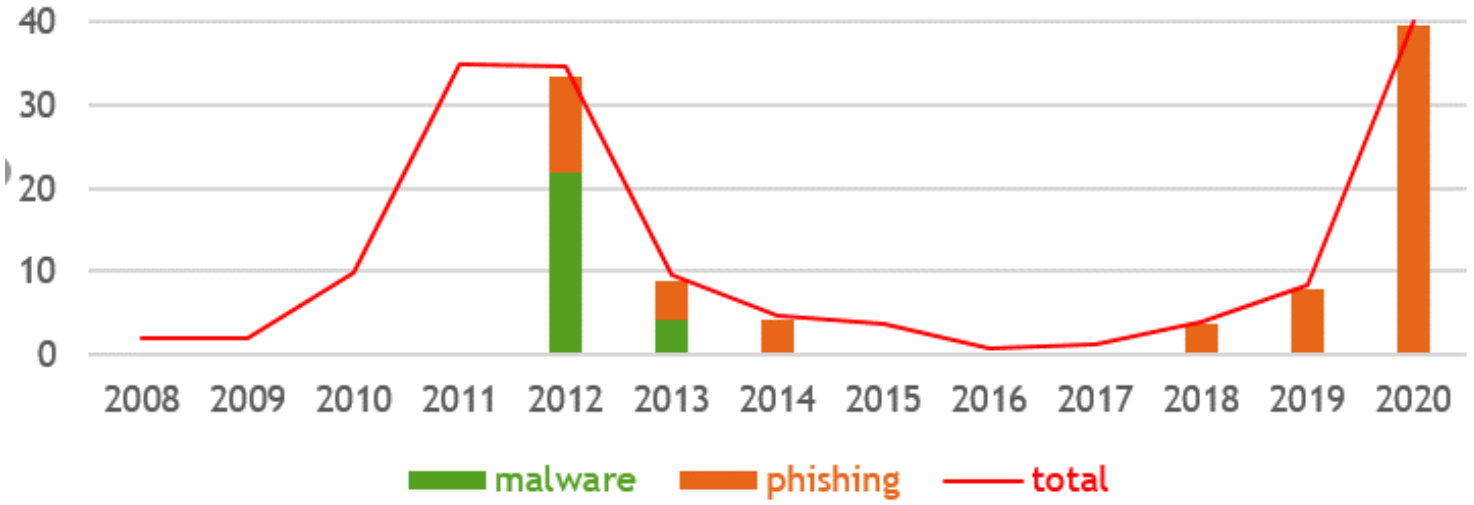
Phishing

Phishing overtook exploit-based malware problems in 2016



[source: Google Transparency report]

Phishing vs malware internet banking fraud



Internet banking fraud in the Netherlands (in million of euros)

[Source: Betaalvereniging]

Non-atomic check and use

aka

TOCTOU (Time of Check, Time of Use)

or

Race conditions

Race condition

- Two concurrent execution threads both execute the statement

  ```
  x = x+1;
  ```

  where x initially has the value 0.
- *What is the value of* `x` *in the end?*

  Answer: `x` can have the value 2 or 1
  - Worse still, in some languages, eg. Java, it can have an arbitrary value

- The root cause of the problem is that `x = x+1` is not an atomic operation, but happens in two steps, reading x and assigning the new value, which may be interleaved in unexpected ways

- Why can this lead to security problems?
- Think of internet banking, and running two simultaneous sessions with the same bank account… *Do try this at home!* ☺

A classic source of (security) problems

- Race condition aka data race is a common type of bug in concurrent programs
  - Basically: two execution threads mess with the same data or object (program variable, file, …) at the same time
  - Not necessarily a *security* bug, but it can be…

- Non-atomic check and use

  aka TOCTOU (Time Of Check, Time of Use)

  is a closely related type of security flaw

  Problem: some precondition required for an action is invalidated between the time it is checked and the time the action is performed
  - Typically, this precondition is access control condition
  - Typically, it involves some concurrency

Classic UNIX race condition

`lpr -r`

- Print utility with `-r` option to remove file after printing
- Could be used to delete arbitrary files

  How?

  1. User executes `lpr -r symlink`
     where `symlink` is a symbolic link
  2. OS checks that user has permission to read & delete this file
  3. While the file is printing move the link is moved, eg to `/etc/passwd`
  4. after printing `lpr`, which has *root permission*, deletes `/etc/passwd`

  Root of the problem: time between check (2) and use (4)

Learning from past mistakes?

`lpr -r` is a classic security flaw from the 1970s, but similar flaws happen decades later

CVE-2003-1073
A race condition in the `at` command for Solaris 2.6 through 9 allows local users to delete arbitrary files via the -r argument with .. sequences in the job name, then modifying the directory structure after at checks permissions to delete the file and before the deletion actually takes place

Combination of race condition with failure to check that file names do not contain ..

Another classic: `mkdir` on Unix

- `mkdir` creates a new directory/folder
- This program executes as root
  - in Linux terminology, it is `setuid root`

- It creates new directory *non-atomically*, in several steps:
  1. enter super-user mode
  2. creates the directory, with owner is root
  3. sets the owner, to whoever invoked `mkdir`
  4. exit super-user mode

- Attack: by creating a symbolic link between steps 2 and 3, attacker can own any file

Example race condition

```
const char *filename="/tmp/erik";
if (access(filename, R_OK)!=0) {
   ... // handle error and exit;
}
// file exists and we have access
int fd open (filename, O_RDONLY);
...
```

Between calls to `access` and `open` the file might be removed, or a symbolic link in the path might be reset!

Race condition & file systems

Interaction with the file system is common source of TOCTOU issues

Signs of trouble:

- Access to files using filenames rather than file handles or file descriptors
  - filenames may point to different files at different moments in time
- Creating files or directories in publicly accessible places, for instance `/tmp`
  - especially if these have predictable file names

Spot the race condition!

```
public class SimpleServlet extends HttpServlet {
  private String query;
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                  throws ServletException, IOException {
    try { Connection conn =
            DriverManager.getConnection("jdbc:odbc ... ");
          query = "INSERT INTO roles" + "(userId, userRole)" +    "VALUES "  + "('"  +
                  request.getParameter("userId") + "'," +
                  "'standard')";
          Statement stmt = conn.createStatement();
          stmt.executeUpdate(query);
          } catch ...
  }
```

Spot the race condition!

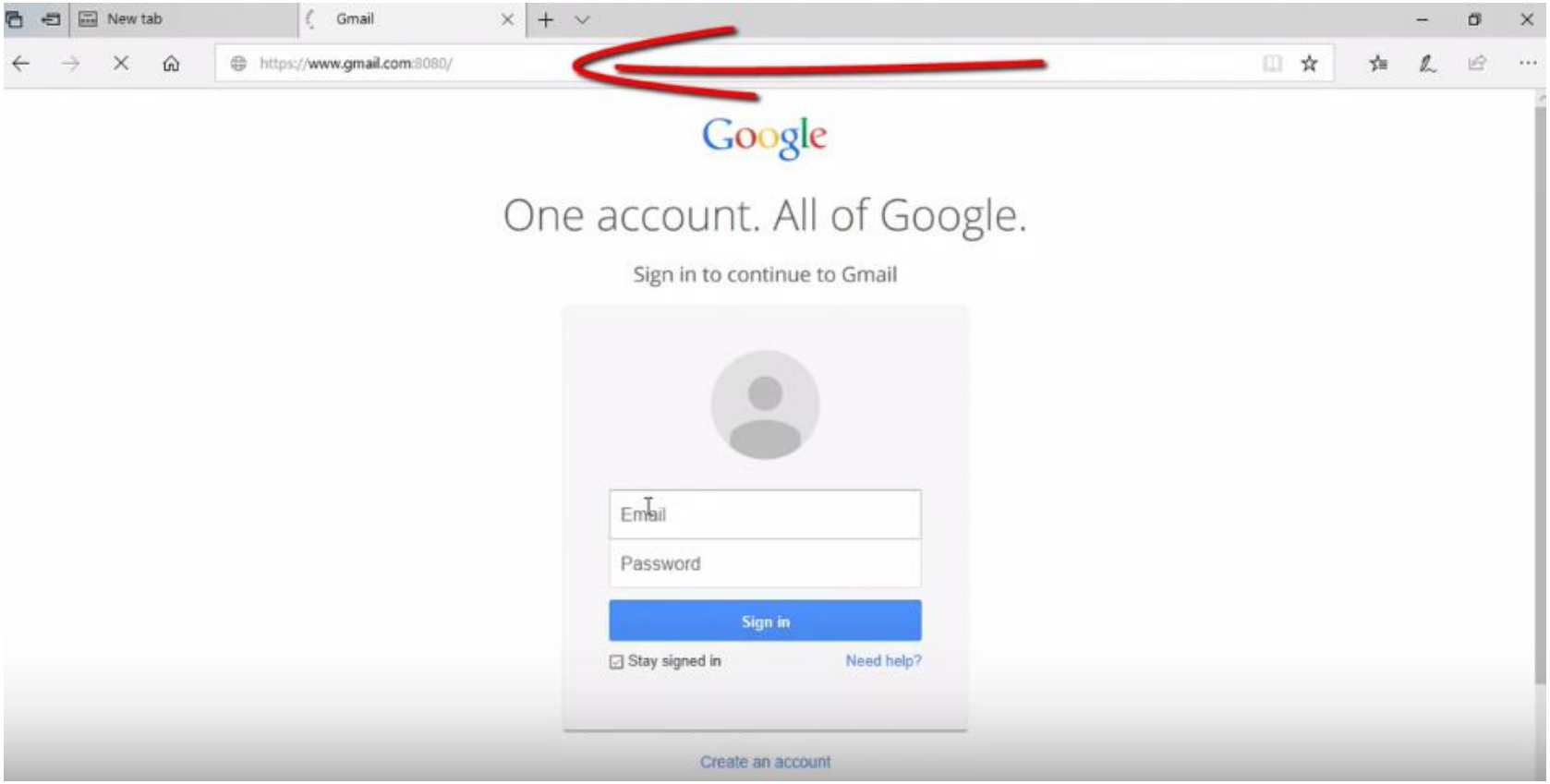
```
public class SimpleServlet extends HttpServlet {
  private String query;
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                throws ServletException, IOException {
    try { Connection conn =
            DriverManager.getConnection("jdbc:odbc ... ");
          query = "INSERT INTO roles" + "(userId, userRole)" +   "VALUES "  + "('"  +
                  request.getParameter("userId") + "'," +
                  "'standard')";
          Statement stmt = conn.createStatement();
          stmt.executeUpdate(query);
          } catch ...
  }
```

Concurrent calls of doGet will act on the *same* Servlet object and hence use the *same* instance field `query`

Spot the race condition!

```
public class SimpleServlet extends HttpServlet {
  private String query;
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                throws ServletException, IOException {
   String query;
    try { Connection conn =
            DriverManager.getConnection("jdbc:odbc ... ");
        query = "INSERT INTO roles" + "(userId, userRole)" +   "VALUES "  + "('"  +
                request.getParameter("userId") + "'," +
                "'standard')";
       Statement stmt = conn.createStatement();
       stmt.executeUpdate(query);
       } catch ...
  }
```

Fix: now every (possibly concurrent) call of doGet has its own `query` field


New tab
Gmail
https://www.gmail.com:8080/
Google
One account. All of Google.
Sign in to continue to Gmail
Email
Password
Sign in
Stay signed in
Need help?
Create an account

Edge & Safari GUI bug [CVE-2018-8383]

Security

Safari, Edge fans: Is that really the website you think you're visiting? URL spoof bug blabbed

Egghead says Apple has yet to patch spoofing vulnerability

By Shaun Nichols in San Francisco 11 Sep 2018 at 05:01 13 SHARE ▼

URL in address bar can be spoofed with a race condition:

- **JavaScript code loads legitimate page; changes address bar, but over non-existent port; and then quickly loads another page**

https://www.theregister.co.uk/2018/09/11/safari_edge_spoofing/

https://youtu.be/Ni2XzF5-ixY

https://youtu.be/dGJSsK55nfQ

Spot the security flaw!

```
char[] pwd;

check_password(char[] guess) {
  for (int i=0; i++; i < guess.length {
      if (pwd[i] != guess[i]) {return false;}
  return true;
}
```

```
char[] pwd;

check_password(char[] guess) {
  for (int i=0; i++; i < pwd.length {
      if (pwd[i] != guess[i]) {return false;}
  return true;
}
```

Spot the second security flaw!

```
char[] pwd;

check_password(char[] guess) {
  for (int i=0; i++; i < pwd.length {
      if (pwd[i] != guess[i]) {return false;}
  return true;
}
```

Timing is a side channel here that leaks information, namely how many characters you got right!

Information Leakage

(incl. side-channel attacks)

Information Leakage

- Fancy examplse: side-channel attacks

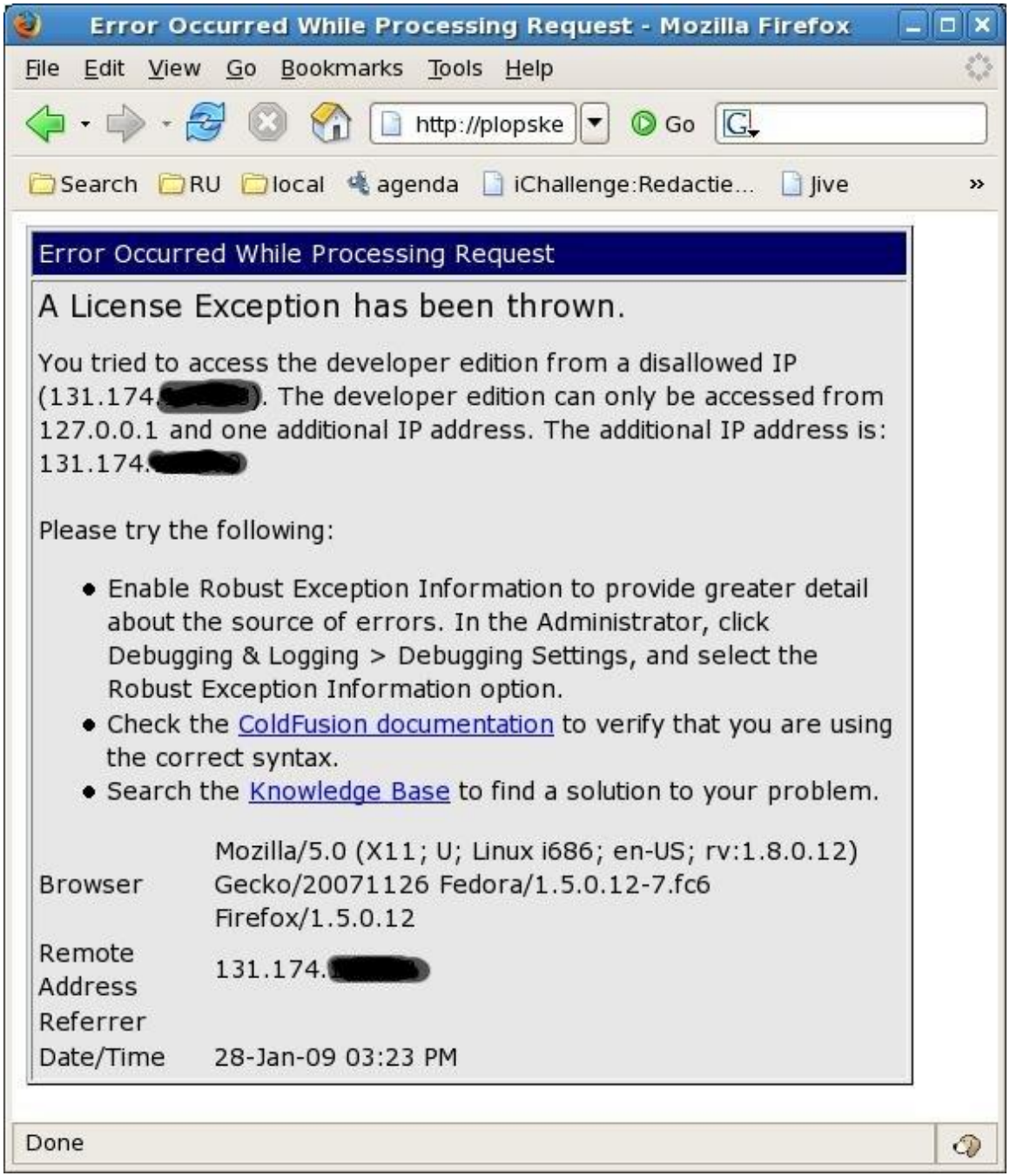
  timing, power analysis, EM-radiation, TEMPEST, ..

  - esp. for crypto keys!
  - since advent of micro-architectural attacks like Spectre & Meltdown a bigger concern for mainstream processors

  More about in the courses: Physical Attacks on Secure Systems, Selected topics on hardware for security, Engineering Cryptographic Software

- Simple examples: errors
  - Error messages as notorious for leaking information
  - Sometimes just the fact *that* an error is produced (recall blind SQL injection), but error messages can leak more info

Error reported by our old course scheduling webpage



Error trace of our institute's old online diary

Database error: Invalid SQL: (SELECT egw_cal_repeats.*,egw_cal.*,cal_start,cal_end,cal_recur_date FROM egw_cal JOIN egw_cal_dates ON egw_cal.cal_id=egw_cal_dates.cal_id JOIN egw_cal_user ON egw_cal.cal_id=egw_cal_user.cal_id LEFT JOIN egw_cal_repeats ON egw_cal.cal_id=egw_cal_repeats.cal_id WHERE (cal_user_type='u' AND cal_user_id IN (56,-135,-2,-40,-160)) AND cal_status != 'R' AND 1225062000 < cal_end AND cal_start < 1228082400 AND recur_type IS NULL AND cal_recur_date=0) UNION (SELECT egw_cal_repeats.*,egw_cal.*,cal_start,cal_end,cal_recur_date FROM egw_cal JOIN egw_cal_dates ON egw_cal.cal_id=egw_cal_dates.cal_id JOIN egw_cal_user ON egw_cal.cal_id=egw_cal_user.cal_id LEFT JOIN egw_cal_repeats ON egw_cal.cal_id=egw_cal_repeats.cal_id WHERE (cal_user_type='u' AND cal_user_id IN (56,-135,-2,-40,-160)) AND cal_status != 'R' AND 1225062000 < cal_end AND cal_start < 1228082400 AND cal_recur_date=cal_start) ORDER BY cal_start mysql

Error: 1 (Can't create/write to file '/var/tmp/#sql_322_0.MYI' ....

File: /vol/www/egw/web-docs/egroupware/calendar/inc/class.socal.inc.php

...

Session halted.

Moral of the story

Informative error messages are great for debugging ☺

But: informative error messages can also help attackers ☹

Did we cover most security problems?

The CWE Top 25

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

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