PA193 Secure coding principles and practices

Seminar 5: Usability and usable security for cryptographic APIs

15. 3. 2022

CROCS
Centre for Research on Cryptography and Security

Martin Ukrop, mukrop@mail.muni.cz
Ph.D. research cooperation
CRoCS, Faculty of Informatics, Masaryk University

Red Hat

Seminar overview

- 10” Usable security intro
- 25” Usable security for developers (5 examples)
- 30” Designing *un*usable APIs
- 30” Generalizing the principles

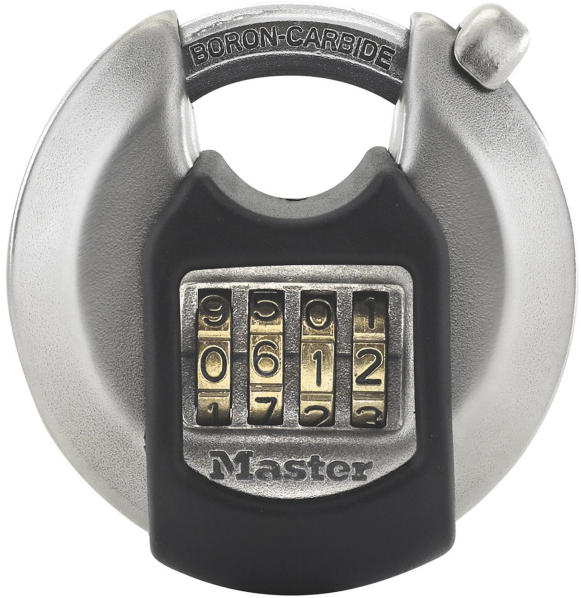
Security vs. usability vs. usable security

Security vs. usability vs. usable security

Combination padlock

Security focus:

- How sturdy is the lock?
- How many times does the attacker need to break the lock loop?
- How many padlock combinations are there?



Security vs. usability vs. usable security

Combination padlock

Usability focus:

- How easy is it to move/read the dials?
- Are alphanumeric combinations easier to remember?

Security vs. usability vs. usable security

Combination padlock

Usable security focus:

- What happens if you forget the combination?
- Do people choose more secure passwords if alphanumeric?
- Can you change the combination?
- What is the default combination?

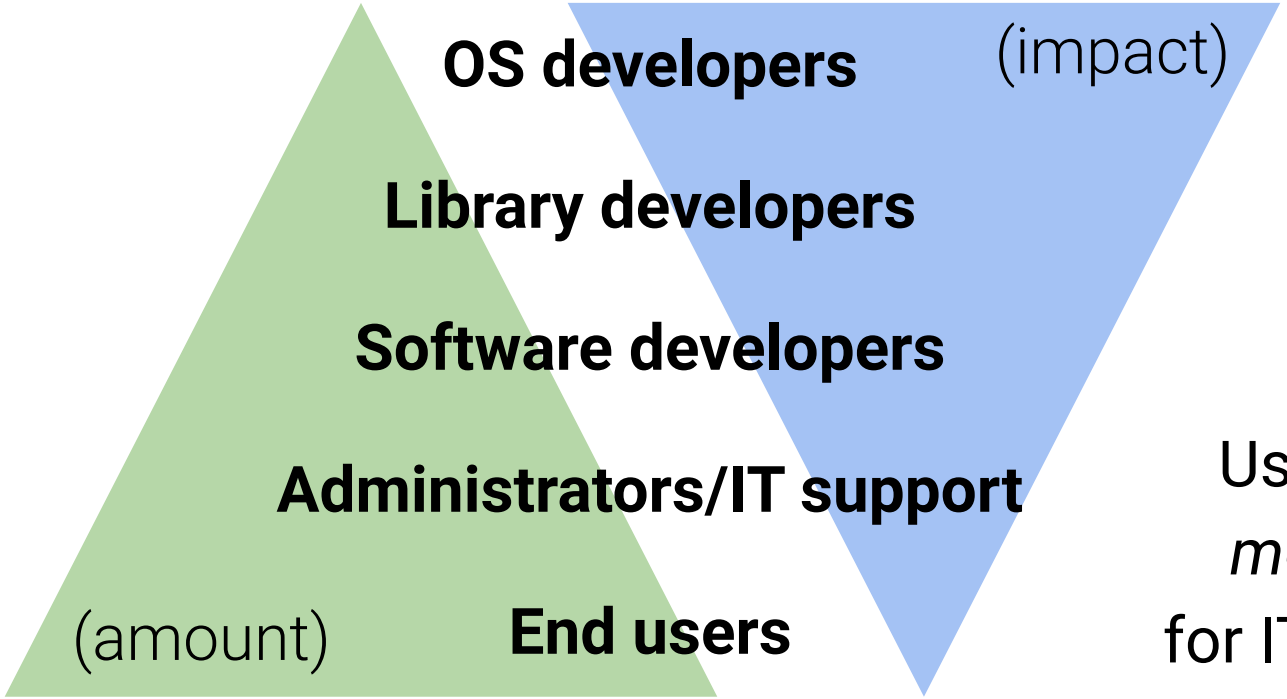
Usable security

Looks into:

- Usability issues with security consequences
- Specifics of the human in the process
  - What would happen if no human was involved?
  - Borders on behavioral sciences, psychology, sociology, pedagogy and others.

- Context is very important!
  - Who is the intended user? What do they know? Where is it?

Usable security: The impact pyramid



Usability is even
more important
for IT professionals!

Where is usability for IT professional?

Everywhere!

- Command line interaction
  - Commands, errors messages, attention management, ...
- File structure & location
  - Logs, configuration files, certificates location, ...
- Interfaces
  - Security APIs, cryptographic operations, ...
- Documentation
  - For security configuration, cryptographic libraries, ...

1: Command line interface

```
[xukrop@styx ~]$ ssh aisa
The authenticity of host 'aisa.fi.muni.cz (147.251.48.1)' can't be established.
ECDSA key fingerprint is SHA256:QcU0hBKPumwmV4WFWf8OzReJc1lLtzr3wVJF5Cqlij8.
ECDSA key fingerprint is MD5:af:79:1b:77:ad:74:3c:35:e3:0b:60:78:f0:a4:3d:7f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'aisa.fi.muni.cz' (ECDSA) to the list of known hosts.
Last login: Mon Nov 20 21:23:45 2017 from eduroam44-237.fi.muni.cz
aisa:/home/xukrop>$
```

- What is the risk?
- What happens after (not) agreeing?

- How can you undo adding a known host?
- Why is last login displayed?
- How much “unwanted” information is appropriate?

1: Command line interface

```
[xukrop@styx ~]$ ssh aisa
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@       WARNING: POSSIBLE DNS SPOOFING DETECTED!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The ED25519 host key for aisa.fi.muni.cz has changed,
and the key for the corresponding IP address 147.251.48.1
is unchanged. This could either mean that
DNS SPOOFING is happening or the IP address for the host
and its host key have changed at the same time.
Offending key for IP in /etc/ssh/ssh_known_hosts:960
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:Js2Haw++eY49mzCoS8ZNMfAKrWXDqSKVpvQmidQvydw.
Please contact your system administrator.
Add correct host key in /home/xukrop/.ssh/known_hosts to get rid of this message.
Offending RSA key in /etc/ssh/ssh_known_hosts:71
ED25519 host key for aisa.fi.muni.cz has changed and you have requested strict checking.
Host key verification failed.
[xukrop@styx ~]$ █
```

2: Configuring product security

- Empirical test of HTTPS deployment usability
  - 28 knowledgeable university students
  - Asked to deploy TLS on Apache to pass security audit
  - What are the challenges?
- Only 4 deployed A-grade TLS
  - Challenges: Find the right information, generate CSR, choose appropriate cipher-suites, strict HTTPS, multiple config files, security/compatibility ballance

Research from *"I Have No Idea What I'm Doing" – On the Usability of Deploying HTTPS*, by K. Krombholz, W. Mayer, M. Schmiedecker and E. Weippl, 2017.

3: Software library configuration

- Libcurl
  - the multiprotocol file transfer library

- Two main directives for SSL validation
  - **`CURL_SSL_VERIFYPEER`** (checking certificate)
  - **`CURL_SSL_VERIFYHOST`** (checking hostname)

3: Software library configuration

- PayPal SDK: version from 27th April 2012

```
curl_setopt($ch, CURL_SSL_VERIFYPEER, TRUE)
curl_setopt($ch, CURL_SSL_VERIFYHOST, TRUE)
```

- **Bool** CURL_SSL_VERIFYPEER
- **Int** CURL_SSL_VERIFYHOST
  - 0: no host verification
  - 1: debug (nearly no verification)
  - 2: verify hostname

CURL example from *The Most DangerousCode in the World: Validating SSL Certificates in Non-Browser Software*, 2012.

4: APIs of cryptographic libraries

- How to get authenticated encryption?
  - that is, confidentiality + integrity
- Composition of encryption and MAC
  - MAC-plaintext-then-encrypt?
  - MAC-ciphertext-after-encrypt?
  - Encrypt-and-MAC-plaintext?

} Do you remember which is the most secure?

- Hide this in by the library interface (NaCl/libsodium)
  - `c = crypto_box(m, n, pk, sk);`
  - `m = crypto_box_open(c, n, pk, sk);`
  - What may be the problem here?

5: APIs of randomness generators

OpenSSL functions for random bytes

```
int RAND_bytes(unsigned char *buf, int num);
```

- Should always be cryptographically strong

```
int RAND_pseudo_bytes(unsigned char *buf, int num);
```

- Pseudorandom, can be strong (see later)

Working as *interfaces*

- *I.e. multiple implementations*
- *I.e. multiple callers*

5: APIs of randomness generators

Documentation: Return values

- ***RAND_bytes()*** returns *1* on success, *0* otherwise. The error code can be obtained by *ERR_get_error(3)*.
- ***RAND_pseudo_bytes()*** returns *1* if the bytes generated are cryptographically strong, *0* otherwise.
- Both functions return *-1* if they are not supported by the current RAND method.

Can you spot the unusable bit?

5: APIs of randomness generators

Implementations: Often the same for both...

Callers: Standard C error scheme incorrect:

```
if (!RAND_bytes(...)) /* handle error ... */
```

- Debian code search: OpenSSL, Ruby, net-snmp, ZNC, DACS, dnsval/dnssec-tools, ...
- GitHub code search: 1 456 results

RAND example by Joseph Birr-Pixton (http://jbp.io)

“Developer-resistant cryptography!”

K. Cairns and G. Steel, 2014

Developer-resistant cryptography!

"To decrease the number of security holes caused by developers, we should not only do everything possible to educate developers, but also make APIs easier to use. By examining errors commonly found in today's applications we can find the weaknesses of today's APIs. ***Using what we learn, we can design APIs that are more intuitive and easier to use correctly, decreasing the overall chance of human error.***"

K. Cairns and G. Steel: *Developer-Resistant Cryptography*. STRINT Workshop, 2014.

Summary: Interface/API issues

- Usability issues
  - *E.g. counterintuitive structure/names/arguments*
- Performance issues
  - *E.g. making copy of structures in arguments*
- Security issues
  - Due to bad technical design
    - *E.g. possible compromise by unexpected call order*
  - Due to bad usability
    - *E.g. prone to accidentally disable certificate verification*

Task: Intentional bad design

Design a C (or C-like) API for encryption/decryption routines with security issues due to bad usability.

- Work in groups of 2-3 people
- Get inspiration from existing API structure
  - OpenSSL AES header, OpenSSL enc/dec Wiki
  - GnuTLS encryption, API reference
  - NaCl encryption
  - mbedTLS encryption module

Share the design

Where is the unusability?

Format: Round-Robin

How to make usable APIs?

How can we generalize?

Format: Think-Pair-Share

Bonus: Usable design in security

1. You are not your user
   - *Don't forget user testing!*
2. Prefer system usability over user training
   - *The best training is no training!*
3. The user is not your enemy
   - *Cooperate with them to have security*
4. Think of usability on all levels
   - *Should the question be asked in the first place?*
5. Have secure defaults
   - *Security is usually not a primary goal*

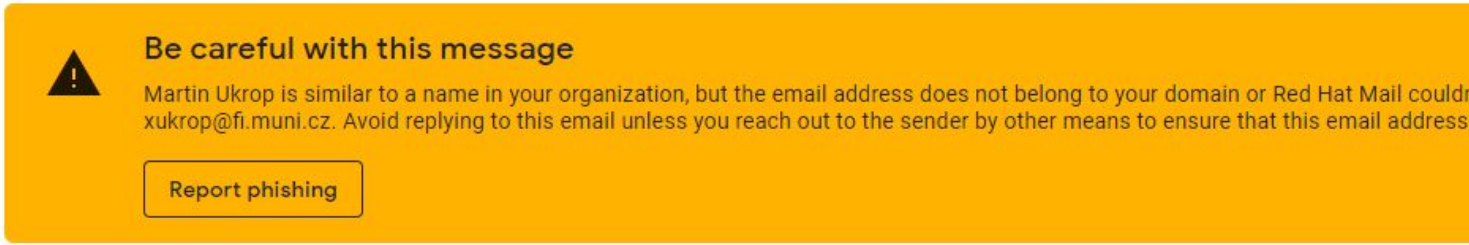
1. You are not your user

- Preferences are subjective
  - The same thing may be differently usable for different people
  - You (IT pro, dev, admin) may have a *very* different view than “the common Johnny”
- User testing is crucial
  - User studies, surveys, focus groups, interviews
  - Beware: self-reported data vs. objective measures

2. System usability over user training

- Training users to behave more securely
  - E.g. check sender email address for phishing attacks
  - Training costs, variable adherence
- Improving the system usability
  - E.g.

Gmail warning, same name as one in the domain, 2019.

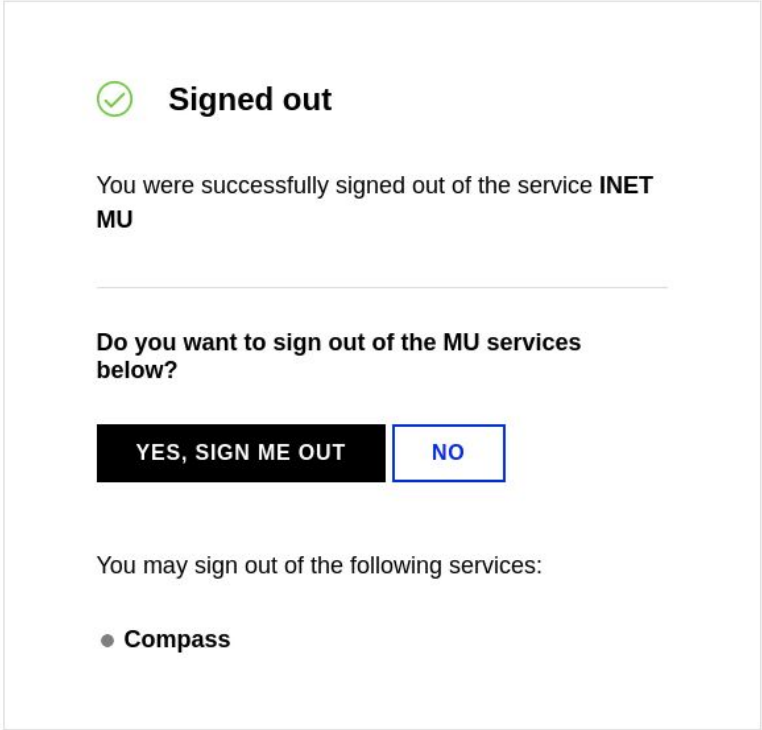


  - Higher development cost, user studies necessary
- Ideally: Combine usability & training

3. The user is not your enemy

- The user is your partner
  - They have knowledge you don't
  - You have knowledge they don't

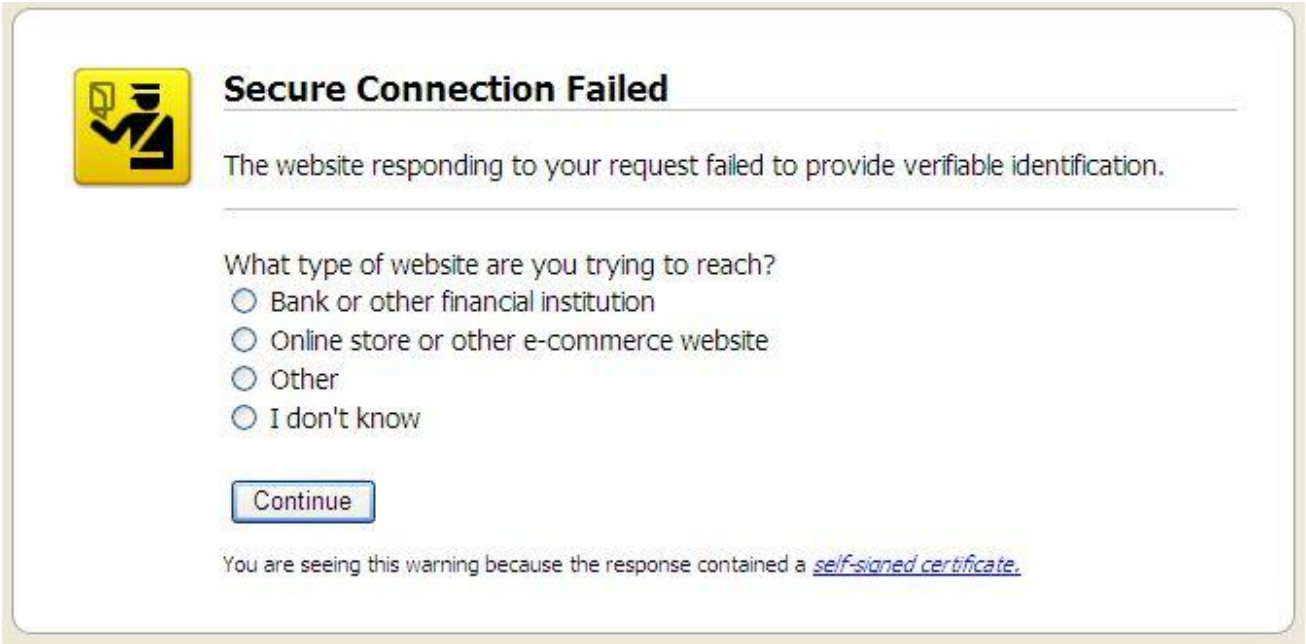
  -> collaborate!
- Examples:
  - SSO sign-out of other services
  - After password leak, ask the user to change the same password on other sites



Unified MU logout, 2020

3. The user is not your enemy

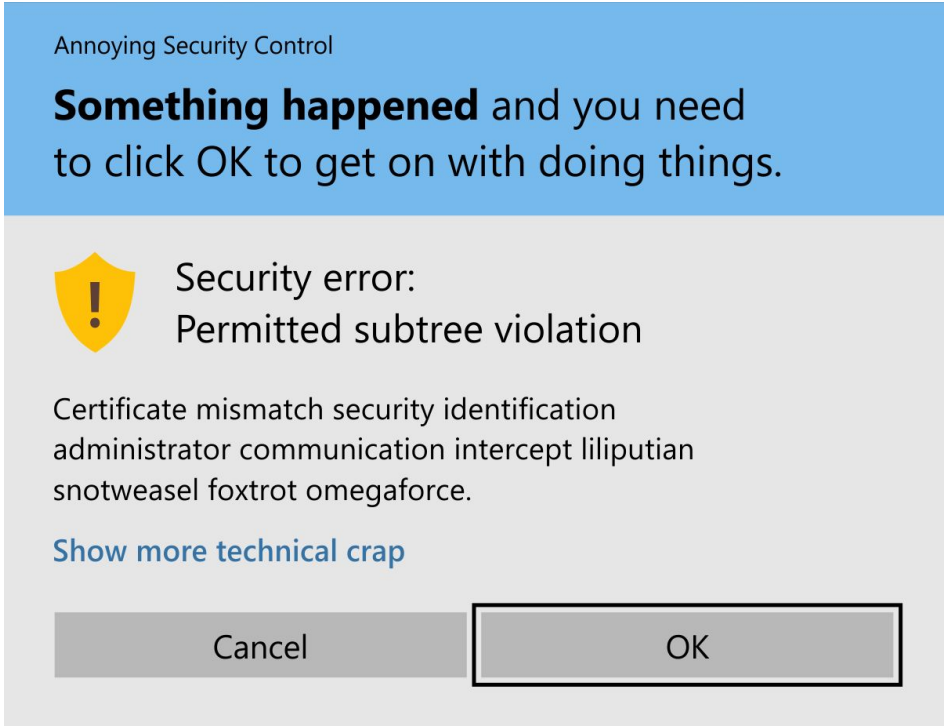
- Possible collaboration in browser warnings



J. Sunshine et al.: *Crying Wolf: An Empirical Study of SSL Warning Effectiveness*. USENIX 2009.

3. The user is not your enemy

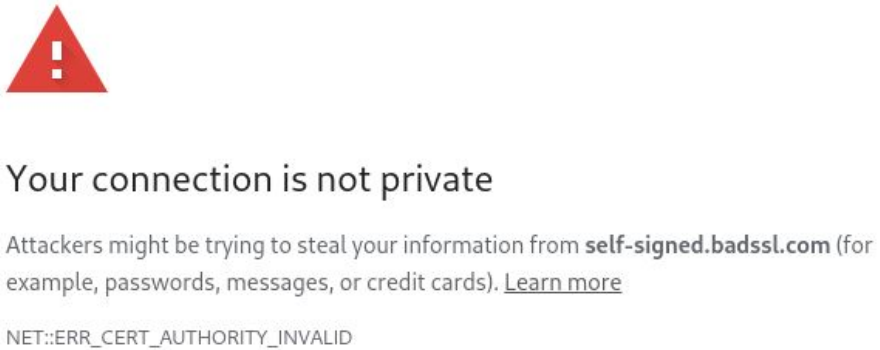
- But don’t be alibistic!
  - Ask only what the user will understand.
  - Use proper language!



Joke adapted from Johnatan Nightingale

4. Think of usability on all levels

- Usable warning message
  - Wording, explanation, icon, nudging to secure behaviour, ...
- Take a step back
  - It may be more usable to not ask at all!
  - Revoked certificates does not issue a warning (it refuses the connection right away!).



Google Chrome (v87), self-signed certificate, 2020

5. Have secure defaults

- Security is a secondary goal
  - Help them do it securely, but don't prevent them from doing it!

Dilbert 2007-11-16

  - Thus, the default way should be the most secure way.

Dancing pigs

"Given a choice between dancing pigs and security, users will pick dancing pigs every time."

(Edward Felten, Princeton University, 1999)

Message text from Secrets & Lies by Bruce Schneier (John Wiley & Sons, 2000).