OWASP AppSec Europe
London 2nd-6th July 2018

Usable Security for Developers: A Nightmare

Achim D. Brucker | @adbrucker

OWASP AppSec Europe
London 2nd-6th July 2018
Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker

Usable Security for Developers: A Nightmare

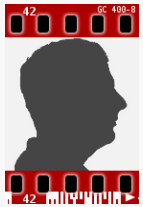
Abstract

The term “usable security” is on everyone’s lips and there seems to be a general agreement that, first, security controls should not unnecessarily affect the usability and unfriendliness of systems. And, second, that simple to use system should be preferred as they minimize the risk of handling errors that can be the root cause of security incidents such as data leakages.
But it also seems to be a general surprise (at least for security experts), why software developers always (still) make so many easy to avoid mistakes that lead to insecure software systems. In fact, many of the large security incidents of the last weeks/months/years are caused by “seemingly simple to fix” programming errors.
Bringing both observations together, it should be obvious that we need usable and developer-friendly security controls and programming frameworks that make it easy to build secure systems. Still, reality looks different: many programming languages, APIs, and frameworks provide complex interfaces that are, actually, hard to use securely. In fact, they are miles away from providing usable security for developers.
In this talk, I will discuss examples of complex and “non-usable” security for developers such as APIs that, in fact, are (nearly) impossible to use securely or that require a understanding of security topics that most security experts to not have (and, thus, that we cannot expert from software developers).

OWASP AppSec Europe
London 2nd-6th July 2018
Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker

About Me

- Security Expert/Architect at SAP SE
  - Member of the central security team, SAP SE (Germany)
    - Security Testing Strategist
  - Work areas at SAP included:
    - Defining the risk-based Security Testing Strategy
    - Evaluation of security testing tools (e.g., SAST, DAST)
    - Roll-out of security testing tools
    - Secure Software Development Life Cycle integration
    - …
- Since December 2015:
  - Associate Professor, The University of Sheffield, UK
  - Head of the Software Assurance & Security Research Team
  - Available as consultancy & (research) collaborations



https://www.brucker.ch/

OWASP AppSec Europe
London 2nd-6th July 2018
Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker

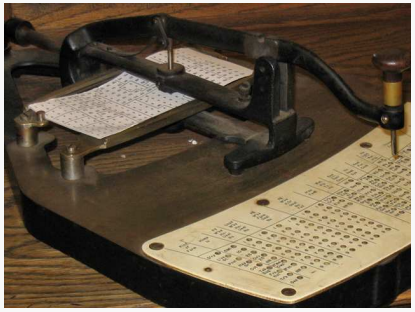
Outline

1. Security experts and developers
2. Secure programming cant’ be that difficult …
3. The most common “fixes”
4. What we should do

Usable Security for Developers: A Nightmare

Achim D. Brucker | @adbrucker

70 years of software development



Since the late 1940ies, we

- program,
- debug, and
- patch

computer systems.

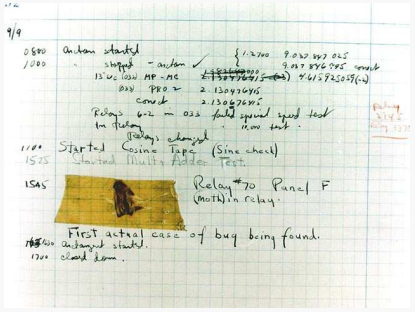
- we do not use punch cards anymore …

OWASP AppSec Europe London 2nd-6th July 2018

Usable Security for Developers: A Nightmare

Achim D. Brucker | @adbrucker

70 years of software development



Since the late 1940ies, we

- program,
- debug, and
- patch

computer systems.

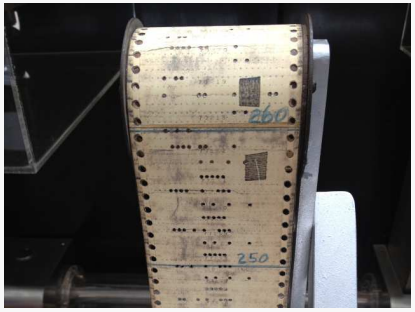
- we do not use punch cards anymore …

OWASP AppSec Europe London 2nd-6th July 2018

Usable Security for Developers: A Nightmare

Achim D. Brucker | @adbrucker

70 years of software development



Since the late 1940ies, we

- program,
- debug, and
- patch

computer systems.

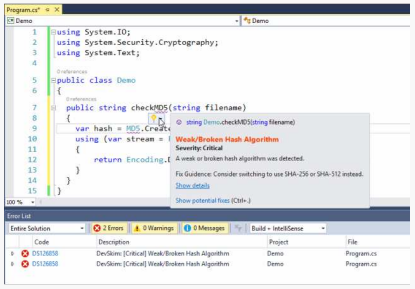
- we do not use punch cards anymore …

OWASP AppSec Europe London 2nd-6th July 2018

Usable Security for Developers: A Nightmare

Achim D. Brucker | @adbrucker

70 years of software development



Since the late 1940ies, we

- program,
- debug, and
- patch

computer systems.

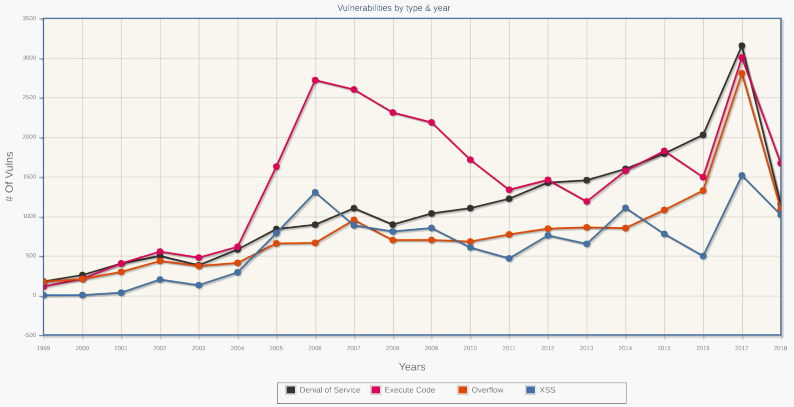
- we do not use punch cards anymore …

Usable Security for Developers: A Nightmare

Achim D. Brucker | @adbrucker

We build software since 70 years
and still make the same old (security) mistakes

Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker



Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker

The common “silver bullet”: The SDLC

Training → Risk Identification → Plan Security Measures → Secure Development & Security Testing → Security Validation → Secure Operations → Security Response

- Central **security experts** (SDLC owner)
  - Organizes security trainings
  - Defines product standard “Security”
  - Defines security testing strategy
  - Validates products
  - …
- Development teams
  - Select technologies
  - Select development model
  - Design and execute security testing plan
  - …

Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker

Works nicely
in theory – let’s move to reality

Usable Security for Developers: A Nightmare

Achim D. Brucker | @adbrucker

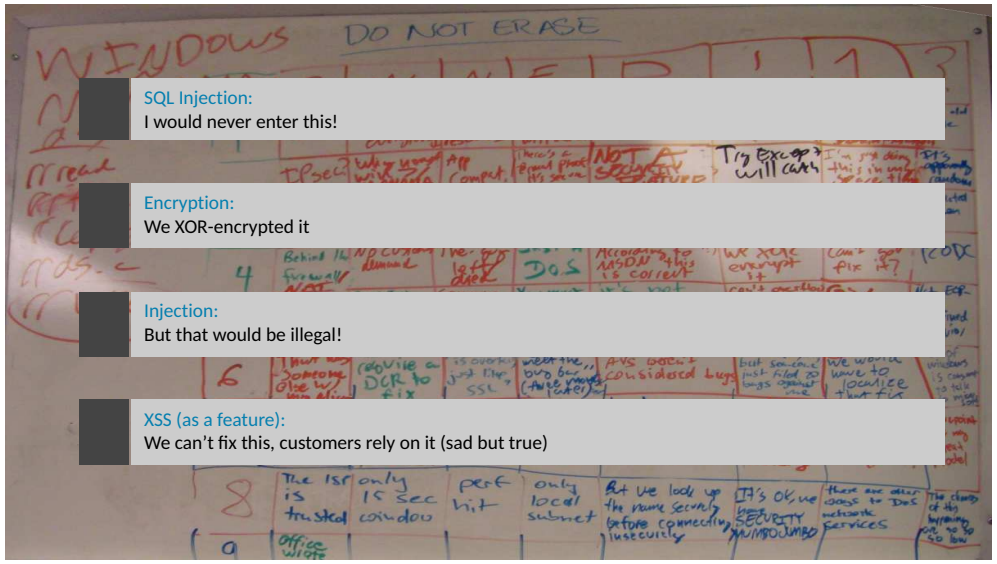
OWASP AppSec Europe London 2nd-6th July 2018

Usable Security for Developers: A Nightmare

Achim D. Brucker | @adbrucker

Introducing the SDLC: View of the security experts

- The whiteboard is from the Microsoft's security team
- I confess, I am guilty too:
  We also had a board with "embarrassing developers quotes"



SQL Injection:
I would never enter this!

Encryption:
We XOR-encrypted it

Injection:
But that would be illegal!

XSS (as a feature):
We can't fix this, customers rely on it (sad but true)

OWASP AppSec Europe London 2nd-6th July 2018

Usable Security for Developers: A Nightmare

Achim D. Brucker | @adbrucker

Introducing the SDLC: View of the developers

- Experience security as
  "The Department of No"
- Confronted with a strange & complex language
  (there are over 1024 CWEs – and counting)

Example of unfriendly APIs: Buffer overflow

```
> man gets
GETS(3S)                                                GETS(3S)

NAME
     gets, fgets - get a string from a stream

SYNOPSIS
     #include <stdio.h>

     char *gets(s)
     char *s;

DESCRIPTION
     Gets reads a string into s from the standard input
     stream stdin. The string is terminated by a newline
     character, which is replaced in s by a null character.
     Gets returns its argument.
```

Let's travel back in time

- Unix V7 (1979)
- Reading strings
- Gets returns a string of arbitrary length

Is there a secure use of gets?

Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker
OWASP AppSec Europe London 2nd-6th July 2018

Example of unfriendly APIs: Buffer overflow

- Wait, let's check the man page on a modern Unix/Linux:

```
NAME
  gets - get a string from standard input (DEPRECATED)


BUGS
  Never use gets(). Because it is impossible to tell without knowing the
  data in advance how many characters gets() will read, and because
  gets() will continue to store characters past the end of the buffer,
  it is extremely dangerous to use.  It has been used to break computer
  security.  Use fgets() instead.
```

Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker
OWASP AppSec Europe London 2nd-6th July 2018

Example of unfriendly APIs: Buffer overflow

- OK, that's sounds easy: Use `fgets(buf, n, stdin)` instead of `gets(buf)`:

```
void f() {
    char buf[20];
    gets(buf)fgets(buf,20,stdin) // NOT: gets(buf);
}
```

- Is this now secure? No, fgets does **not** always null-terminate
- we need to manually null terminate the buffer (and reserve space for the null character)

```
void f() {
    char buf[21];
    fgets(buf,20,stdin);
    buf[20]='\0';
  }
```

- C-Programming has a lot in comming with (insurance) contracts: allways read the small print

Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker
OWASP AppSec Europe London 2nd-6th July 2018

Example of unfriendly APIs: Error handling

> "Most OpenSSL functions will return an integer to indicate success or failure. Typically a function will **return 1 on success or 0 on error**. All return codes should be checked and handled as appropriate. Note that not all of the libcrypto functions return 0 for error and 1 for success. There are exceptions which can trip up the unwary. For example **if you want to check a signature with some functions you get 1 if the signature is correct, 0 if it is not correct and -1 if something bad happened** like a memory allocation failure." (OpenSSL)

- Recall the common C convention:
  - 0 indicates success
  - any non-zero value indicates failure

Example of unfriendly APIs: Error handling

- Which one is correct:
  1. Consider

```
if (some_verify_function())
  /* signature successful *?
```

  2. Consider

```
if ( 1 != some_verify_function())
  /* signature successful *?
```

  3. Consider

```
if ( 1 == some_verify_function())
  /* signature successful *?
```

- The last one is correct

Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker
OWASP AppSec Europe London 2nd-6th July 2018

Example of unfriendly APIs: The Java 8 Crypto API

Just a nightmare:

- Many configurations to choose from
  - algorithm
  - mode of operation
  - padding scheme
  - right keys and sizes
  - …
- Most ciphers are oudated/broken. Only two can still be recommended
  - AES (symmetric)
  - RSA (asymmetric)
- Many providers use insecure defaults (e.g., ECB mode)

Using the Java crypto API, is already hard for somebody who understands crypto …

Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker
OWASP AppSec Europe London 2nd-6th July 2018

Example of unfriendly APIs: XSS (Java)

- Most Web Frameworks for Java do not provide input/output encoding as default
- Developers need to include third party encoding libraries
  (e.g., OWASP Java Encoder: https://github.com/OWASP/owasp-java-encoder)
- **and** add calls to the encoder manually:

```
PrintWriter out = ....;
out.println("<textarea>"+Encode.forHtml(userData)+"</textarea>");
```

- You need to insert the **right** (there are many) encoder **each time**.

Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker
OWASP AppSec Europe London 2nd-6th July 2018

Common mitigations

- Provide training
  - Do we really expect that our developers understand all these details?
- Write (coding) guidelines
  - Guidelines without tool support are (mostly) worthless
- Use generic application security testing tools
  - without configuration, these tools are prone to both high false-positive rates and high false-negative rates
  - many tools are developed for security experts (and not for developers)
  - penetration tests

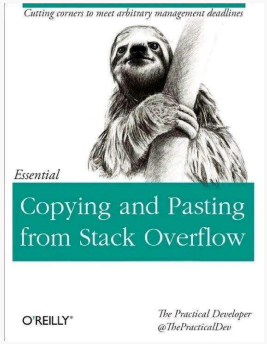
In their generality, these actions are often not very effective!

Security experts and developers need to work together to achieve the common goal: secure software!
(Disclaimer: security experts might need to learn how to code)!

Think positive: security enables developers to produce high-quality and secure software!

- Start early in the development:
  - Select frameworks and/or programming languages that are secure by design
  - Develop custom APIs-Wrappers that are easy to use and require only little security knowledge
  - To consider
    - Configure your DAST/IAST/SAST tool to support your custom APIs
    - In the fix recommendations of your DAST/IAST/SAST tool, point developers to the recommended frameworks
    - If you develop APIs, make your examples secure by default

OWASP AppSec Europe London 2nd-6th July 2018
Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker



If you do not support your developers, they will seek for help elsewhere!

OWASP AppSec Europe London 2nd-6th July 2018
Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker

Let's close with a good example: Modern Rails

- Modern versions of Rails are pretty secure by default
- Input/output encoding is enabled by default and, in exceptional cases, needs to be disabled explicitly:

```
<%= account.balance.html_safe % >
```

(one can argue, if `html_safe` is a good name denoting un-sanitized (non-trusted) channels)

- Suddenly, a simple `grep` becomes a powerful static analysis tool

OWASP AppSec Europe London 2nd-6th July 2018
Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker

Call for action

Let's build framework and APIs are easy to use securely!

Call for action

Let's build framework and APIs are easy to use securely!

Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker

Call for action

Let's build framework and APIs are easy to use securely!

Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker

Call for action

Let's build framework and APIs are easy to use securely!

Usable Security for Developers: A Nightmare
Achim D. Brucker | @adbrucker

Thank you for your attention!
Any questions or remarks?

Contact:

Dr. Achim D. Brucker
Department of Computer Science
University of Sheffield
Regent Court
211 Portobello St.
Sheffield S1 4DP, UK

a.brucker@sheffield.ac.uk
@adbrucker
https://de.linkedin.com/in/adbrucker/
https://www.brucker.ch/
https://logicalhacking.com/blog/

Document Classification and License Information