The Abridged History of Application Security


MANICODE
SECURE CODING EDUCATION
extremely
The Abridged History of Application Security
Things are Getting a Lot Better

Jim Manico

jim@manicode.com

LOCO MOCO

twitter.com/manicode

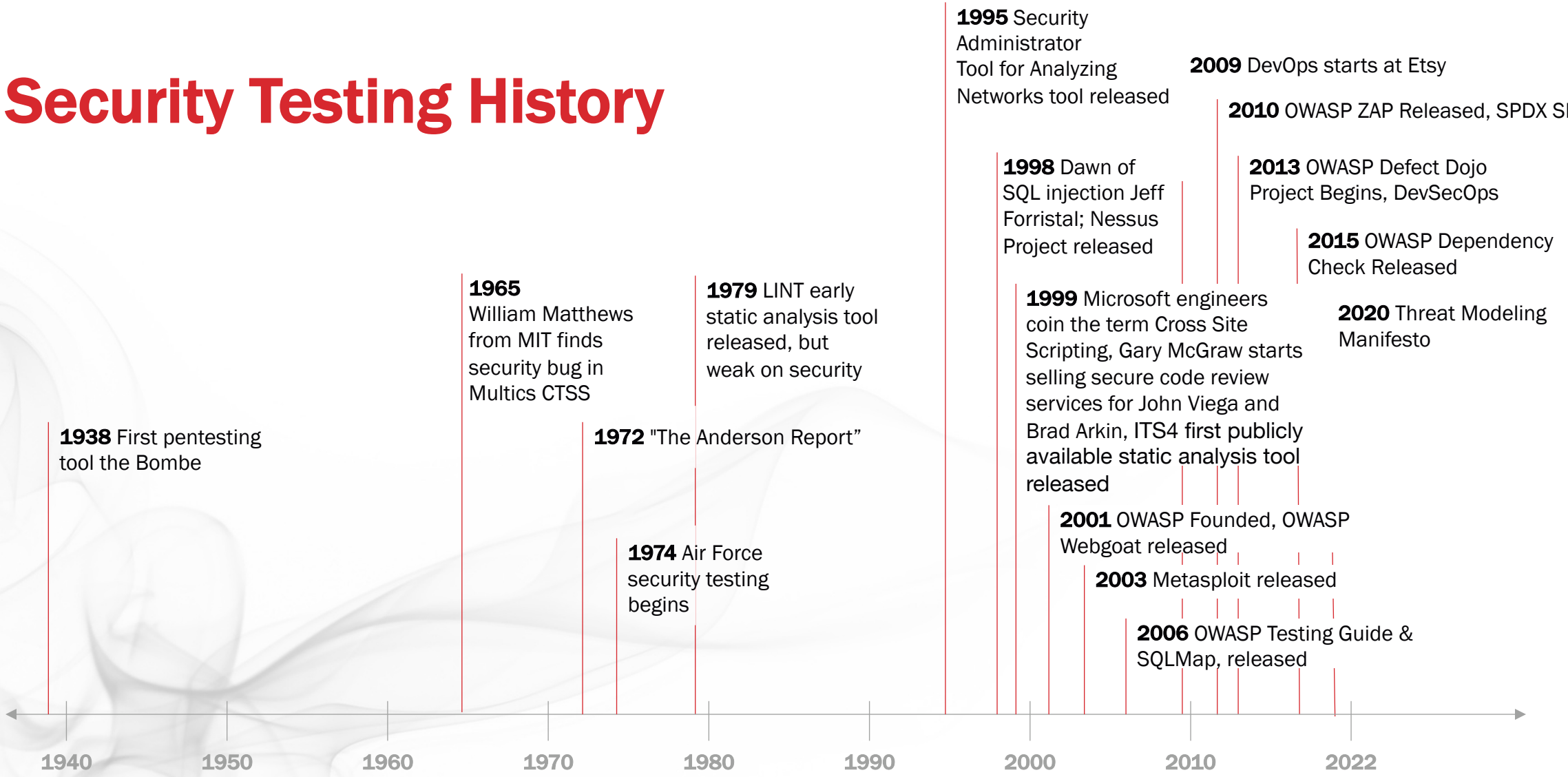
- Former OWASP Global Board Member
- Co-Founder of LocoMocoSec
- Founder/CEO of Manicode Security
- 25+ years of software development experience
- Author of "Iron-Clad Java, Building Secure Web Applications" from McGraw-Hill/Oracle-Press
- OWASP Project Leader
  - OWASP ASVS Standard
  - OWASP Cheat Sheet Series
  - OWASP Java Encoder / HTML Sanitizer
  - OWASP Top 10 Proactive Controls

InfoSec Dark Ages

- October 1967 Task Force
- February 1970 R-609 Published
- October 1975 R-609 Declasified

IS MESSED UP!

Security Testing History



2023

- Security Testing Integrated Into GitHub
- DevSecOps with SAST, DAST, SCA and IAST
- Pentesting/AppSec Services Still Expensive (not enough professionals out there)
- OWASP MTSG being used for compliance by some of the largest software companies on the planet

HTTP/S History

1994 Netscape creates initial version of HTTPS

1999 TLS 1.0 released

2006 TLS 1.1 released

2008 TLS 1.2 released

2009 SSLLabs released to public as a way to verify security configuration of HTTPS websites

2010 Chrome starts to HSTS preload some sites

2011 Forward secrecy live in modern browsers

2013 TLS 1.2 live in modern browsers

2015 Lets Encrypt starts! Usable Cryto Era Begins! MS's.

2016 Overt ½ the web HTTPS; Chrome 51 defaults to HTTP/2 and only allows TLS

2017 CAA becomes mandatory, Google says goodbye to HPKP

2018 Let's Encrypt offers wildcard; TLS 1.3 live in FireFox and Chrome; CT required for new certs

2021 PKI Consortium Formed from the ashes of the Certificate Authority Security Council, Chrome 91 defaults to HTTPS

1995 2010 2015 2022

2021 Chome 90 Defaults to HTTPS

Chromium Blog

News and developments from the open source browser project

A safer default for navigation: HTTPS

Tuesday, March 23, 2021

Starting in version 90, Chrome's address bar will use *https://* by default,

June 2023

- https://transparencyreport.google.com/
- 97% or more pages loaded by Chrome on MacOS are HTTP/S
- March 14, 2015 43% or more pages loaded by Chrome on MacOS is HTTP/S

SSL Report: manicode.com (198.199.114.91)

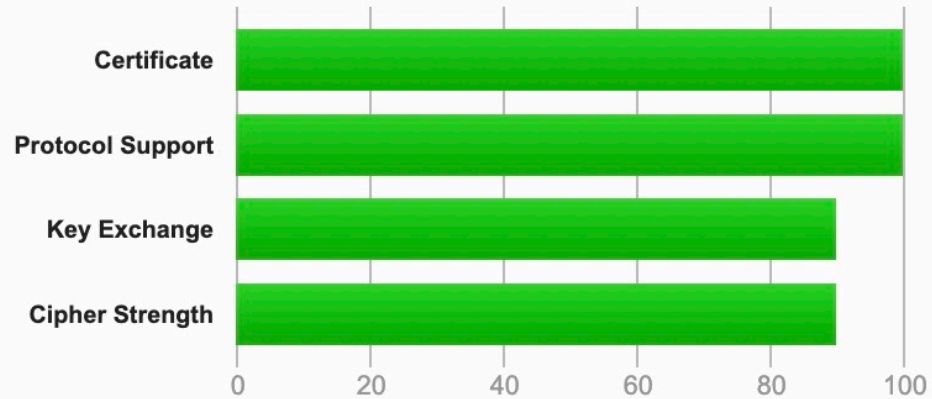
Assessed on: Wed, 26 Oct 2022 17:55:56 UTC | Hide | Clear cache

Scan Another »

Summary

Overall Rating

A+



Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This site works only in browsers with SNI support.

This server supports TLS 1.3.

HTTP Strict Transport Security (HSTS) with long duration deployed on this server. MORE INFO »

DNS Certification Authority Authorization (CAA) Policy found for this domain. MORE INFO »

Password History

1961 First password based system created at MIT's CTSS led by Fernando Corbató.

1966 MIT PhD candidate Allan Scherr requests print out of master password file in CTSS and becomes the first password hacker

1970s Crypt(3) released in Unix uses old M-209 code from WW2 (unix up to 6th edition)

1978 Crypt(3) released in Unix now DES based (7th edition); first stretching, salting and password policy

1980 passw limite

1991 MD5 message-digest algorithm - 128-bit hash value

1994 FreeBSD MD5 based crypt(3) with 1000

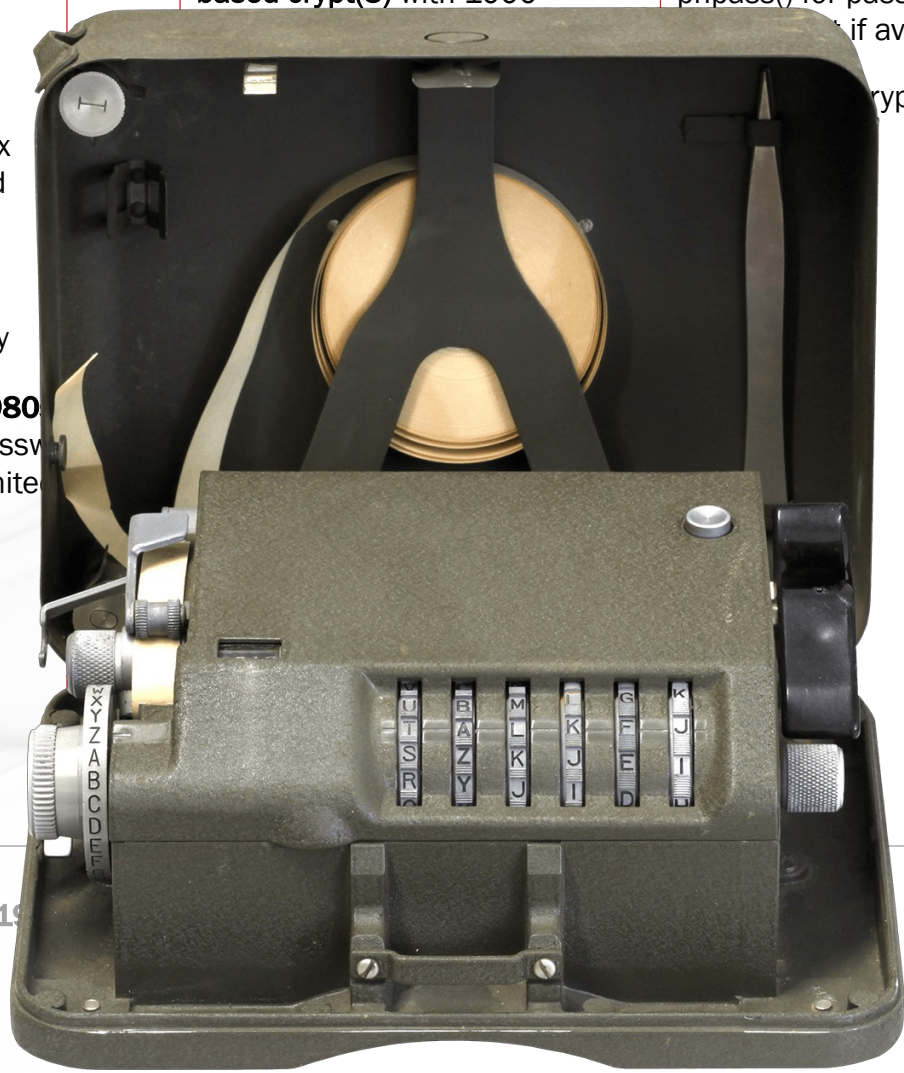
2007 PHP apps start using phpass() for password storage, if available

rypt published

2015 Argon2id wins password hashing competition https://password-hashing.net/

2016 Dr. Akhawe from Dropbox publishes password storage strategy

2019 PHP 7.3 natively supports Argon2id



1960 1965 1970 1975 1980 19 2015 2022

M-209B, cryptography collection of the Swiss Army headquarters. Photographed by Rama, Wikimedia Commons, licensed under CeCILL v2 and CC-BY-SA-2.0-FR

How Dropbox securely stores your passwords

Devdatta Akhawe | September 21, 2016

It's universally acknowledged that it's a bad idea to store plain-text passwords. If a database containing plain-text passwords is compromised, user accounts are in immediate danger. For this reason, as early as 1976, the industry standardized on storing passwords using secure, one-way hashing mechanisms (starting with Unix Crypt). Unfortunately, while this prevents the direct reading of passwords in case of a compromise, all hashing mechanisms necessarily allow attackers to brute force the hash offline, by going through lists of possible passwords, hashing them, and comparing the result. In this context, secure hashing functions like SHA have a critical flaw for password hashing: they are designed to be fast. A modern commodity CPU can generate millions of SHA256 hashes per second. Specialized GPU

2023

- Argon2id supported everywhere
- OWASP Cheatsheet has very surgical advice on the use of Argon2id

Rather than a simple work factor like other algorithms, Argon2id has three different parameters that can be configured. Argon2id should use one of the following configuration settings as a base minimum which includes the minimum memory size (m), the minimum number of iterations (t) and the degree of parallelism (p).

- m=37 MiB, t=1, p=1
- m=15 MiB, t=2, p=1

Both of these configuration settings are equivalent in the defense they provide. The only difference is a trade off between CPU and RAM usage.

Jim Manico
@manicode

I don’t like bcrypt and that’s a fact
Argon2id is where it’s at
72 byte limit can kiss my ass
Low ram usage? No thanks I’ll pass

Jeremi M. Gosney
@jmgosney

Replying to @manicode

Argon's a KDF and that's no cap
If you're doing real time auth
You need bcrypt in your app
Memory hardness has no meat, it's all fat
For resistance to acceleration
Cache hardness is where it's at

Jeremi Gosney · 1st

Principal Software Architect

Austin, Texas, United States · **Contact info**

Phobos Group

Jim Manico @manicode · 1d

Argon2id won the password competition

For password storage you need a little revision

bcrypt truncates and that's a fact

it's a shitty limit and is easier to attack

Argon2id has better password cracking resistance

I'm here to help, here is a reference

- https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

Jeremi M. Gosney @jmgosney · 1d

Replying to @manicode

Which one of us is the cracker?
It seems you've forgot.
And a Hashcat developer?
I think you are not.
And the password competition?
I was a judge for that too.
With all these credentials
who is OWASP to argue?
😮

Jim Manico
@manicode

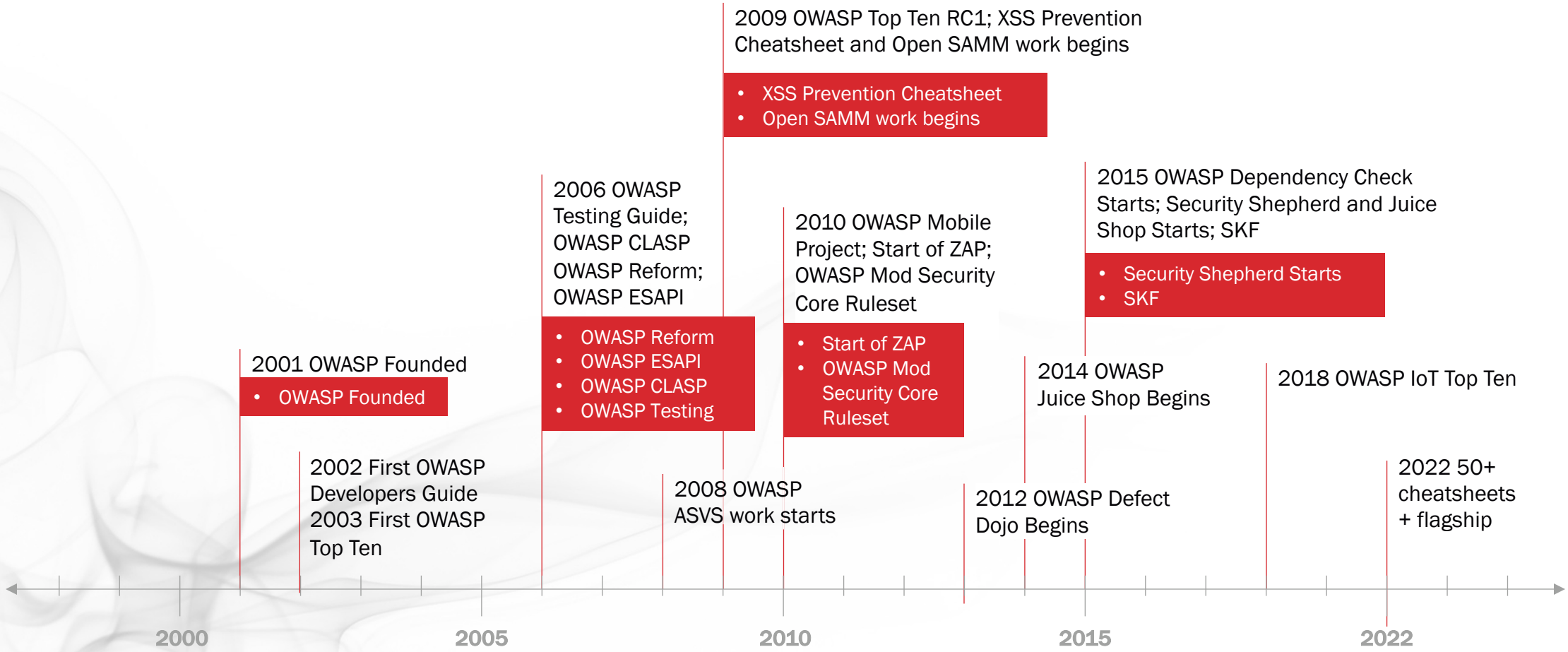
The other judges outvoted you
And 1+1 still equals 2

VICTORY

You can talk  about PHP

But it natively supports Argon2id

OWASP Project History



Flagship Projects 

- OWASP Amass
- OWASP Application Security Verification Standard
- OWASP AppSensor
- OWASP Cheat Sheet Series
- OWASP Cloud-Native Application Security Top 10
- OWASP CSRFGuard
- OWASP CycloneDX
- OWASP Defectdojo
- OWASP Dependency-Check
- OWASP Dependency-Track
- OWASP Juice Shop
- OWASP Mobile Security Testing Guide
- OWASP ModSecurity Core Rule Set
- OWASP OWTF
- OWASP SAMM
- OWASP Security Knowledge Framework
- OWASP Security Shepherd
- OWASP Top 10 Low-Code/No-Code Security Risks
- OWASP Top Ten
- OWASP Web Security Testing Guide
- OWASP ZAP

XSS History

1999 Microsoft engineers coin the term Cross Site Scripting

2002 HTTPOnly supported in IE6

2004 The Content Restrictions Standard was proposed by Robert Hansen which became Content Security Policy

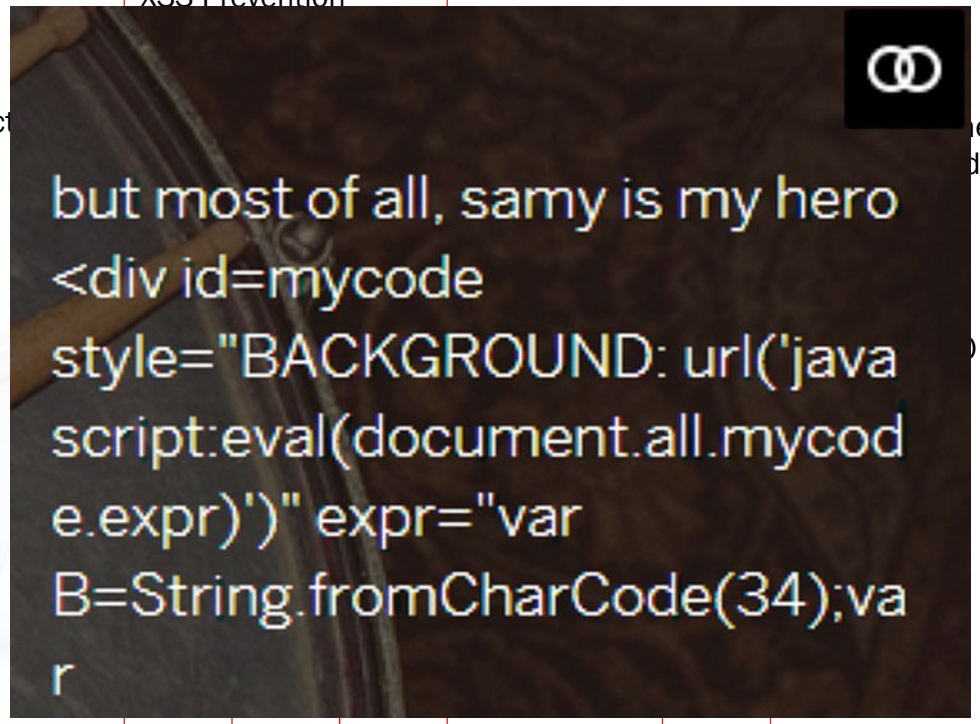
2005 But most of all, samy is my hero

2005 Amit Klein first DOM XSS publication

2006 OWASP Reform project starts

2009 OWASP XSS Prevention

2012 CSP 1.0 published



2022 Trusted Types Live in Chrome, React

2000 2005 2010 2015

Trusted Types help prevent Cross-Site Scripting

☆☆☆☆☆

TL;DR

We've created a new experimental API that aims to prevent DOM-Based Cross Site Scripting in modern web applications.

By Krzysztof Kotowicz

Software Engineer in the Information Security Engineering team at Google

We're currently working on the specification and implementation details for this API. We'll keep this post updated

https://developer.mozilla.org/en-US/docs/Web/API/HTML_Sanitizer_API

mdn web docs References Guides MDN Plus Theme

References > Web APIs > HTML Sanitizer API

Related Topics

HTML Sanitizer API

▼ Interfaces

Sanitizer

HTML Sanitizer API

Experimental: This is an experimental technology

Check the Browser compatibility table carefully before using this in production.

Secure context: This feature is available only in secure contexts (HTTPS), in some or all supporting browsers.

The **HTML Sanitizer API** allow developers to take untrusted strings of HTML and

2023

- AutoEscaping templates the norm
- CSP3 with strict-dynamic is easier to deploy and is live in all major browsers
- Trusted Types available in many frameworks
- Being a bit of a ***punk*** on Twitter helps encourage Apple to deliver CSP3 in Safari

Jim Manico
@manicode

I believe that Safari's lack of CSP3 support and similar W3C standards is a reason to boycott and stop using it.

I now see Safari as a browser that primarily impedes the secure web.

Edge, Chrome and FireFox all support CSP3 well.

Why not Safari? I'm open to counter opinions.

4:34 PM · Aug 25, 2021 · Twitter for iPhone

Philippe De Ryck @PhilippeDeRyck · Aug 25, 2021

That holds for a lot of people, but the market share is not insignificant. @usefathom has Safari at 8% for my site this year.

I'm a happy @brave user on computers, but use Safari on mobile...

Browser	Uniques
Chrome	74%
Firefox	10%
Safari	8%
Edge	6%
Mozilla	1%

Jim Manico @manicode · 8/26/21

I am trying not to harass John he is good people and I believe this is above his pay-grade.

1 5

John Wilander 🇺🇦 @johnwilander · 8/26/21

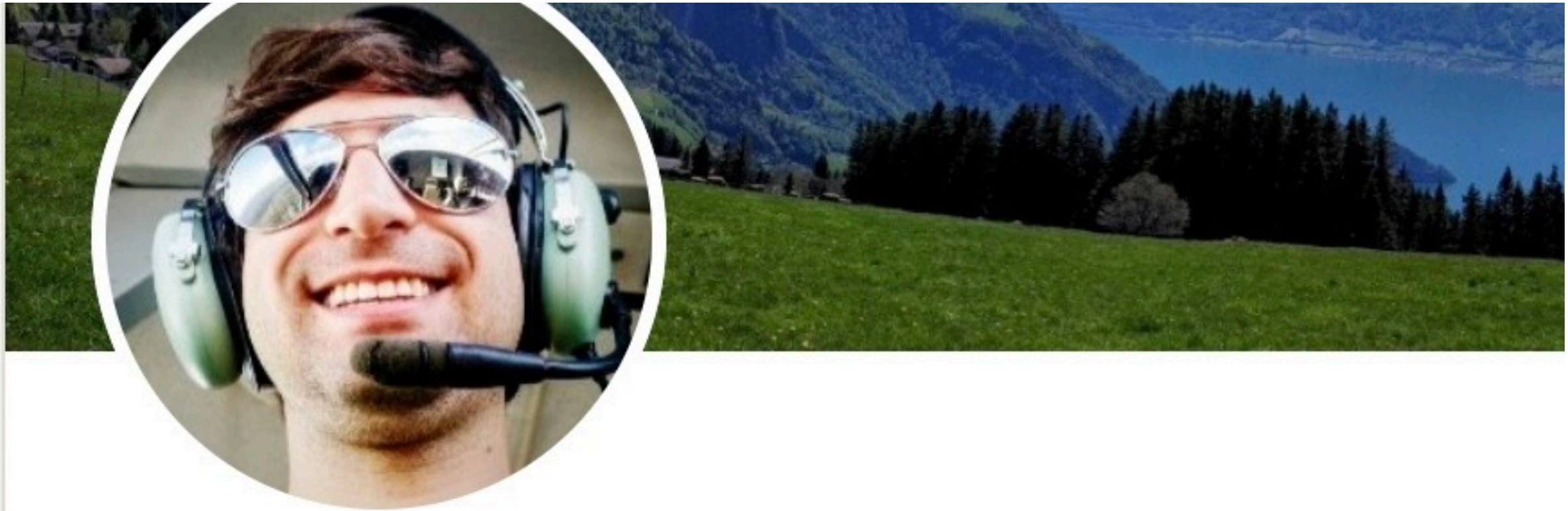
Don’t worry. I want to see CSP3 in WebKit asap and always forward good faith feedback to the team. If you want to talk directly to the person in charge, he’s @othermaciej.

2 10

John Wilander (He/Him) · 1st

Manager WebKit Security & Privacy and hacker fiction novelist

San Francisco Bay Area · **Contact info**



Lukas Weichselbaum · 2nd

Senior Staff Information Security Engineer at Google

Zurich, Switzerland · **Contact info**

PEOPLE ›

Lukas Weichselbaum

About

Lukas Weichselbaum is a Staff Information Security Engineer at Google with 10+ years of industry experience who frequently speaks at international infosec and developer conferences.

He's passionate about securing Web applications from common Web vulnerabilities and leads the Google-wide CSP adoption effort. Lukas also co-authored the **CSP3 W3C specification** and launched **CSP Evaluator**, a tool for developers and security experts to check if a Content Security Policy serves as a strong mitigation against XSS attacks.

Before joining Google, Lukas worked as a Security Consultant and graduated from Vienna University of Technology in Austria where he researched dynamic analysis of Android malware and founded Andrubis - one of the very first large scale malware analysis platforms for Android applications.

Lukas Weichselbaum @we1x · 8/26/21

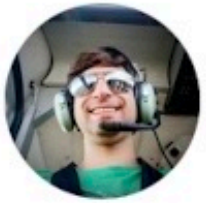
Would you accept pull requests? 🙂

3 5

John Wilander 🇺🇦 @johnwilander · 8/26/21

Absolutely! WebKit is open source and it's not just Apple building web platform features for it. E.g. Igalia and Sony do tons of work. We haven't expressed opposition to CSP3 beyond 1) sad it isn't backwards compatible with CSP2, and 2) CSP too complex for devs in general.

2 3 9



Lukas Weichselbaum @we1x · 8/26/21

Thanks John! Was this opposition about 'strict-dynamic' or CSP3 in general?

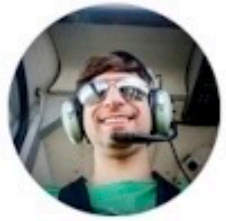
In our experience, 'strict-dynamic' makes deployment of an effective CSP easier/possible and is also backward compatible. Support in Safari could even simplify the policy as we could drop fallbacks.

1 1 3

John Wilander 🇺🇦 @johnwilander · 8/26/21

We've said we would like CSP to be split up. At least fork off 'script-policy' and give it recognizable directives like 'same-origin' and 'same-site'. Along those lines. A dev reading the CSP spec for the first time is unlikely to deem it approachable.

3 4



Lukas Weichselbaum @we1x · 8/26/21

Thanks for explaining. I definitely agree that it would be good if we could have a simpler version of CSP in the future. I also agree that it would make sense to fork off the scripting part.

Regarding strict-dynamic PRs, I need to talk with some folks, but will reach out again 🙂

Jim Manico
@manicode

A important correction on this topic: **Kate Cheney from Apple is implementing CSP3** in WebKit. See for instance: trac.webkit.org/changeset/2831... - “CSP: Implement 'strict-dynamic' source expression" landed in the open source tree about a month ago **from** Kate! How awesome!

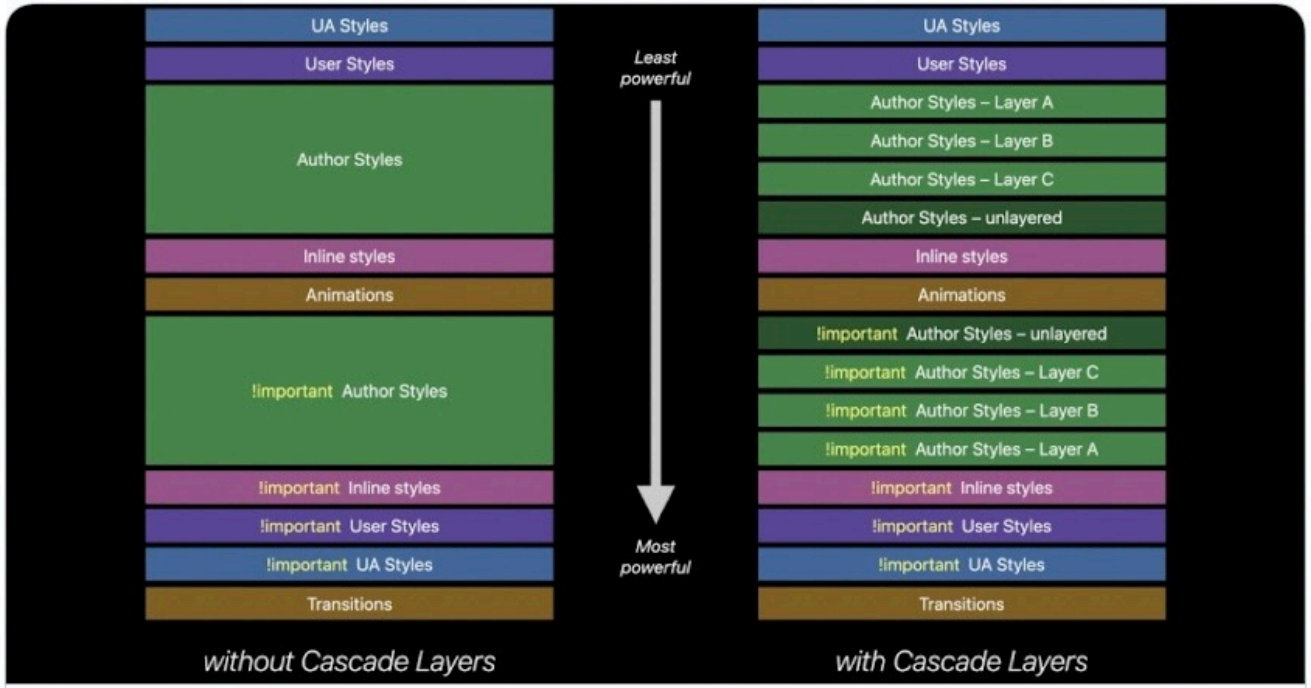
> **Liran Tal | Node4Shell** @liran_tal · 10/25/21
>
> Let the browser war begin! 🔫
>
> Just kidding, it's already started.
> Here's @manicode's take: twitter.com/snyksec/status...

8:44 AM · 10/26/21 · Twitter for Mac

John Wilander 🇺🇦 @johnwilander · 3/14/22

Replying to @manicode

"**WebKit in Safari 15.4** improves support for Content Security Policy Level 3 (...) New support for 'strict-dynamic', 'unsafe-hashes', and 'report-sample' source expressions give developers more flexibility." webkit.org/blog/12445/new... **Happy to deliver** you **this** message, my friend!



webkit.org
New WebKit Features in Safari 15.4

1 8 21

Jim Manico
@manicode

Replying to @johnwilander

John, this truly makes me beam with joy. Thank you and your team for their hard work in delivering this critical security standard in Safari!

1:43 AM · 3/15/22 · Twitter for Mac

CSP 3 EVERYWHERE

2023

- AutoEscaping templates the norm
- CSP with strict-dynamic is easier to deploy and live in all major browsers
- Trusted Types has made its way into frameworks
- DOMPurify becomes a web standard and is appearing natively in browsers

AppSec is Global 230+ OWASP Chapters Worldwide

OWASP® Foundation

Members	Groups	Countries
111,621	233	70

The Future of AppSec

2023

- AI is alive
- Federal Information Security Modernization Act of 2022
- NIST 800-63 and 800-53
- Usable Crypto Everywhere
- Microservice Security does not need to sacrifice performance
- GDPR/CCPA
- SBOM and Supply Chain Focus (log4shell)
- Threat Modeling is maturing and becoming automated (Top10)
- RUST for the Linux Kernel
- Open Policy Agent
- DevSecOps
- SemGrep and CodeQL

The Future of AppSec is You

Have a great **conference**!