OWASP
Open Web Application
Security Project

Application Security Verification Standard 4.0

Josh Grossman, OWASP ASVS Project co-leader

September 2019 – OWASP Helsinki

WHO
AM I?

Josh Grossman

- Over 10 years of IT Security, IT Risk and development experience
- Application Security Consultant and Head of Security Services for AppSec Labs
- Consulting for clients internationally and locally

Josh Grossman

Josh Grossman

- Co-leader of the OWASP ASVS Project
- Major Contributor to the OWASP Top Ten Proactive Controls project
- Contributor to the OWASP Top 10 Risks and OWASP JuiceShop projects

What is the ASVS?

- Designed to be an actual application security standard
- Set of leading practices
- Community and Industry Driven

- Completely developed in the open at GitHub
  - Submit issues! Submit PRs! Translate please!

Who is involved?

- Andrew van der Stock, Daniel Cuthbert, Jim Manico, Josh Grossman, Mark Burnett
- Amazing contributors and reviewers such as Abhay Bhargarv, Elar Lang, ossie-git, Ron Perris, Tonimir Kisasondi, Serg Belokamen, Jason Axley, and Adam Caudill

- You

SO WHAT'S NEW?

What’s new

- Now completely written in Markdown
  - Uses MASVS script and CSV generation
  - Easy to translate
  - Easy to determine what changed, when, by whom, and why
- NIST 800-63b compliance
- IoT ASVS Preview Chapter

Modern web applications

- Full support for server-less, responsive applications
- Containers
- API
- DOM
- Templating

CWE all the things

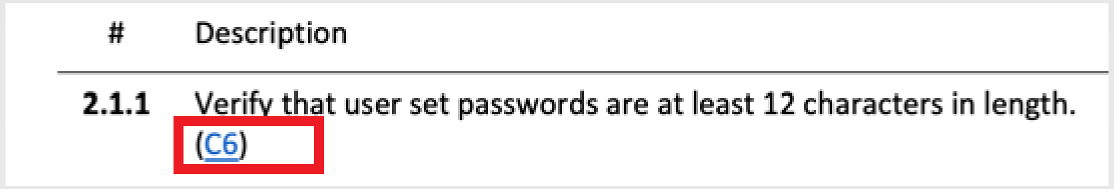
- Most requested feature for the last decade finally delivered

- Let’s talk about CWE for a minute
  - ASVS is a control based standard
  - Weaknesses are not controls
  - CWE is an imperfect mapping, but it’s the mapping we have

- Not every item ended up with a CWE. CWE needs our help

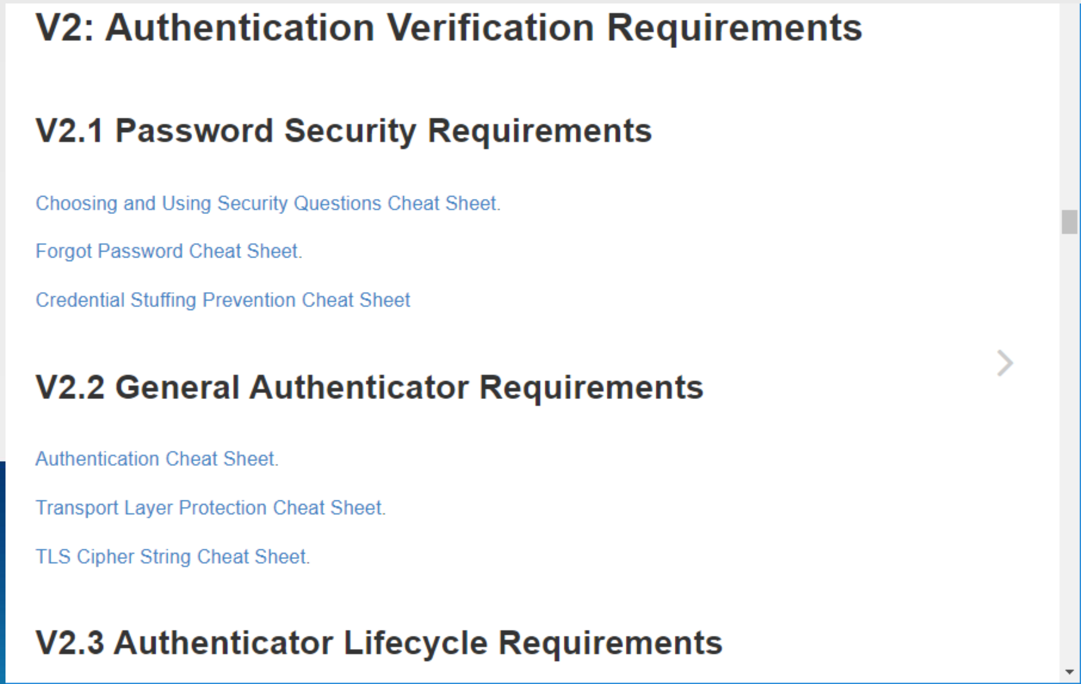
Mappings!

- Mapping ASVS controls to the relevant Proactive Control



#	Description
2.1.1	Verify that user set passwords are at least 12 characters in length. (C6)

- Mapping to cheat sheets project



V2: Authentication Verification Requirements

V2.1 Password Security Requirements

Choosing and Using Security Questions Cheat Sheet.

Forgot Password Cheat Sheet.

Credential Stuffing Prevention Cheat Sheet

V2.2 General Authenticator Requirements

Authentication Cheat Sheet.

Transport Layer Protection Cheat Sheet.

TLS Cipher String Cheat Sheet.

V2.3 Authenticator Lifecycle Requirements

OWASP Open Web Application Security Project

What’s changed

- Basically everything

- Renumbered completely
- Each section is reorganized and re-ordered
- De-duped. Do not omit any section for your level

L1 is the new minimum

- OWASP Top 10 2017 is simply not sufficient

- Level 1 is now completely testable using pentest techniques
- It’s the only level that is completely penetration testable

- Stop penetration testing. Hybrid reviews at least!

PCI DSS 6.5.x

- Yep

- We even included buffer overflows, integer and safer string operations, as well as ensuring that folks compiled code properly!

What’s gone

- Less impactful controls
- Controls that were implemented by one browser or language
- “Since”

- Mobile Chapter (use MASVS)
- IoT Chapter (use IoT Project)

V4: Access Control Verification Requirements

Control Objective

Authorization is the concept of allowing access to resources only to those permitted to use them. Ensure that a verified application satisfies the following high level requirements:

- Persons accessing resources hold valid credentials to do so.
- Users are associated with a well-defined set of roles and privileges.
- Role and permission metadata is protected from replay or tampering.

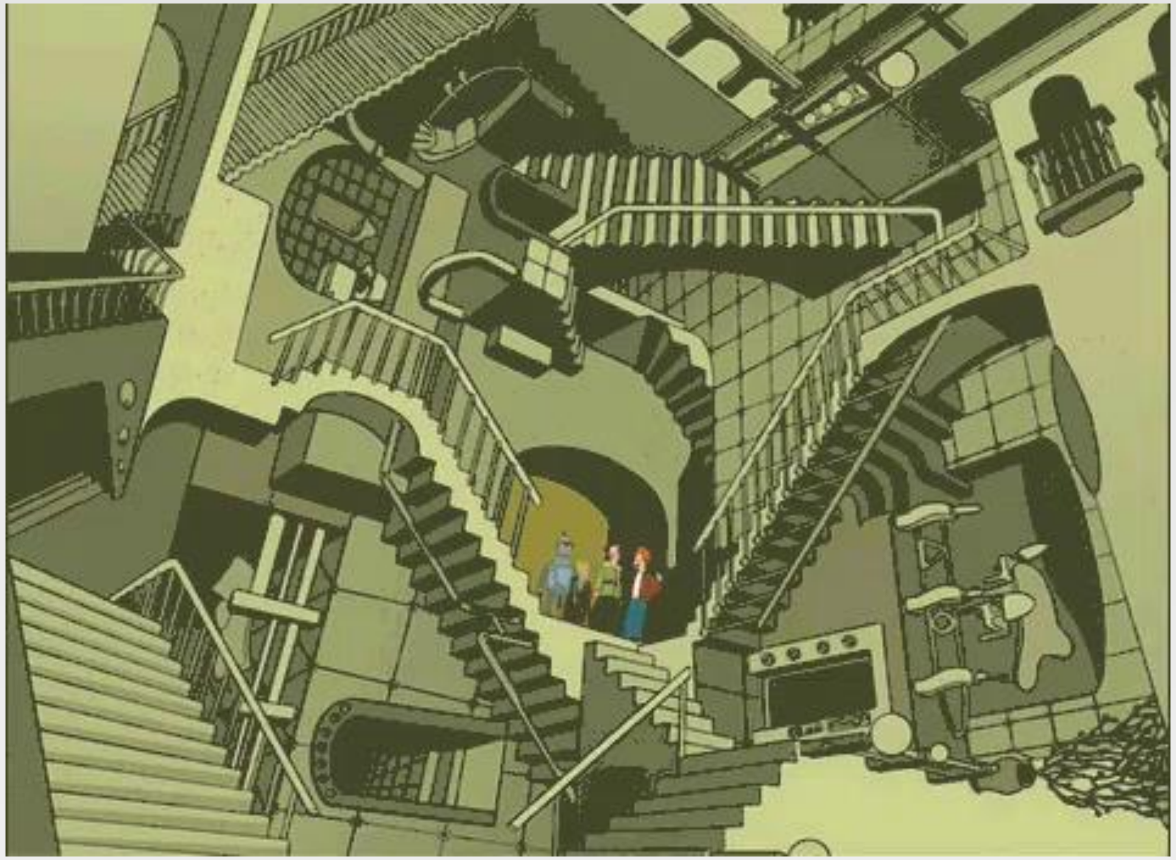
Security Verification Requirements

V4.1 General Access Control Design

#	Description	L1	L2	L3	CWE
4.1.1	Verify that the application enforces access control rules on a trusted service layer, especially if client-side access control is present and could be bypassed.	✓	✓	✓	602
4.1.2	Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.	✓	✓	✓	639
4.1.3	Verify that the principle of least privilege exists - users should only be able to access functions, data files, URLs, controllers, services, and other resources, for which they possess specific authorization. This implies protection against spoofing and elevation of privilege. (C7)	✓	✓	✓	285
4.1.4	Verify that the principle of deny by default exists whereby new users/roles start with minimal or no permissions and users/roles do not receive access to new	✓	✓	✓	276

DETAILED CHANGES

V1 Architecture



V1 Architecture

- Completely revamped

- Replaces “dead” section with something that can produce secure by default software
- Requires Secure Software Development Lifecycle
- Requires threat modeling
- Design and build security in!

- Levels 2 and 3 only

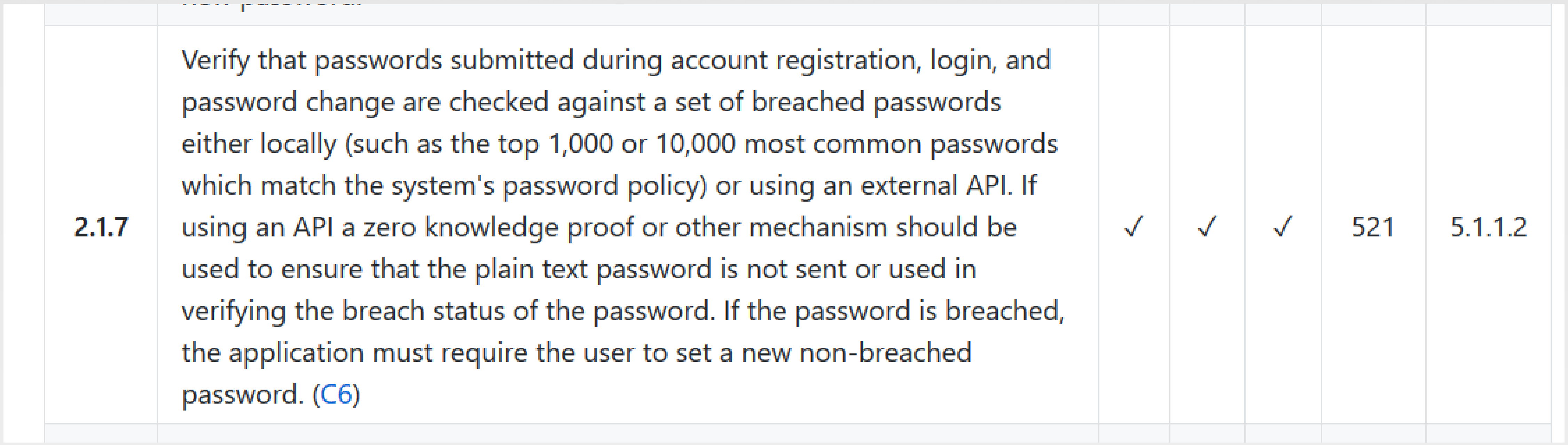
V2 Authentication

IF YOU DON'T WANT ME TO SHOP ONLINE
YOU REALLY SHOULDN'T USE MY NAME FOR ALL YOUR PASSWORDS

V2 Authentication

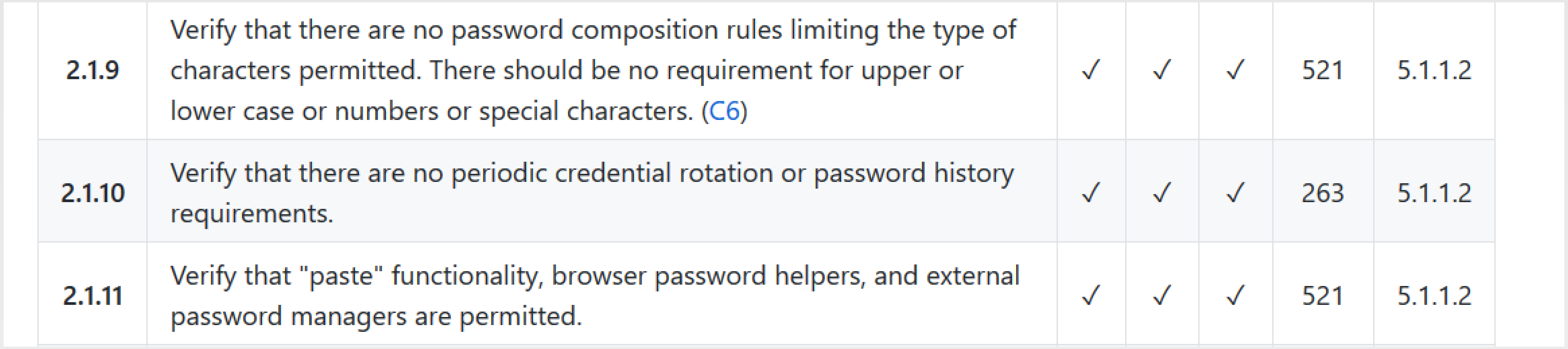
- Aligned with NIST 800-63b
  - Except that we had to go to 12 characters for SFA passwords
- Credential lifecycle from issuance to retirement
  - Credential storage requirements
  - Credential recovery
  - Credentials construction and protection, including credential stuffing
- FIDO support

V2 Authentication



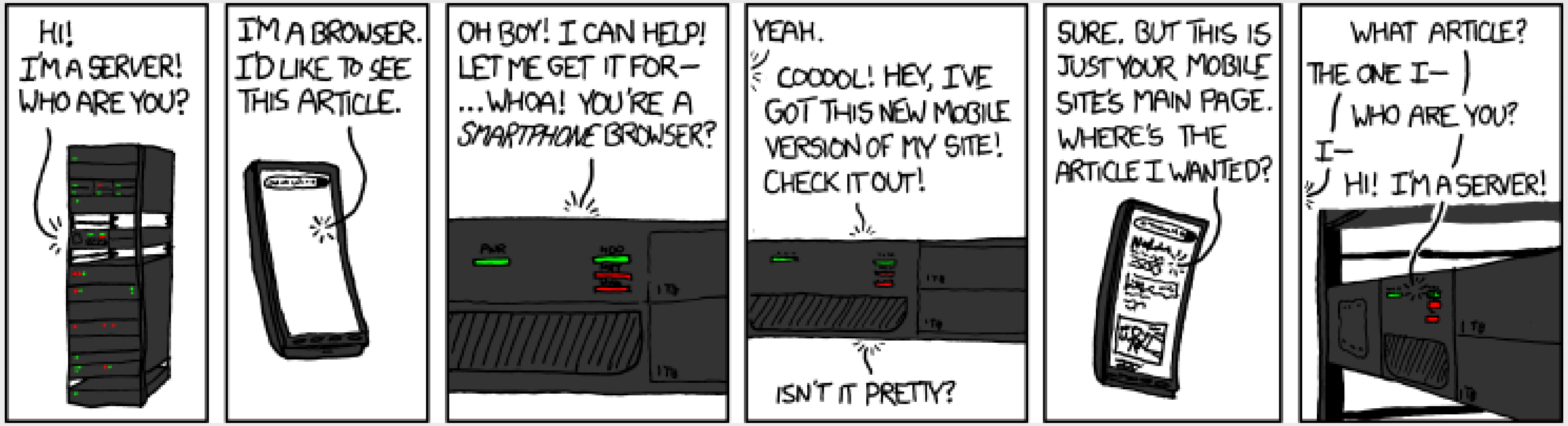
2.1.7	Verify that passwords submitted during account registration, login, and password change are checked against a set of breached passwords either locally (such as the top 1,000 or 10,000 most common passwords which match the system's password policy) or using an external API. If using an API a zero knowledge proof or other mechanism should be used to ensure that the plain text password is not sent or used in verifying the breach status of the password. If the password is breached, the application must require the user to set a new non-breached password. (C6)	✓	✓	✓	521	5.1.1.2

V2 Authentication



2.1.9	Verify that there are no password composition rules limiting the type of characters permitted. There should be no requirement for upper or lower case or numbers or special characters. (C6)	✓	✓	✓	521	5.1.1.2
2.1.10	Verify that there are no periodic credential rotation or password history requirements.	✓	✓	✓	263	5.1.1.2
2.1.11	Verify that "paste" functionality, browser password helpers, and external password managers are permitted.	✓	✓	✓	521	5.1.1.2

V3 Session Management

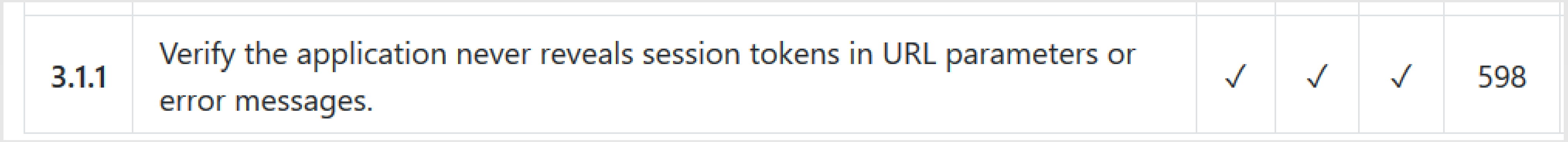

HI! I'M A SERVER! WHO ARE YOU?
I'M A BROWSER. I'D LIKE TO SEE THIS ARTICLE.
OH BOY! I CAN HELP! LET ME GET IT FOR— ...WHOA! YOU'RE A SMARTPHONE BROWSER?
YEAH.
COOOOL! HEY, I'VE GOT THIS NEW MOBILE VERSION OF MY SITE! CHECK IT OUT!
ISN'T IT PRETTY?
SURE. BUT THIS IS JUST YOUR MOBILE SITE'S MAIN PAGE. WHERE'S THE ARTICLE I WANTED?
WHAT ARTICLE?
THE ONE I—
WHO ARE YOU?
I—
HI! I'M A SERVER!

https://www.xkcd.com/869/

V3 Session Management

- Deals with APIs, JWT, token, API, and cookie based session management
- Longer timeouts for many apps for refresh tokens
- Federated session management including full logout
- Half open attack!

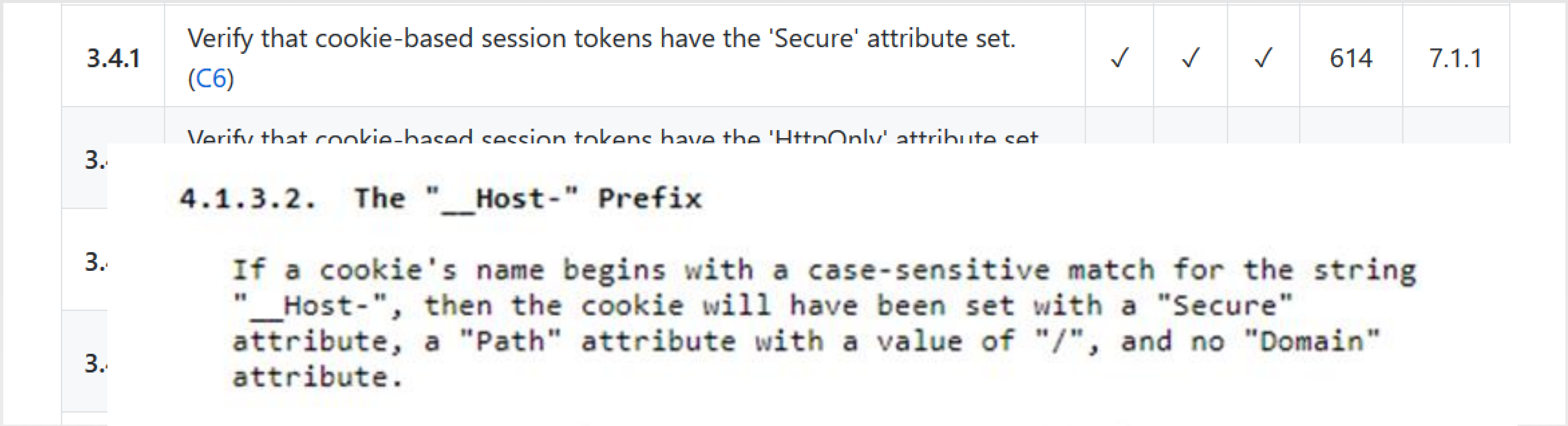
V3 Session Management



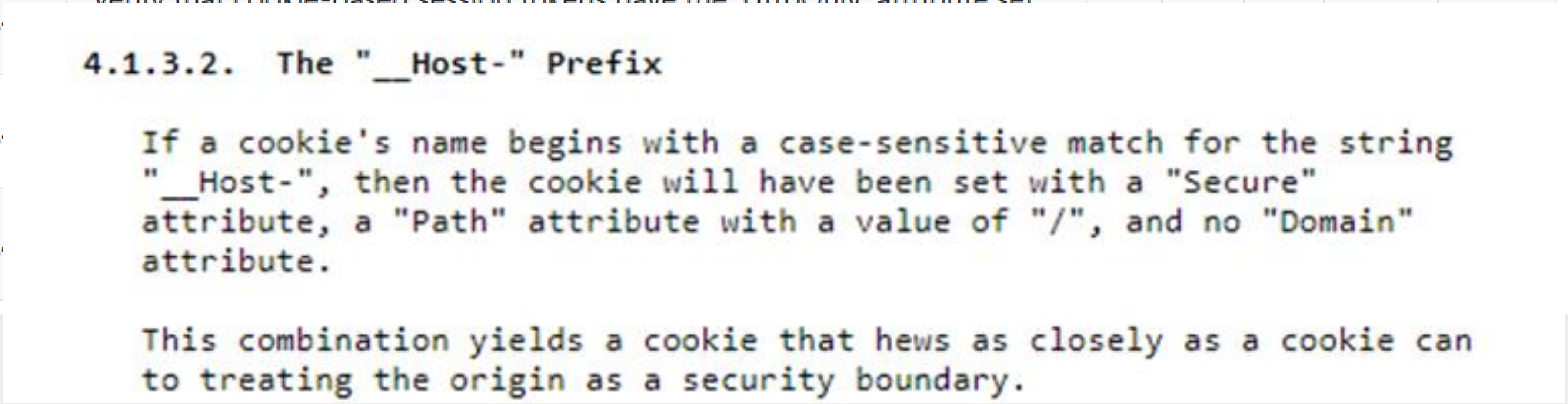
3.1.1	Verify the application never reveals session tokens in URL parameters or error messages.	✓	✓	✓	598

```
Invalid signature. Expected 8Qh5lJ5gSaQylkSdaCIDBoOqKzhoJ0Nutkkap8RgB1Y=

got 8Qh5lJ5gSaQylkSdaCIDBoOqKzhoJ0Nutkkap8RgBOo=
```

V3 Session Management



3.4.1	Verify that cookie-based session tokens have the 'Secure' attribute set. (C6)	✓	✓	✓	614	7.1.1
3.	Verify that cookie-based session tokens have the 'HttpOnly' attribute set					
3.						
3.						

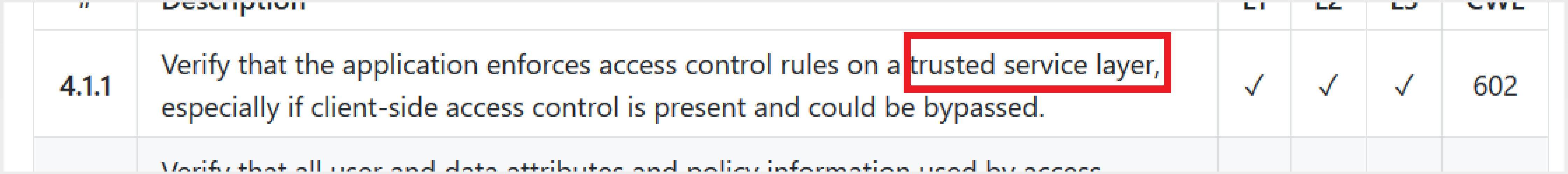


```
4.1.3.2.  The "__Host-" Prefix

   If a cookie's name begins with a case-sensitive match for the string
   "__Host-", then the cookie will have been set with a "Secure"
   attribute, a "Path" attribute with a value of "/", and no "Domain"
   attribute.

   This combination yields a cookie that hews as closely as a cookie can
   to treating the origin as a security boundary.
```

https://tools.ietf.org/html/draft-ietf-httpbis-rfc6265bis-03

V4 Access Control

- Covers API access control security
- Covers serverless and cloud access control (”server-side” is gone)

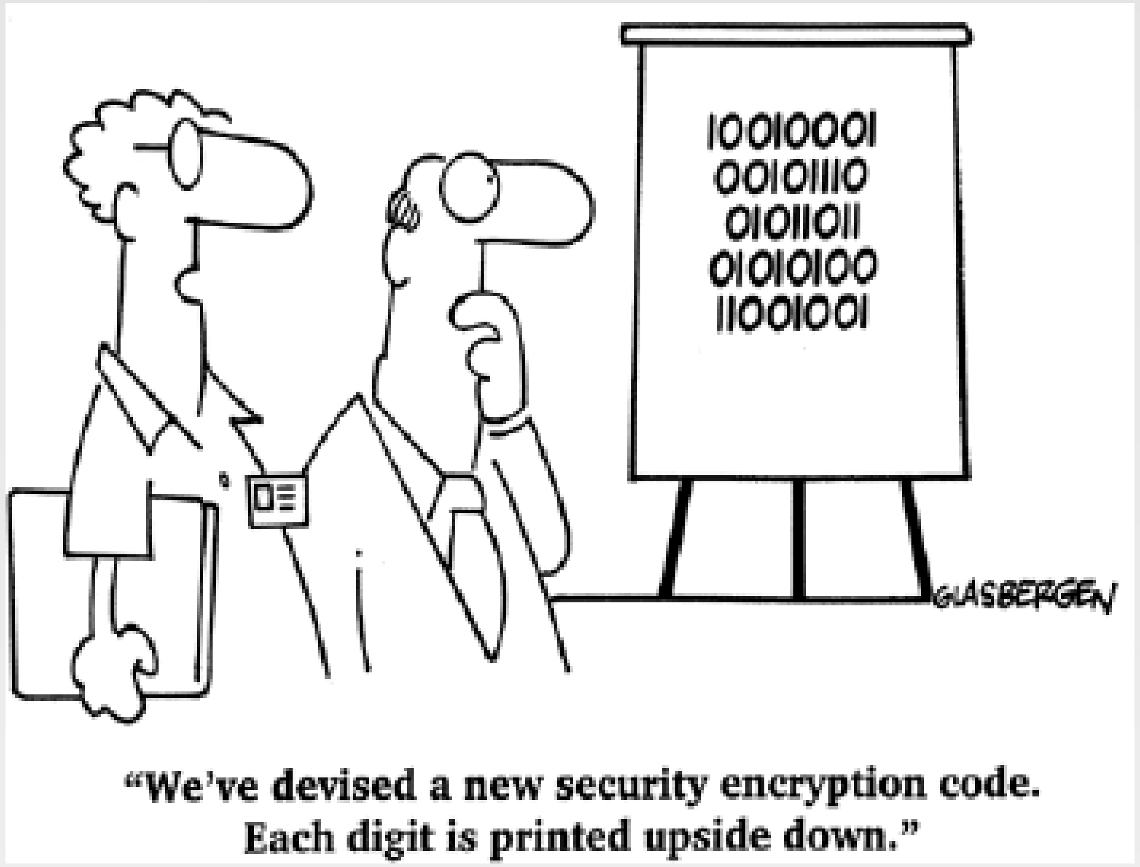


#	Description	L1	L2	L3	CWE
4.1.1	Verify that the application enforces access control rules on a trusted service layer, especially if client-side access control is present and could be bypassed.	✓	✓	✓	602
	Verify that all user and data attributes and policy information used by access				

V5 Validation, Sanitization, and Encoding

- Major revamp
- Divided into easy to consume sections
  - Standardized Input and Output Pipelines
  - Input validation for modern applications and APIs
  - Output encoding is the ~~new~~ old hotness
  - Injection prevention (including XXE and XML attacks)
  - SSRF **is** the new hotness
  - Deserialization **is** the new hotness

V6 Stored Cryptography



"We've devised a new security encryption code. Each digit is printed upside down."

V6 Stored Cryptography

- Data classification – only encrypt the things you need
- Algorithm agility
- Random values
- Secret management (i.e. HSMs, TPMs, OS key stores)

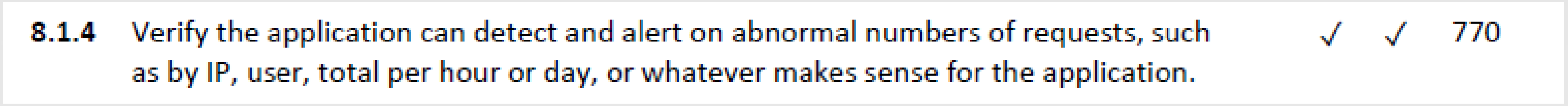
V7 Error handling and Logging

- It’s about reducing the time to detect breaches
- Don’t log everything, but make it easier to detect breaches
- Log protection
- User friendly error handling improvements

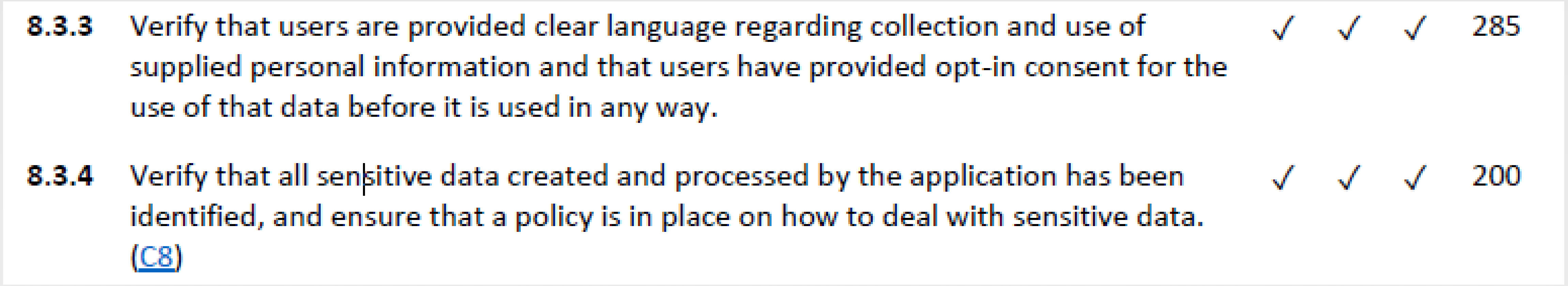
V8 Data Protection

- Protecting the most important data assets
- Sensitive Private Data Protection
  - Covers many technical requirements for GDPR and other privacy laws
- Client side protection (HTML 5 local storage etc)

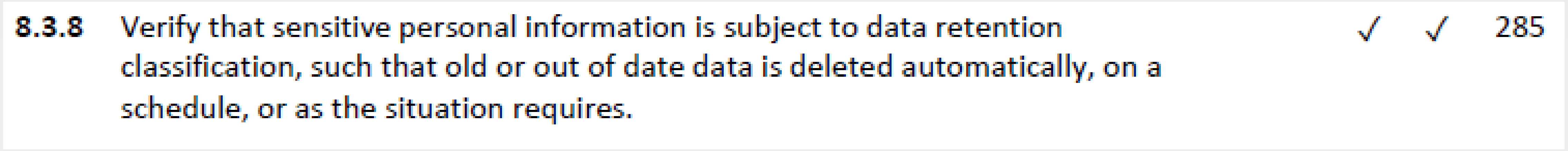
V8 Data Protection



8.1.4	Verify the application can detect and alert on abnormal numbers of requests, such as by IP, user, total per hour or day, or whatever makes sense for the application.		✓	✓	770



8.3.3	Verify that users are provided clear language regarding collection and use of supplied personal information and that users have provided opt-in consent for the use of that data before it is used in any way.	✓	✓	✓	285
8.3.4	Verify that all sensitive data created and processed by the application has been identified, and ensure that a policy is in place on how to deal with sensitive data. (C8)	✓	✓	✓	200



8.3.8	Verify that sensitive personal information is subject to data retention classification, such that old or out of date data is deleted automatically, on a schedule, or as the situation requires.		✓	✓	285

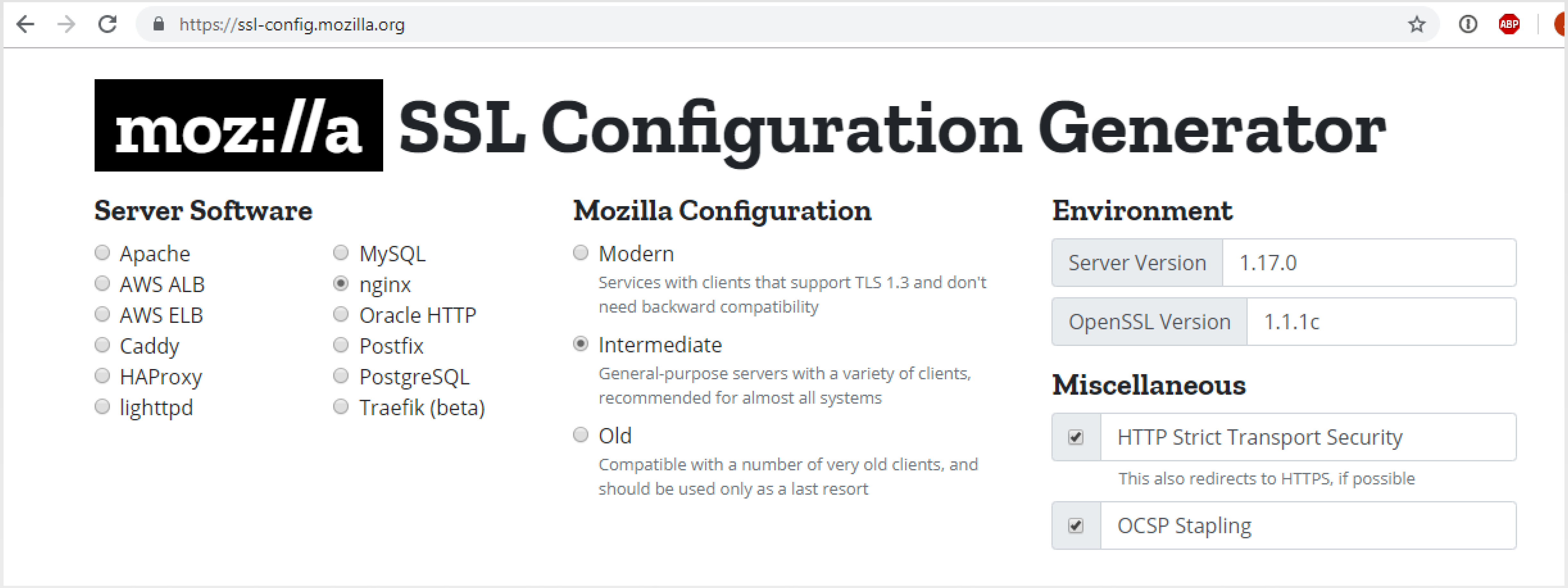
OWASP
Open Web Application Security Project

OWASP.ORG

V9 Communications Security

- Now DevSecOps friendly
  - No more building a secure chain / path … what does that even mean?

- Easier to comply … automatically
- TLS all the things
- Use modern configuration builders
- Use the verification tools you use today



https://ssl-config.mozilla.org

moz://a SSL Configuration Generator

Server Software

- ○ Apache
- ○ AWS ALB
- ○ AWS ELB
- ○ Caddy
- ○ HAProxy
- ○ lighttpd
- ○ MySQL
- [✓ nginx]
- ○ Oracle HTTP
- ○ Postfix
- ○ PostgreSQL
- ○ Traefik (beta)

Mozilla Configuration

- ○ Modern
  Services with clients that support TLS 1.3 and don't need backward compatibility
- [✓ Intermediate]
  General-purpose servers with a variety of clients, recommended for almost all systems
- ○ Old
  Compatible with a number of very old clients, and should be used only as a last resort

Environment

Server Version	1.17.0
OpenSSL Version	1.1.1c

Miscellaneous

- [✓] HTTP Strict Transport Security
  This also redirects to HTTPS, if possible
- [✓] OCSP Stapling

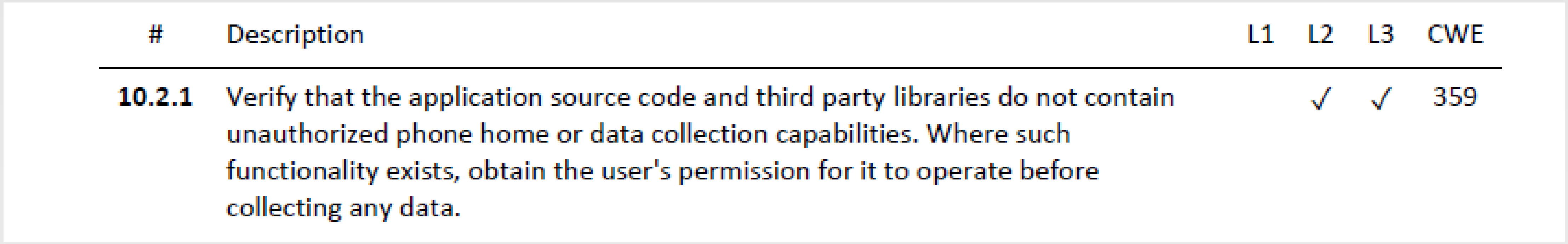
V10 Malicious Code

V10 Malicious Code

- Major update to this section to cover more than just time bombs and Easter eggs
  - Detect malicious code introduction
  - Continuous detection through building
  - Privacy invading libraries
  - Malicious business logic, such as salami attacks
  - Privacy invading permissions (camera, location, microphone, contacts, etc)

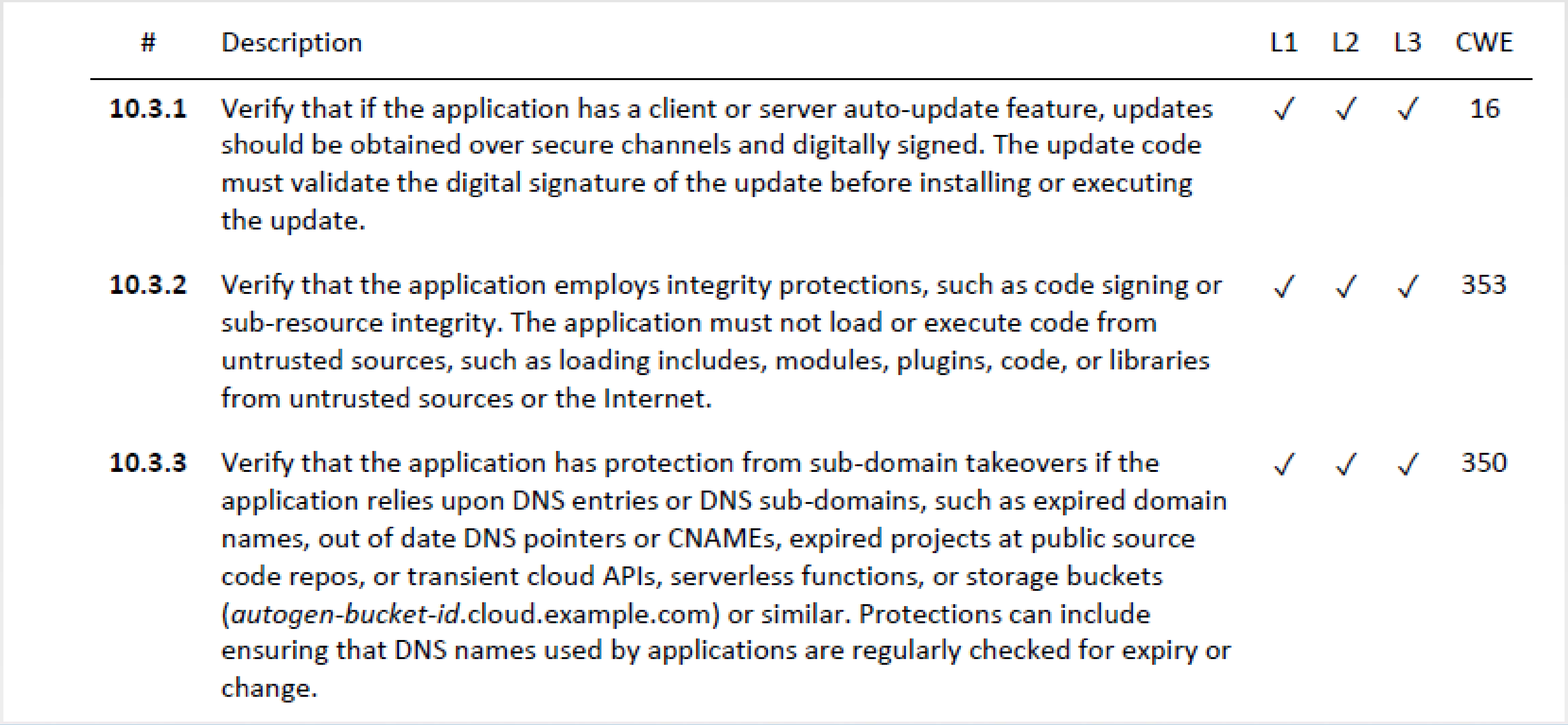
- Mostly Level 3 except the Application Integrity Controls

V10 Malicious Code



#	Description	L1	L2	L3	CWE
10.2.1	Verify that the application source code and third party libraries do not contain unauthorized phone home or data collection capabilities. Where such functionality exists, obtain the user's permission for it to operate before collecting any data.		✓	✓	359

V10 Malicious Code

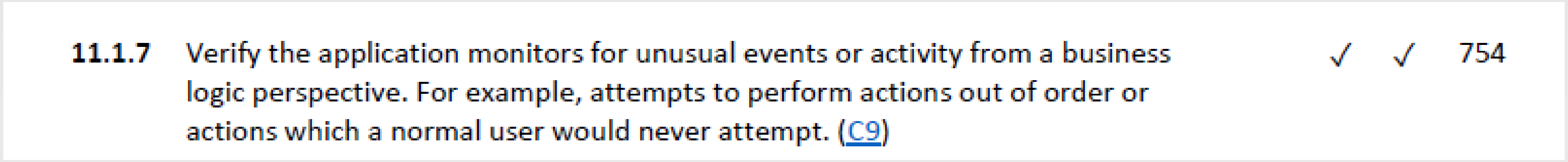


#	Description	L1	L2	L3	CWE
10.3.1	Verify that if the application has a client or server auto-update feature, updates should be obtained over secure channels and digitally signed. The update code must validate the digital signature of the update before installing or executing the update.	✓	✓	✓	16
10.3.2	Verify that the application employs integrity protections, such as code signing or sub-resource integrity. The application must not load or execute code from untrusted sources, such as loading includes, modules, plugins, code, or libraries from untrusted sources or the Internet.	✓	✓	✓	353
10.3.3	Verify that the application has protection from sub-domain takeovers if the application relies upon DNS entries or DNS sub-domains, such as expired domain names, out of date DNS pointers or CNAMEs, expired projects at public source code repos, or transient cloud APIs, serverless functions, or storage buckets (*autogen-bucket-id*.cloud.example.com) or similar. Protections can include ensuring that DNS names used by applications are regularly checked for expiry or change.	✓	✓	✓	350

V11 Business Logic Verification

- Key considerations:
  - Are steps being performed in the right order?
  - Are inputs within normal limits?
  - Are steps performed in “human time”? (not fast like an automated tool)
  - Are anti-automation controls being used
- Threat model (attack driven design) business logic risks
- Race conditions that might affect business logic (TOCTOU)
- Monitoring, alerting, and detection of unusual business logic events

V11 Business Logic Verification



11.1.7	Verify the application monitors for unusual events or activity from a business logic perspective. For example, attempts to perform actions out of order or actions which a normal user would never attempt. (C9)	✓	✓	754

V12 Files and Resources

V12 Files and Resources

- Major revamp
  - Architecture items moved to architecture
  - Configuration items moved to configuration
- Upload - Size and number
- Contents and integrity
- Storage, including malicious file detection
- Execution, including LFI, RFI, and SSRF
- Download

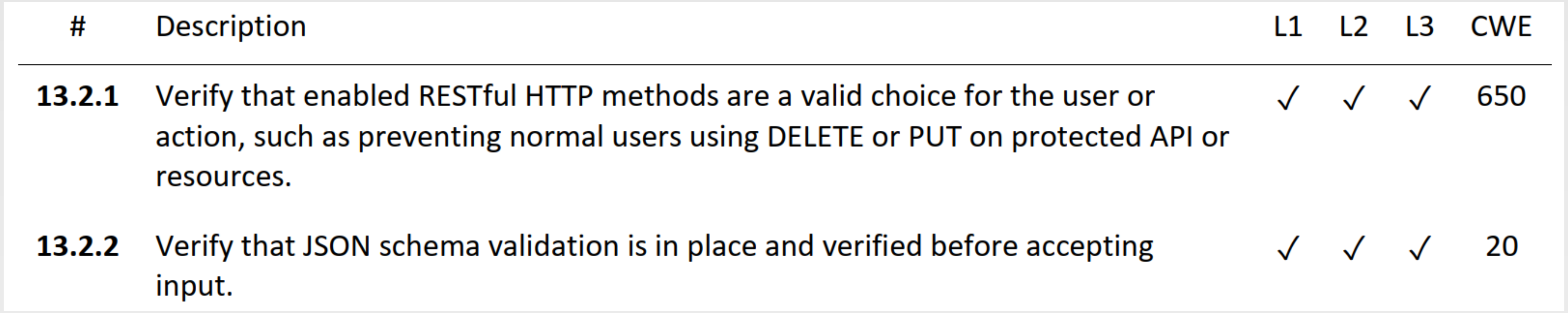
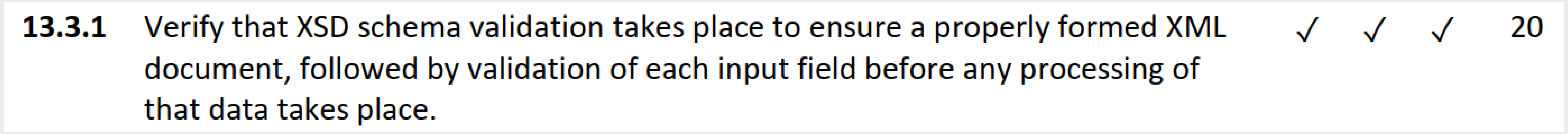
V12 Files and Resources

#	Description	L1	L2	L3	CWE
12.1.1	Verify that the application will not accept large files that could fill up storage or cause a denial of service attack.	✓	✓	✓	400
12.1.2	Verify that compressed files are checked for "zip bombs" - small input files that will decompress into huge files thus exhausting file storage limits.		✓	✓	409
12.1.3	Verify that a file size quota and maximum number of files per user is enforced to ensure that a single user cannot fill up the storage with too many files, or excessively large files.		✓	✓	770

V13 API Security

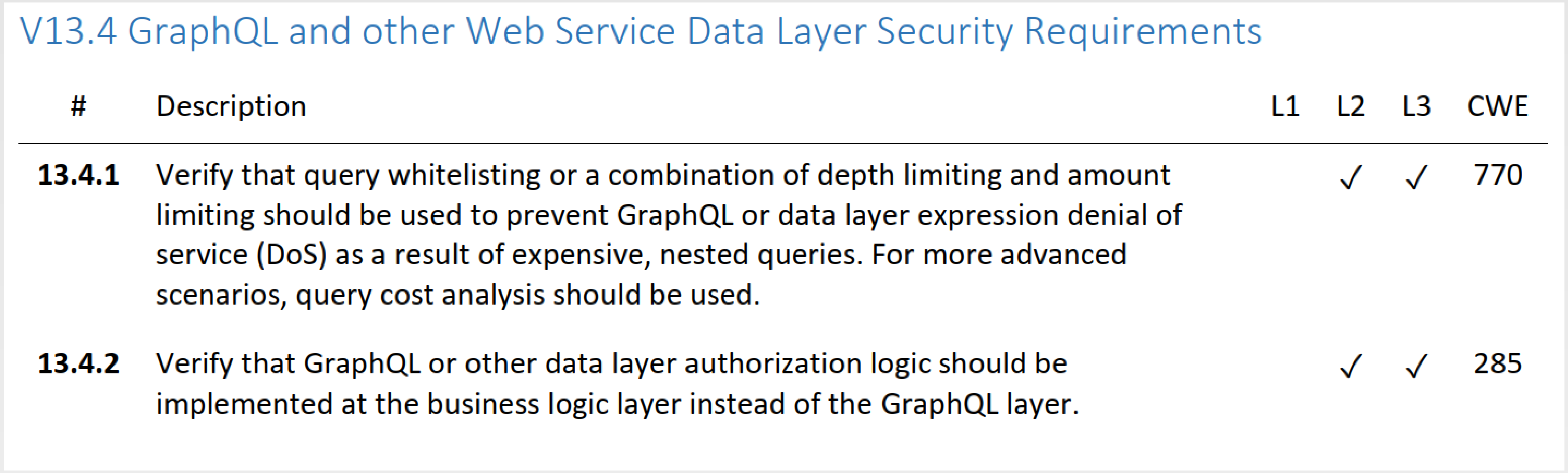
- Total revamp
- General controls include common sense API controls
  - Must be read in conjunction with authentication, authorization and session management
- RESTful includes schema validation, CORS and origin attacks
- SOAP – fewer more impactful items, far clearer, and not obfuscated
- GraphQL and Web Service Data Layer

V13 API Security

#	Description	L1	L2	L3	CWE
13.2.1	Verify that enabled RESTful HTTP methods are a valid choice for the user or action, such as preventing normal users using DELETE or PUT on protected API or resources.	✓	✓	✓	650
13.2.2	Verify that JSON schema validation is in place and verified before accepting input.	✓	✓	✓	20
13.3.1	Verify that XSD schema validation takes place to ensure a properly formed XML document, followed by validation of each input field before any processing of that data takes place.	✓	✓	✓	20

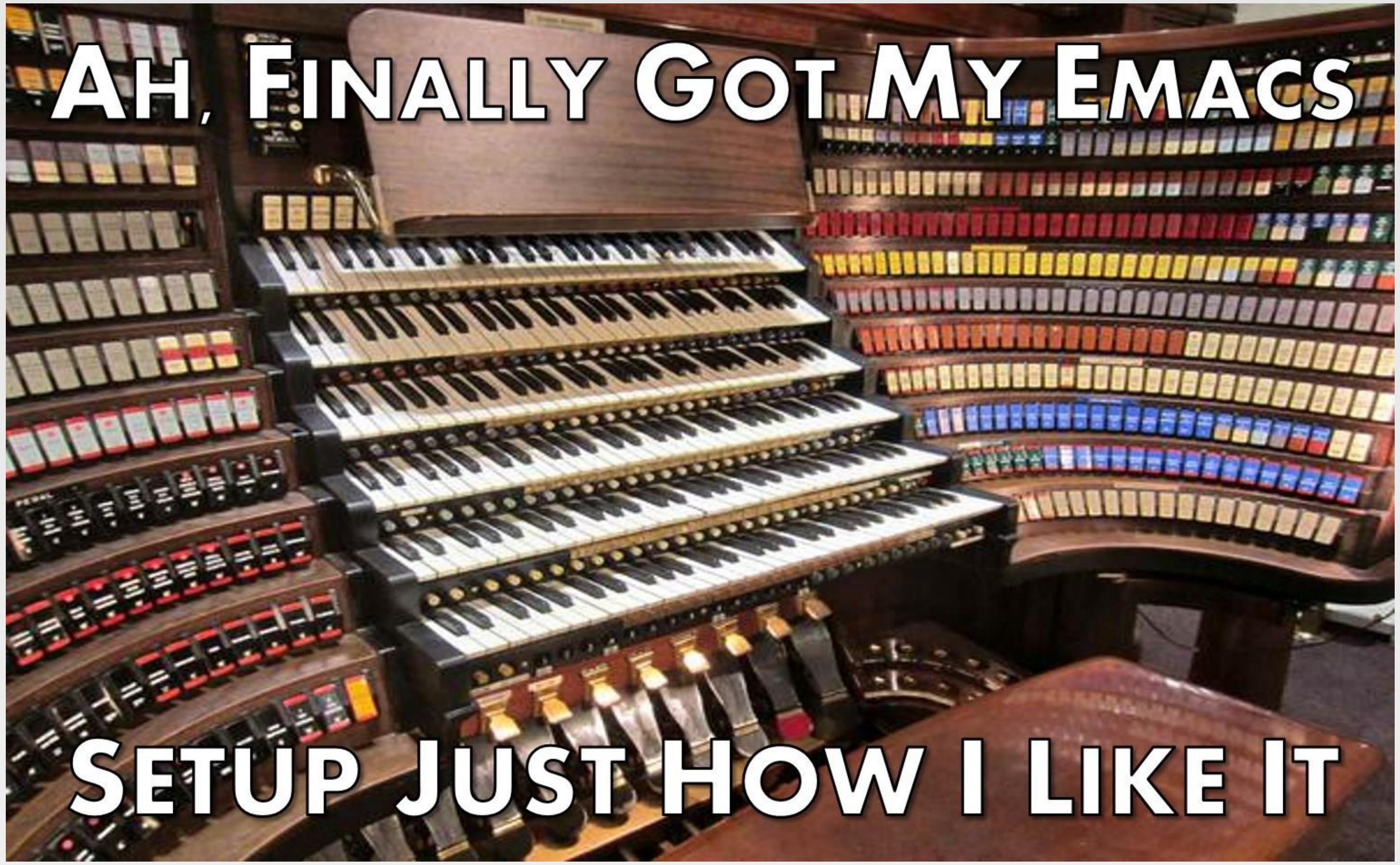
V13 API Security



V13.4 GraphQL and other Web Service Data Layer Security Requirements

#	Description	L1	L2	L3	CWE
13.4.1	Verify that query whitelisting or a combination of depth limiting and amount limiting should be used to prevent GraphQL or data layer expression denial of service (DoS) as a result of expensive, nested queries. For more advanced scenarios, query cost analysis should be used.		✓	✓	770
13.4.2	Verify that GraphQL or other data layer authorization logic should be implemented at the business logic layer instead of the GraphQL layer.		✓	✓	285

V14 Configuration



V14 Configuration

- Major revamp
- Repeatable, Continuous integration, continuous deployment
- Support for DevSecOps culture and agile practices
- Dependency checks mandated
- HTTP Security Headers

V14 Configuration

14.1.4	Verify that the application, configuration, and all dependencies can be re-deployed using automated deployment scripts, built from a documented and tested runbook in a reasonable time, or restored from backups in a timely fashion.	✓	✓	

14.2.4	Verify that third party components come from pre-defined, trusted and continually maintained repositories. (C2)	✓	✓	829
14.2.5	Verify that an inventory catalog is maintained of all third party libraries in use. (C2)	✓	✓	

Mobile ASVS

- Focused on mobile specific issues (client-side and transport security)
- Application server side should be assessed using the main ASVS
- Similar structure to main ASVS
- Different team – the OWASP Mobile Security project
- Very closely linked to the OWASP Mobile Security Testing Guide
- Provides platform independent guidance

I’M IN. HOW DO I USE IT?

Generally Accepted Security Practices

Secure code review

Security architecture

Integration testing

Peer coding checklists

Developer training

Unit testing

Hybrid reviews

DevSecOps automation

Vulnerability programs

Consultant training

Tool Benchmark

Secure coding checklist

Penetration test

Deployment checklist

Functional constraints

Planning Sprint Assistance

Non-functional and functional features

Supplier Benchmarking

How do I use it?

- Use it as is or fork it

- Level 1 – entry level, penetration testing
- Level 2 – most apps
- Level 3 – apps that can kill you or run the world economy

- Deep standards have no bounds

How to get involved

- Grab a copy today and start to migrate from earlier versions and T10
- We need translations. Please join the #project-asvs channel on OWASP Slack
- We need case studies! If you use ASVS, we'd love to hear from you!
- Create issues or pull requests if you identify issues

When can I get it?

- The ASVS 4.0.1 Final is available now!

- OWASP Wiki – Word, PDFs, CSVs, and Hot Linkable markdown
- GitHub – Most Recent Static Version is in the 4.0.1 branch
- GitHub – Development Version is in the master branch

- You can also get the base version of this presentation so you can give this to your local chapter, school, college, or workplace! (Thanks again to Andrew van der Stock!)

josh.grossman@owasp.org (ASVS) | josh@appsec-labs.com ($dayjob)

@joshcgrossman

THANK YOU