OWASP Top 10: 2021 Edition

Matthew Sheehan

About OWASP Top 10

1st release in 2003. - 7th release in 2021.

	Release
Chrome	2008
iPhone	2007
Azure	2008
AWS	2006
Git	2005
Facebook	2004
OWASP Top 10	**2003**

About OWASP Top 10

- Targeted at developers, architects, and security professionals.
- The ten most critical risks identified in web applications.
  - Risk = Likelihood x Impact
- Built as a data-driven ***awareness*** document, not as a standard.
- Was actually suppose to be OWASP Top 10: 2020. 😅

What are the OWASP Top 10: 2021

- **Broken Access Control**
- **Cryptographic Failures**
- **Injection**
- **Insecure Design**
- **Security Misconfiguration**
- **Vulnerable & Outdated Components**
- **Identification & Authentication Failures**
- **Software & Data Integrity Failures**
- *Security Logging & Monitoring Failures*
- *Server-Side Request Forgery*

A01: Broken Access Control

- Unauthorized access to accounts
- Unauthorized elevation of privilege
- Unauthorized create/read/update/delete operations
- JWT tampering or Cookie manipulation
- CORS misconfigurations

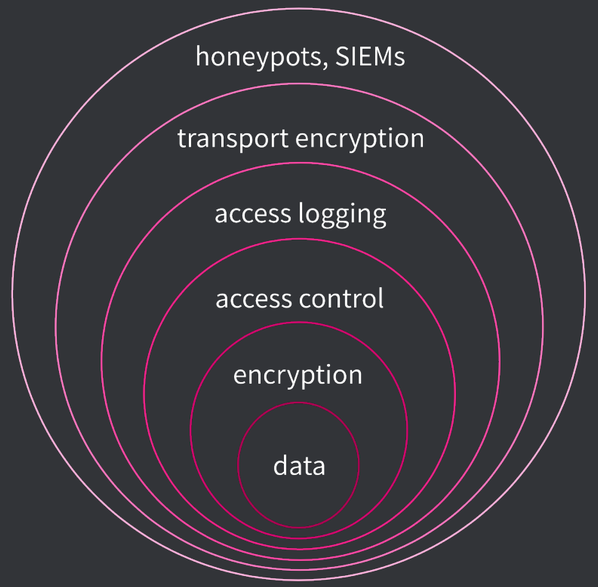
Recommendations

- Principle of Complete Mediation
- Principle of Least Privilege / Deny by default
- Minimize CORS usage

⬆⬆⬆⬆ A05: Broken Access Control

A02: Cryptographic Failures

- Ineffective/missing data-at-rest encryption.
- Ineffective/missing TLS.
- Ineffective/missing password hashing.

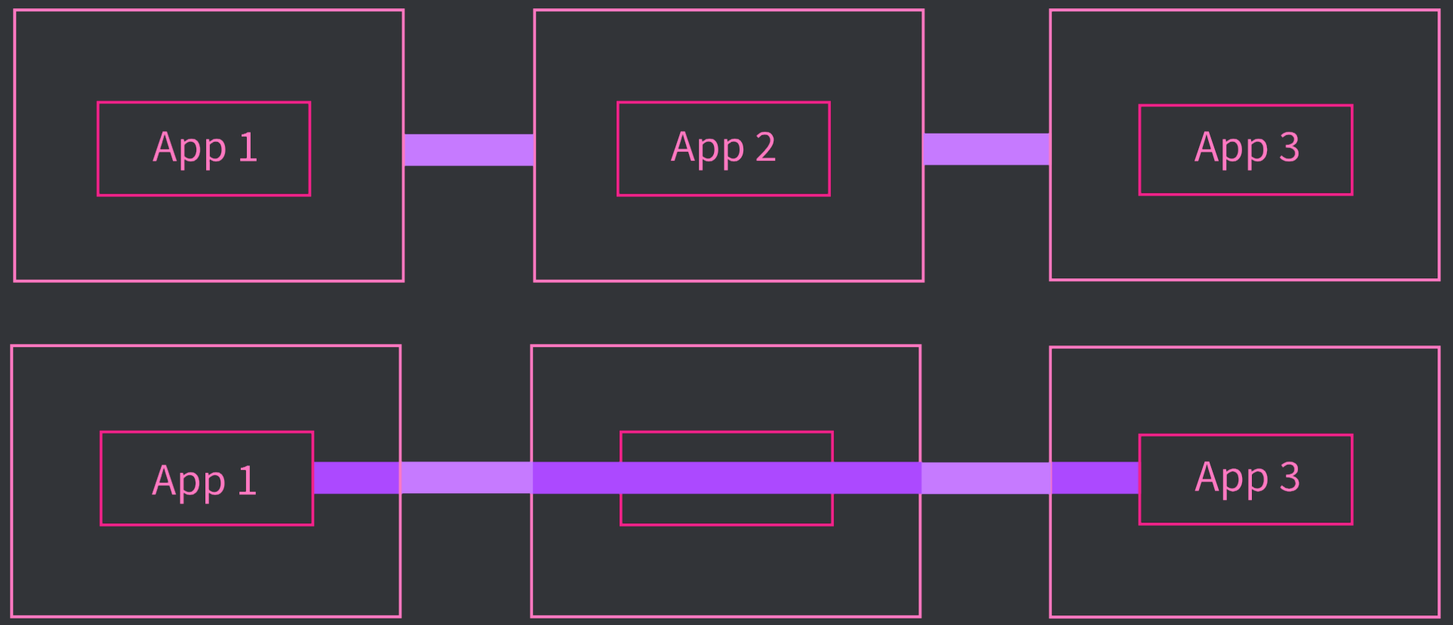


Recommendations

- Static code analysis tools
- HTTP Strict-Transport-Security Header (HSTS)
- Defense-in-depth
- Classify data processed
- Only store data you need
- Application Level Encryption

⬆ Previously known as **A03:2017 Sensitive Data Exposure**.

A02: Cryptographic Failures




```
{
    "to": "App 1",
    "from": "App 2",
    "message": "encrypt(msg)"
}
```

A03 Injection

- First version since 2010 not A01.
  - Improved frameworks attributed for the decline.
- LDAP injection, XSS, SQL

Recommendations

- Validate, sanitize, escape any data that crosses trust boundaries.
- Use frameworks that assemble HTML safely.
- WAFs have many rules for blocking and detecting injection.

⬇⬇ **A01:2017 Injection**, includes **A07:2017 Cross-Site Scripting**

A03 Injection

```
(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")
 @(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:(2(5[0-5]|[0-4][0-9])|1[0-9][0-9]|[1-9]?[0-9]))\.){3}
 (?:(2(5[0-5]|[0-4][0-9])|1[0-9][0-9]|[1-9]?[0-9])|[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\])
```

matt'or'1'!='@sheehan.codes

```
"SELECT * FROM [table] WHERE [email] = '" + email + "'"
```

A04: Insecure Design

- The difference between an insecure design and an insecure implementation.
- Requires considering security before code is written. 😬
- Secure design cannot be bolted on later.

Recommendations

- Threat modeling, secure reference architectures
- Establish a Secure Development Lifecycle
- Establish security guiding principles
- Unit test, integration test
- Adopt better frameworks

A05: Security Misconfiguration

- Infrastructure as Code has lead to an increase in **catching** security misconfigurations.
- Improper configurations on cloud services, missing security hardening.
- Unnecessary features enabled/installed.
- Challenges:
  - Keeping up with the latest public cloud provider changes.
  - Security is an emergent property of systems. Requires a multi-disciplinary approach.

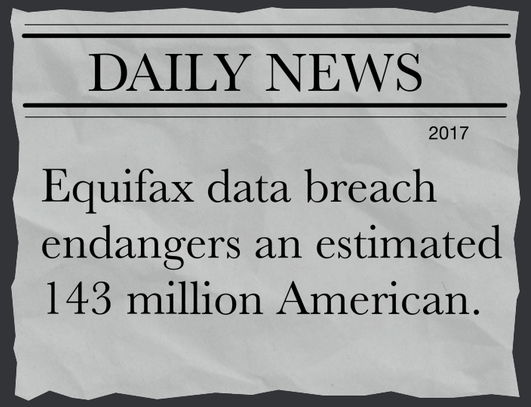
Recommendations

- Static code analysis tools
- Create "paved-roads" for development
- Code reviews
- Read the docs

⬆ **A06: Security Misconfiguration**, includes **A04:2017 XML External Entities**

A06: Vulnerable and Outdated Components

- Would probably be A01 if re-evaluated after Log4Shell
- Modern software includes a lot of external code.
- Includes OS, runtimes, and libraries.
- Not upgrading dependencies in a risk-based, timely fashion.



Recommendations

- Know the versions of all the components you use and indirectly use.
- Minimize the attack surface - remove unused dependencies.
- Obtain packages from official sources, ensure signed (A08:2021-Software and Data Integrity Failures)
- CI/CD tools to warn for outdated components.
- Continuously monitor for vulnerabilities.

⬆⬆⬆ Previously known as **A09:2017 Using Components with Known Vulnerabilities**

A07: Identification and Authorization Failures

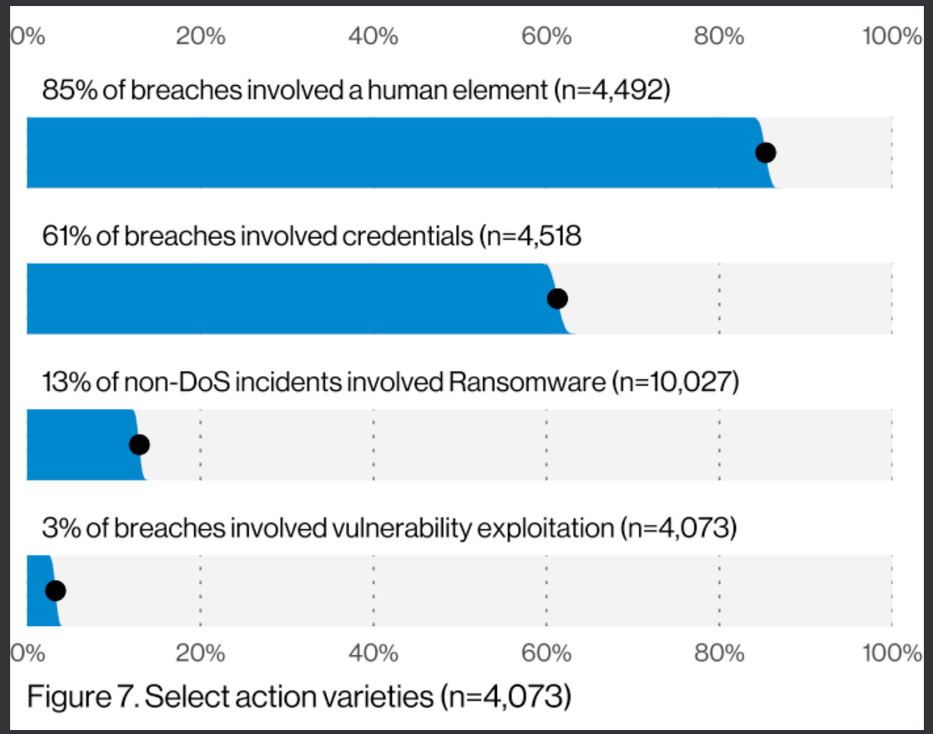


Figure 7. Select action varieties (n=4,073)

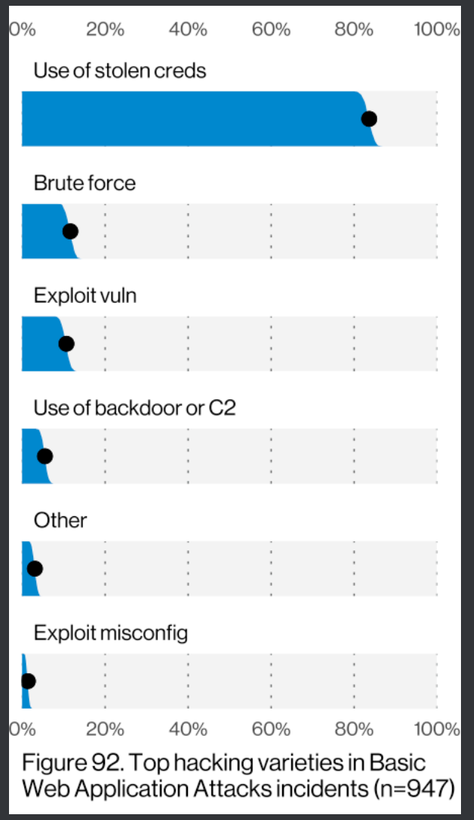


Figure 92. Top hacking varieties in Basic Web Application Attacks incidents (n=947)

Verizon 2021 Data Breach Investigations Report

A07: Identification and Authorization Failures

- Permitting weak or well known passwords.
- Permitting automated brute force / credential stuffing attacks.
- Permitting weakly hashed passwords (A02: Cryptographic Failures).
- Insecure password recovery.
- Sessions that never expire.
- Enumeration attacks.

Recommendations

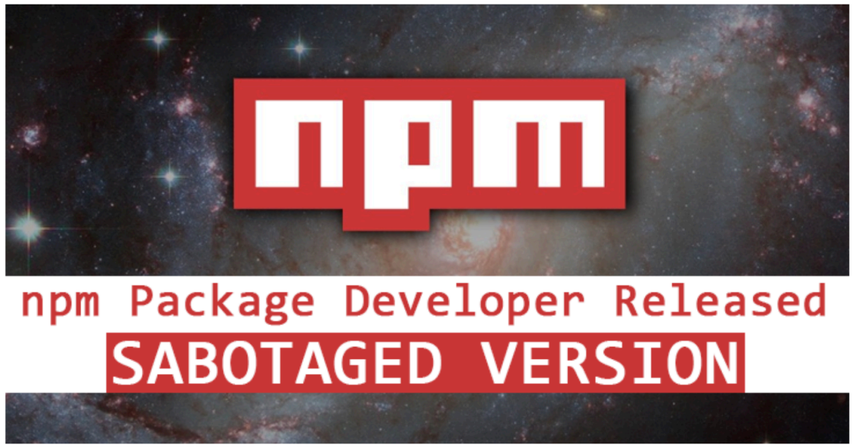
- Follow best practices for passwords and rate limiting logins.
- Validate authentication and authorization for every request that shouldn't be public.
- Countermeasure CSRF / XSS attacks > require re-authentication for sensitive features.
- Add Multi-factor Authentication where possible

⬇⬇⬇⬇⬇ Replaces **A02: Broken Authentication**

A08: Software and Data Integrity Failures

npm Package Developer Released Sabotaged Version That Deletes Files for Users Based in Russia

By **Guru** - March 18, 2022 0



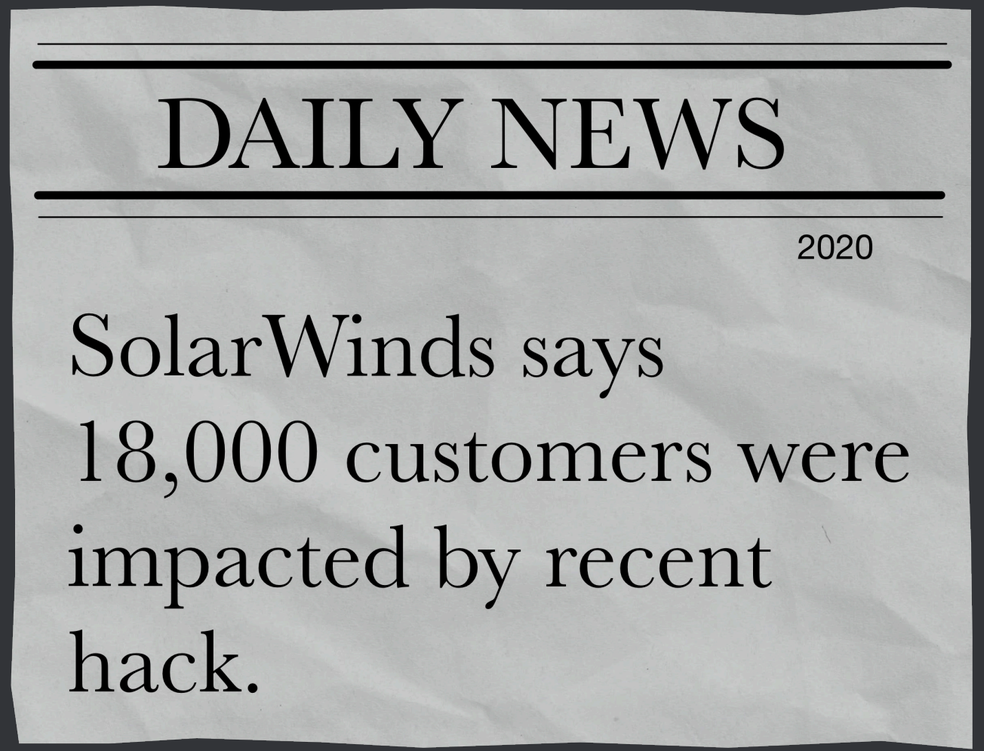
Several weeks ago, the developer of the “node-ipc,” a popular npm package with more than a million weekly downloads has protested the Russo-Ukrainian War by releasing sabotaged versions of the library. Due to this escalating situation, the open-source and software supply chain is under security concern.

A08: Software and Data Integrity Failures

- Plugins or libraries from untrusted sources.
- CI/CD pipelines without commit or build checks.

Recommendations

- Verify software and data is from trusted sources.
- Verify components do not contain known vulnerable dependencies.
- CI/CD has proper configuration and access control.



✳ Includes **A08:2017 Insecure Deserialization**

A09: Security Logging and Monitoring Failures

- Attackers rely on insufficient logging and monitoring to achieve their goals before detection.
- Average time to identify a breach is 280 days.
- 80% of breaches involve PII.

Recommendations

- Centralized, append-only logging and monitoring store.
- Automatic alerting of suspicious activity.
- Audit trail for logins, access failures, and "high-value" transactions.
- Data hygiene across sources.
- Expire the data when not needed anymore.

⬆ Previously known as **A10:2017 Insufficient Logging and Monitoring**

A09: Security Logging and Monitoring Failures

What to log

- Input validation failures (invalid parameter names, invalid protocol)
- Output validation failures (database record mismatch)
- Authentication failures and successes
- Application errors
- Application startup, shutdown, configuration changes
- High-risk functions (All access control events, deleting users, assigning privileges, all actions by administrative users, data imports and exports)

What not to log

- PII and PHI
- Access tokens and Session IDs
- Connection strings
- Encryption keys and secrets

https://cheatsheetseries.owasp.org/cheatsheets/Logging*Vocabulary*Cheat_Sheet.html

A10: Server-Side Request Forgery

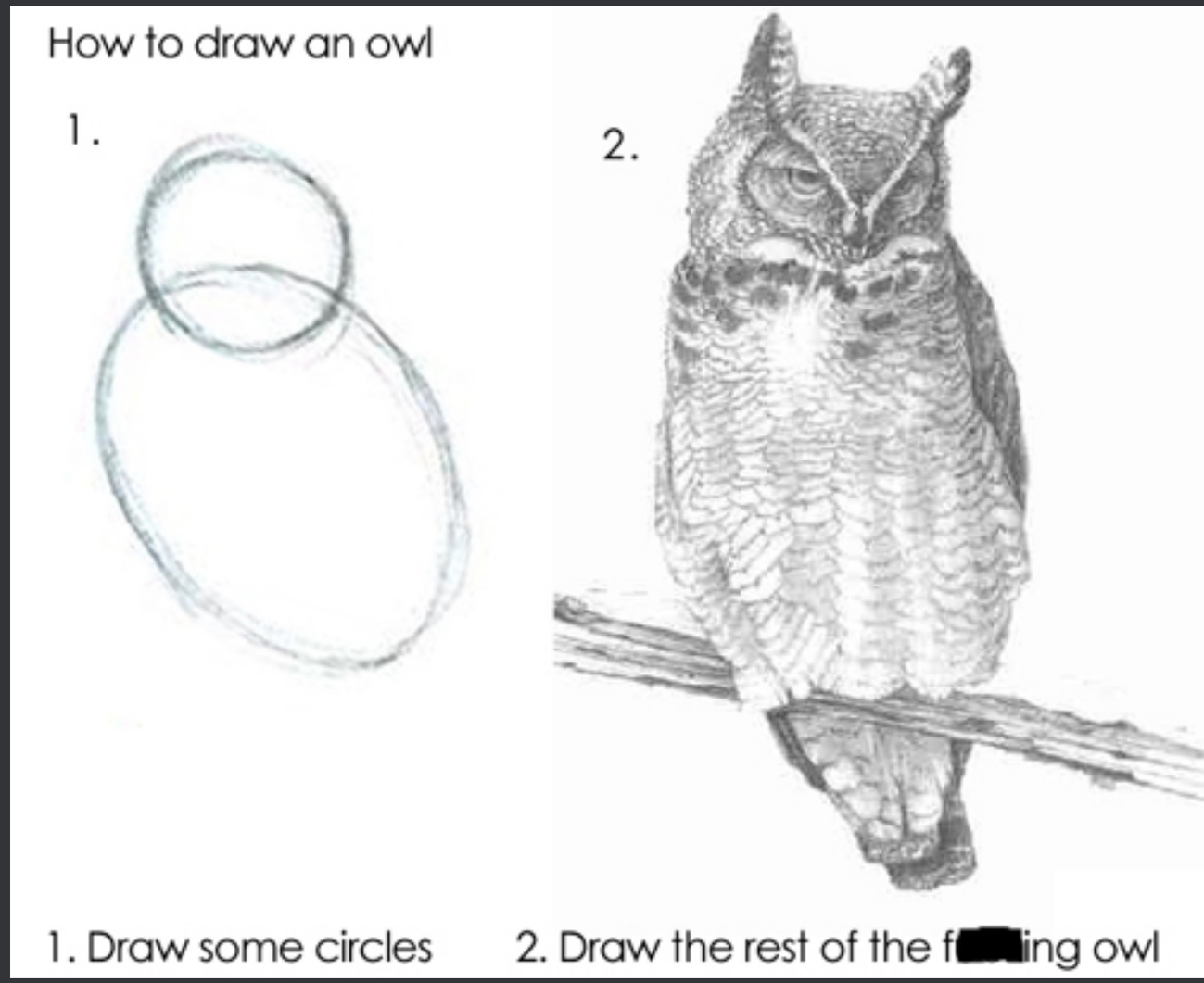
- It's just as important to do authentication & access control between the backend services.
- Avoid taking URLs as a parameter that the server acts on.
- Limit services with network controls.
- Gitlab in 2021

Recommendations

- Restrict outgoing access on web servers, restrict incoming access on internal servers.
- Zero-Trust Architecture, perimeter security is not sufficient.
- Avoid taking URLs as a full parameter that the server acts upon.
- Build safe URLs with URL encoding of parameters.

Next Steps

- Security is a process not a product.
- Security is always a moving target.
- A lot of security exists outside of code.
- Adopt a security mindset, shift(expand)-left.



Other projects

- OWASP ASVS
- OWASP SAMM
- OWASP Proactive Controls

Links

- https://owasp.org/Top10/
- https://github.com/OWASP/Top10
- https://www.verizon.com/business/resources/reports/dbir/2021/masters-guide/
- https://youtu.be/opRMrEfAIiI