Information Security

Security Fundamentals

Stuart Cianos, CISSP <scianos@alphavida.com>

Version 1.5 - updated 2020-12-19 for CISE Security Program @ Merrit College

TOC

Overview

Information Security encompasses processes and methodologies, but is also a state of being... such that our processes and methodologies drive us towards our overall desired condition. This incorporates many factors - some technical, some environmental and some might opine the most important - which is the human factor.

As information security professionals living in an evolving landscape of constant technical integration and innovation, we are in the unique position of being aware of the implications as well as having the opportunity to make a positive impact within the realms of technical and thought leadership.

About the Speaker

Stu is a Bay Area native with a passion for technology and its ability to create positive impact in the lives of those it touches. With a focus on information security, Stu has served both the public and private sectors across a variety of industries including healthcare, education, social commerce/advertising, government technologies and online entertainment. Stu holds a CISSP and enjoys being active in the information security community. In his free time, Stu is an avid street photographer and ethnographer with a love for the abstract. When he's not capturing California's amazing urban landscape with his camera, he enjoys the arts and gardening.

Stuart currently serves as a Security Architect with BlueOwl LLC.


SEGMENT ONE

What is Information Security

Or... perhaps we can distill this down to an easier question:

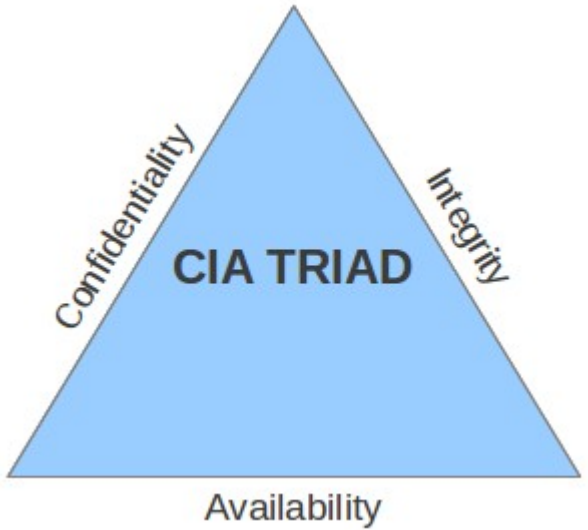
What is the CIA Triad?

The CIA Triad

1. **Confidentiality:** The attribute of data that ensures information is only being exposed to appropriately authorized parties and other systems.

2. **Integrity:** The attribute of data that ensures the information accurately reflects reality. Data and systems/ processes cannot be modified without authorization.

3. **Availability:** The attribute of data that ensures it is always available to appropriate parties when required for use.



Source: THE (ISC) 2 CYBERSECURITY LEXICON,
https://www.isc2.org/-/media/ISC2/Training/The-ISC2-Cybersecurity-Lexicon.ashx

The Five Pillars of Information Security

… are important additional concepts & builds upon the CIA triad

Core Concepts - CIA Triad

1. **Confidentiality:** The attribute of data that ensures information is only being exposed to appropriately authorized parties and other systems.
2. **Integrity:** The attribute of data that ensures the information accurately reflects reality. Data and systems/ processes cannot be modified without authorization.
3. **Availability:** The attribute of data that ensures it is always available to appropriate parties when required for use.

plus these - Makes 5 pillars

4. **Non-Repudiation:** A method of guaranteeing message transmission between parties via digital signature and/or encryption.
5. **Authentication:** The process or action of verifying the identity of a user or process.

Source: THE (ISC) 2 CYBERSECURITY LEXICON,
https://www.isc2.org/-/media/ISC2/Training/The-ISC2-Cybersecurity-Lexicon.ashx

The Five Pillars of Information Security

Real World Use Cases

Core Concepts - CIA Triad

1. **Confidentiality:** Before transmitting medical records electronically, encryption might be used to ensure that unauthorized recipients cannot steal the data in transit.
2. **Integrity:** A software system checks the signature on the package to ensure it wasn't modified without authorization and is from a trusted developer.
3. **Availability:** A company implements denial of service protection on a circuit to ensure their systems remain available during a DDOS incident.

plus these - Makes 5 pillars

4. **Non-Repudiation:** An electronic document management system uses digital certificates to allow users to electronically sign documents.
5. **Authentication:** The ATM requires a PIN# to authenticate the user to their card.

The Five Pillars of Information Security

Real World Use Cases

Core Concepts - CIA Triad

1. **Confidentiality:** Principles of least privilege, the need-to-know principle.
2. **Integrity:** Physical integrity of the medium, logical integrity of the data; ensuring that the data is able to be stored and retrieved without fault.
3. **Availability:** Access to resources.

plus these - Makes 5 pillars

4. **Non-Repudiation:** The ability to prove authorship.
5. **Authentication:** Drives access control primitives like authentication, authorization, and audit.

The CIA Triad

The base concepts defined by the CIA Triad are the lingua franca used by information security professionals, regulators, and lawmakers on the domestic and international scene.

Example: FISMA

44 U.S. Code § 3542 - Definitions

(b) **Additional Definitions.—** As used in this subchapter:

(1) The term “information security” means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide—
(A) ***integrity***, which means guarding against improper information modification or destruction, and includes ensuring information nonrepudiation and authenticity;
(B) ***confidentiality***, which means preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information; and
(C) ***availability***, which means ensuring timely and reliable access to and use of information.

Professional Integrity and Balance in Implementation

Information security professionals should seek a balanced approach… which allows the organization to *safely* excel to their maximum potential.

CIA Triad by Example

C, I, and/or A or nothing at all?

Alice uses her credit card to buy a hammer. The POS network was compromised and both track 1 and 2 of her credit card stripe were exfiltrated.

C, I, and/or A or nothing at all?

Bob writes a home-grown "crypto" algorithm called ROT13+ and implements it. Later, he discovers that it's possible to change the ciphertext in transit to *reliably* manipulate plaintext.

C, I, and/or A or nothing at all?

Cindy enjoys visiting a photo sharing site. One day, she tries to access it, but it's down. Later, the site comes back online stating that "too much" network traffic knocked their entire data center offline.

C, I, and/or A or nothing at all?

The CEO of a company opens an email from a board member's address. She opens an attachment which silently installs software on her personal laptop.

C, I, and/or A or nothing at all?

An operating engineer at a waste treatment plant notices SCADA devices on an air-gapped network emitting low frequency sounds while checking vibration tolerances on a pump.

C, I, and/or A or nothing at all?

A plastic surgeon at a private clinic decides to take a break at the local coffee shop. While there, the MD reviews diagnostic images on his laptop in plain view of other patrons.

C, I, and/or A or nothing at all?

A nurse working at a hospital wants to see the results of his own medical test, but his PCP is out until tomorrow. He accesses his own medical record using his own employee access to see his data.


Security
Governance

Security Governance vs. Security Management

1. **Due Diligence:** Understanding the risks to an organization. This might include comprehensive reviews of policies, comprehensive audit of an organization's posture, vetting mergers and acquisitions, etc.

2. **Due Care:** Taking necessary actions to address possible risks. This might include the implementation of training, the use of security tools to perform penetration tests, etc.

As written by Massachusetts Justice Samuel Putnam (1768-1853), directs trustees "to observe how men of prudence, discretion and intelligence manage their own affairs, not in regard to speculation, but in regard to the permanent disposition of their funds, considering the probable income, as well as the probable safety of the capital to be invested."

Excerpt from: https://en.wikipedia.org/wiki/Prudent_man_rule

Security Governance vs. Security Management

1. **Information Security Governance:** Information security governance is concerned with the oversight and accountability of the organization's information security program. Information security governance will include policy development, strategic planning based on risk, funding security initiatives, and to support the overall goals of the organization via governance. *More concerned with Due Diligence.*

2. **Information Security Management:** Information security management is concerned with the day to day operation of the information security program. This might include making decisions based on risks or events, enforcement of policies, implementation of the security program, allocating resources, and planning. *More concerned with Due Care.*

Due Diligence and Due Care

Documentation is important in IT governance. It may mean the difference between being able to demonstrate Due Diligence/Care vs. negligence of duty.

Data Classification

Data Classification

Data classification is the process of identifying data elements and:

- Determining its location
- Determining its level of access
- Determining its level of protection

Data Classification

Data classification is a form of both due care and due diligence:

- A data classification *policy*
- The *act* of classifying data
- Documenting the classification

Data Classification

May be driven by policy *and* regulation:

- An organization classifies data to protect proprietary information; and
- PCI-DSS classifies PII to protect card holders and merchants.

Data Classification

May need to address other concerns, such as the recovery point and recovery time objectives of the impacted data.

Data Classification

Basic, broad process commonly used:

1. Create an inventory of data/assets
2. Classify the data elements
3. Label the data elements appropriately
4. Handle the data appropriately


Security
Controls

Security Controls

Core controls which help support Confidentiality, Integrity, and Availability in an organization...

1. **Separation of Duties:** aka Separation of Powers. Duties are disseminated amongst multiple individuals. Avoids appearance of collusion as well as fraud or real collusion.

2. **Least Privilege:** Information and resources are provided to the extent necessary to complete a job. Individuals have the *minimum* access necessary to perform the assigned tasks.

3. **Job Rotation:** Rotate team members in sensitive positions to prevent or expose errors or malicious activity. Also has benefit of opportunities for additional employee experience.

4. **Mandatory Vacation:** Like job rotation, mandatory vacations can help expose errors, oversights and other forms of malicious activity during absence. It also helps prevent burnout.

5. **Dual Person Control:** The requirement that two (or more) individuals be required to complete a single task. This can prevent fraud, collusion, and the appearance of impropriety. A good example of this control expressed as a mathematical process or equation is "Shamir's Secret".

Security Controls

Security controls are put in place to reduce risks around an organization's security posture.

1. **Preventive Controls:** Controls put in place to prevent an incident. An example of a preventive control is the lock on a door to a data center.

2. **Detective Controls:** Controls put in place to identify an incident when it occurs. An example would be an alarm on a router interface that is saturated due to a DDOS attack.

3. **Directive Controls:** Specify acceptable rules of behavior. Example: HR Policies.

4. **Deterrent Controls:** Discourage people from violating directive controls.

5. **Corrective Controls:** Controls put in place to minimize the risk or impact *after* an event has occurred. An example of a corrective control would be applying a BGP routing update to swing traffic in response to a DDOS attack.

6. **Compensating Controls:** Controls which are implemented to substitute for the loss of primary controls and further mitigate risk.

7. **Recovery Controls:** Controls which are implemented to restore conditions to normal after an event.

Security Controls

Security controls can exist in many spaces...

1. **Physical Controls:** A control that exists in physical space. An example of various controls that exist in physical space are locks, fingerprint scanners, walls, fire alarms/suppression and generators.

2. **Procedural Controls:** Controls that exist as a set of documented procedures or processes. Examples include disaster recovery/BC plans & drills, employee training, and incident response processes.

3. **Technical Controls:** A control that exists within a technical system. Examples include user authentication, access controls, network devices like routers and firewalls, and endpoint threat mitigation software.

4. **Compliance Controls:** A control that exists as a regulatory requirement or legal requirement. Examples include regulations like PCI-DSS, legal requirements like HIPAA, and breach notification laws like those of California and Massachusetts.

Security Controls

Security controls may be strategically layered to provide multiple levels of protection.

Security Controls

An auditor may validate that required controls are in place and properly implemented, particularly in regulated industries.

Security Controls

Frameworks, like NIST's Cybersecurity Framework, are used by security professionals and auditors to ensure comprehensive coverage.

	Risk Assessment (ID.RA): The organization understands the cybersecurity risk to organizational operations (including mission, functions, image, or reputation), organizational assets, and individuals.		· **ISO/IEC 27001:2013** A.12.6.1, A.18.2.3 · **NIST SP 800-53 Rev. 4** CA-2, CA-7, CA-8, RA-3, RA-5
		ID.RA-2: Threat and vulnerability information is received from information sharing forums and sources	· **ISA 62443-2-1:2009** 4.2.3, 4.2.3.9, 4.2.3.12 · **ISO/IEC 27001:2013** A.6.1.4 · **NIST SP 800-53 Rev. 4** PM-15, PM-16, SI-5
		ID.RA-3: Threats, both internal and external, are identified and documented	· **COBIT 5** APO12.01, APO12.02, APO12.03, APO12.04 · **ISA 62443-2-1:2009** 4.2.3, 4.2.3.9, 4.2.3.12 · **NIST SP 800-53 Rev. 4** RA-3, SI-5, PM-12, PM-16
		ID.RA-4: Potential business impacts and likelihoods are identified	· **COBIT 5** DSS04.02 · **ISA 62443-2-1:2009** 4.2.3, 4.2.3.9, 4.2.3.12 · **NIST SP 800-53 Rev. 4** RA-2, RA-3, PM-9, PM-11, SA
		ID.RA-5: Threats, vulnerabilities, likelihoods, and impacts are used to determine risk	· **COBIT 5** APO12.02 · **ISO/IEC 27001:2013** A.12.6.1 · **NIST SP 800-53 Rev. 4** RA-2, RA-3, PM-16
		ID.RA-6: Risk responses are identified and prioritized	· **COBIT 5** APO12.05, APO13.02 · **NIST SP 800-53 Rev. 4** PM-4, PM-9
	Risk Management Strategy (ID.RM): The organization's priorities, constraints, risk tolerances, and assumptions are established and used to support operational risk decisions.	**ID.RM-1:** Risk management processes are established, managed, and agreed to by organizational stakeholders	· **COBIT 5** APO12.04, APO12.05, APO13.02, BAI02.03, · **ISA 62443-2-1:2009** 4.3.4.2 · **NIST SP 800-53 Rev. 4** PM-9
		ID.RM-2: Organizational risk tolerance is determined and clearly expressed	· **COBIT 5** APO12.06 · **ISA 62443-2-1:2009** 4.3.2.6.5 · **NIST SP 800-53 Rev. 4** PM-9
		ID.RM-3: The organization's determination of risk tolerance is informed by its role in critical infrastructure and sector specific risk analysis	· **NIST SP 800-53 Rev. 4** PM-8, PM-9, PM-11, SA-14
			· **CCS CSC** 16

Example: a door lock, retina scanner, camera and access control software are four controls which might be placed at ingress/egress points in a facility to provide multiple controls.

In the example:

- the door lock and retina scanner are *preventative*, *physical* controls.
- The retina scanner is also a *technical*, *compensating* control.

Security Controls

In the example:

- The camera is a *detective and preventive, physical, deterrent* control.

In the example:

- The access control software is *technical, preventative and detective.*

A Brief Risk Management Overview

Risk Management: A few definitions

Risk Management is the process of identifying, measuring, and mitigating/transferring risk...

1. **Asset:** Something of value. May exist in physical, digital, or any other space.
2. **Threat:** A risk to an asset, man-made or natural...
3. **Vulnerability:** A weakness absent a control.
4. **Risk:** The likelihood of a threat occuring.
5. **Threat Modeling:** A process to identify threats and vulnerabilities from an attacker's point of view.
6. **Control:** Discussed previously; a control is a countermeasure which mitigates or transfers risk.
7. **Qualitative Assessment:** A scenario-driven risk analysis exercise which identifies risk based on subjective loss (i.e. public relations issues, etc.)
8. **Quantitative Assessment:** An objective risk analysis exercise based on the value of an asset, likelihood of impact, cost of event, cost of protective controls, etc.

Risk Management

Many of the decisions an organization makes will be based on qualitative (subjective/perceived) or quantitative (financial) risk assessment.

Risk Management

An organization should create a risk register, which tracks risks to the organization.

Risk Management

An organization may choose to:

- Accept a risk
- Transfer a risk (i.e. Insurance)
- Mitigate a risk (i.e. controls)

Qualitative Risk Assessment

For each triad member (C, I, A), consider the impact to each asset… Example:

Asset	Confidentiality	Integrity	Availability
Public Documentation	Low Impact	High Impact	Medium Impact
Source Code	High Impact	High Impact	High Impact
User Table w/ PW Hashes	High Impact	High Impact	High Impact

Quantitative Risk Assessment

Quantitative risk assessment relies on quantitative data, such as:

- Asset value
- Cost of impact (single loss expectancy)
- The annual frequency of the threat

Quantitative Risk Assessment

Discussion of quantitative risk is beyond the scope of this presentation.

Pro Tip: Learn the equations, but… consider using software that automates these calculations.

Quantitative Risk Assessment

Discussion of quantitative risk is beyond the scope of this presentation.

Research for additional details:

https://resources.infosecinstitute.com/category/certifications-training/cissp/domains/security-and-risk-management/cissp-risk-management-concepts

https://www.pmi.org/learning/library/quantitative-risk-assessment-methods-9929

Web Application Review

Modern Web Application, Many Components

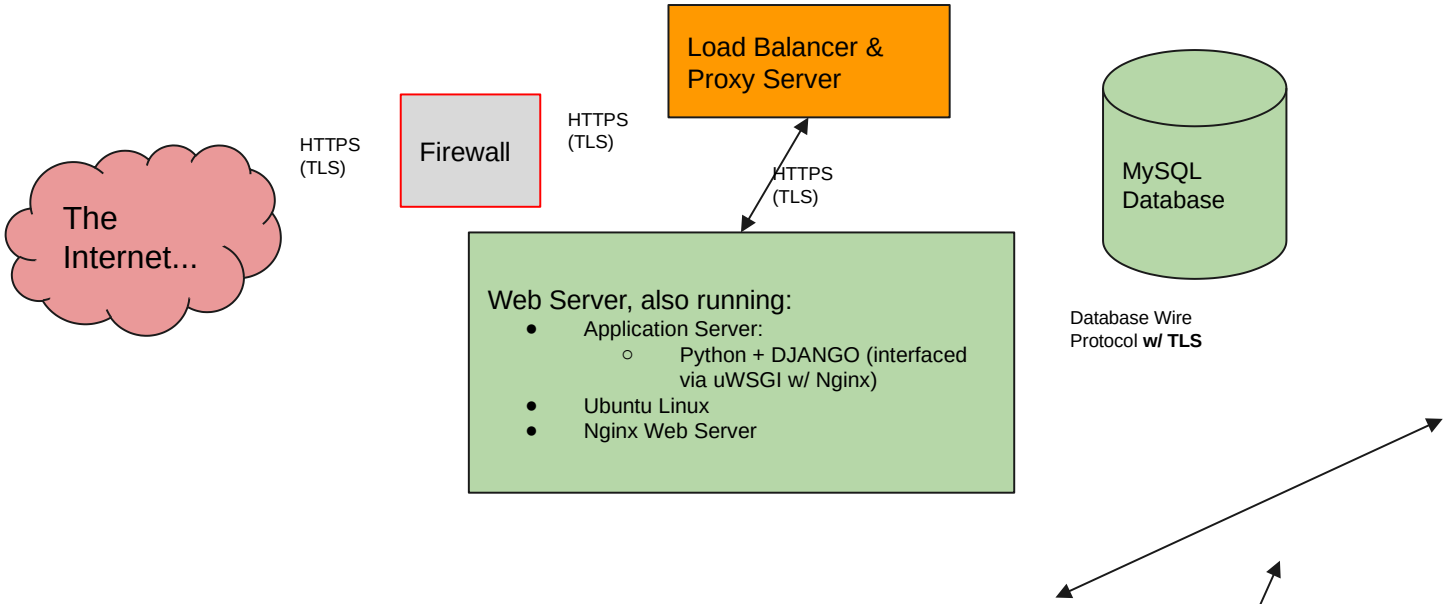
A modern web application may use many different components internally and on the backend...

1. **Web Server:** Software which speaks, at a minimum, the HTTP and/or HTTPS protocols. Usually linked to TLS stack.
2. **Web Application:** An application which provides input and output using the HTTP protocol. May not necessarily "speak" HTML; i.e. API/REST interfaces...
3. **SQL Database:** A relational database server which uses the "SQL" language for DML and DDL. MySQL, PostgreSQL, and SQLite use SQL.
4. **"NoSQL" Database:** A colloquial term for databases which don't use a SQL dialect. Common systems include key/value stores, column-oriented, document database, etc.
5. **Distributed Database:** A database which distributes resources across a network. Cassandra is a popular distributed database.
6. **Queue/IPC/Message Busses/Gossip:** Queues, distributed interprocess communication systems, gossip protocols, and message busses are widely used in distributed systems.
7. **Proxy Servers:** Used to both insulate backend systems and offer features like accelerated caching. May also handle edge TLS termination.
8. **Load Balancer:** A load balancer distributes traffic to multiple backend systems. These may also double as proxy servers in some implementations.

Controls should be evaluated for each individual component. Watch out for nebulous components like Gossip Protocols which may be embedded in distributed systems.

A *very* simple web application stack

A modern web application may use many different components internally and on the backend...



Modern Web Applications

Web applications have transitioned from stateless, transactional applications to single page web applications (SPA). A single page application uses Javascript to manipulate the DOM.

Modern Web Applications

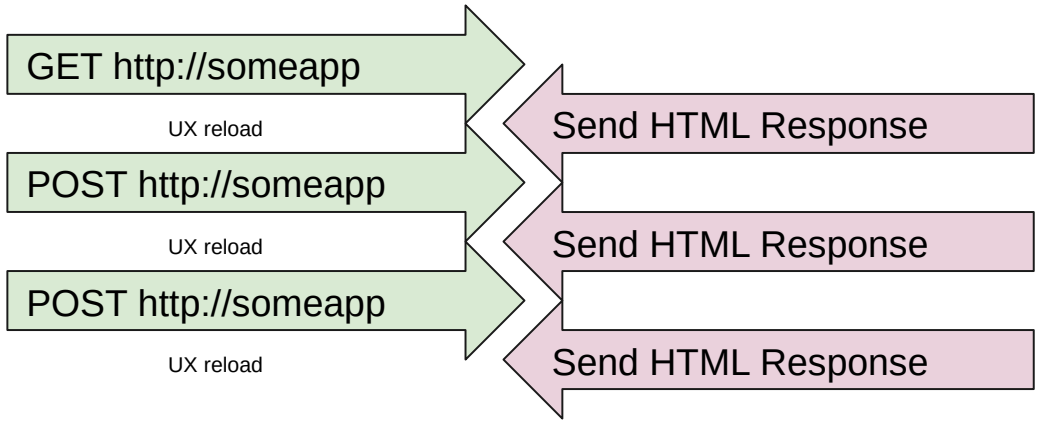
Single page applications usually provide a transitionless, stateful UI experience for the end user. These applications typically use XHR calls to execute transactions in the background.

XHR stands for XmlHttpRequest, which is the name of the Javascript object class. Be advised; XmlHttpRequet *is not* just limited to the HTTP protocol *and* is not limited to XML!!

The application server handing the XHR requests will usually use a RESTful pattern, and use JSON or other encoding as a method of serializing data for transfer.

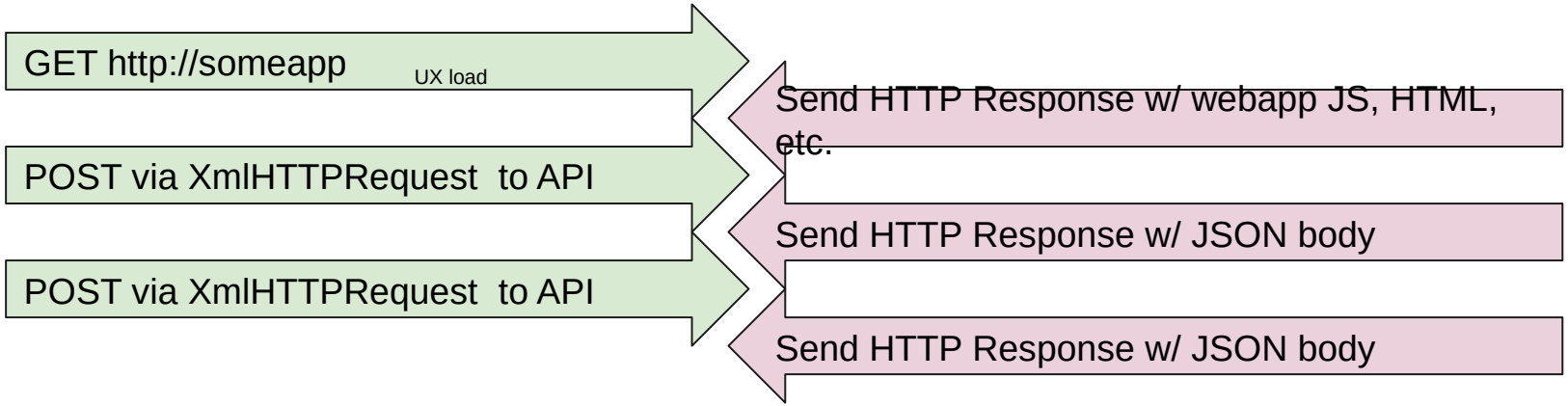
Traditional Web Application Transaction

A traditional web application experiences multiple page reloads during the normal course of operation. State is traditionally maintained via cookies, but some do use HTML5 Web Storage.



Single Page Web Application Transaction

A single page web application has the ability to load its user interface artifacts on the first page load. Subsequent requests for data are via XHR, the DOM is modified via Javascript, and the UX does not reload.



Modern Web Applications

Web applications and APIs tend to be exposed, and are at high risk of exploitation. Resources like OWASP's TOP 10 (covered later) discuss these risks and countermeasures.

Modern Web Applications

Use techniques like ***threat modeling*** to identify weaknesses in systems and controls.

Recommended Reading:

Threat Modeling: Designing for Security by Adam Shostack

Cookies and HTML5 Web Storage

HTTP is a stateless protocol...

HTTP - the protocol that drives the web - is itself stateless between requests. This means that the protocol itself does not provide the means to establish persistent sessions, or to persist data, without the user agent's (browser, for example) involvement.

There are, however, ways to persist state using techniques such as cookies and HTML5 web storage.

Two ways to persist state

1

Cookies:
A cookie is a value which is set by the server via a header, and is returned on subsequent request to relevant pages via a header. This allows for persistence of small amounts of data, like session identifiers.

Cookies have several security touchpoints, namely that they *may* be restricted to only be returned to encrypted (HTTPS) endpoints, and that they may be qualified to a domain or subdomain. **Importantly, an HTTP-only cookie is not directly accessible by Javascript via the DOM; this may be a userful property.**

2

HTML5 Web Storage:
HTML5 web storage is a newer API available for applications not requiring backwards compatibility with legacy user agents. It provides persistence within the user agent beyond that of a cookie.

HTML5 web storage has the benefit of being a true local storage API which is easily accessible and manipulated by Javascript. Note that while cookies are commonly vulnerable to CSRF, HTML5 web storage may be vulnerable to XSS when handled improperly in Javascript code.

How do cookies work?

Transaction Step	Server	Client (User Agent)
1. **Server sets the cookie by sending a response header "Set-Cookie"**	**Send Header via HTTP response to set the cookie:** Set-Cookie: <name>=<value>[; <Max-Age>=<age>] [; expires=<date>][; domain=<domain_name>] [; path=<some_path>][; secure][; HttpOnly]	**Stores the cookie** in the user agent's "cookie jar"
2. **Client makes request to server for web page or resource**	**Receives header from client on HTTP request:** Cookie: <name>=<value> [;<name>=<value>]... The application has access to these values via the web server/app API	**Sends the cookie** header to the client with the previously set cookie value: Cookie: <name>=<value> [;<name>=<value>]...

How does HTML5 Web Storage work?

API Call	Server	Client (User Agent)
1. **Server sets the data by calling API; i.e.:** localStorage.setItem("<key>", "<value>");	**No transactional involvement! The server may send the code to the page, and that code may do something *on the client...* i.e. perhaps as a session identifier for API calls from a SPA.**	**Stores the data** in the appropriate storage pool (local or session)
2. **Client makes request to server for web page or resource:** somevar = localStorage.getItem("<key>");	**No transactional involvement! The server may send the code to the page, and that code may do something *on the client...* i.e. perhaps as a session identifier for API calls from a SPA.**	**Retrieves the data** in the appropriate storage pool, and returns it to the caller of the API (i.e. to the Javascript that called the localstorage API)

When to use one or the other?

There is no right or wrong answer... it depends on your use case.

Cookies	HTML5 Web Storage
● You need to support user agents that don't support HTML5 web storage ● The data is mostly used server-side ● You need to prevent Javascript access (i.e. HttpOnly flag)	● You need a key/value database which is accessible from the Javascript API ● You are writing a single page web application which uses XHR callbacks (i.e. Javascript SPA) ● The data is mostly used client-side

- Client side data can be manipulated by the client!
  - It is the responsibility of the application's developer to design the use case securely.

- Techniques to prevent local manipulation of data include:
  - Use of cryptographic techniques like HMAC validation of cookie values.

- Techniques to prevent local manipulation of data include:
  - Use of cookies or web data as session ID only; store session state on backend and lookup by ID.

- Security issues exist in both solutions
  - Cookies are vulnerable to CSRF
  - Cookies not set secure will be transmitted over plaintext (HTTP)
  - Loose scope can result in leakage

Cookies and HTML5 Web Storage

- Consider combining with technologies like JSON Web Tokens, which provide some additional controls around the weaknesses of both Cookies and HTML5 web storage.


Cryptography

Cryptography

Cryptography … is the practice and study of techniques for secure communication in the presence of third parties called adversaries.

Excerpt from: https://en.wikipedia.org/wiki/Cryptography (accessed 2018-08-21)

Cryptography

Must face public scrutiny and survive in the light of day… trust no algorithm which is not public, and which has not faced extensive, widespread scrutiny by experts in the field of cryptanalysis…

Cryptography

Crypto red-flags:

- Can't see the source code
- Not published, not scrutinized
- Created during a hackathon at your startup

A few important terms...

1. **Plain text:** The original message, prior to any cipher being applied to it. The message before encryption.
2. **Cipher:** An algorithm (set of mathematical operations) which is used to perform encryption and decryption.
3. **Ciphertext:** The encrypted output of a cipher.
4. **Cryptanalysis:** Studying algorithms to break them... examples include plaintext recovery without key, hash collisions, etc.
5. **Key:** A piece of information that determines the output of cryptographic algorithms.
6. **Code:** The use of code word substitution. *Note that this is used in discussion of cryptography differently than casual use.*
7. **Steganography:** Hiding the *existence* of the message; a common example is hiding a message inside of image data.
8. **Block Cipher:** A cipher which operates on blocks of data of a fixed length, versus a...
9. **Stream Cipher:** A cipher which operates on a single byte of data at a time.


Cryptographic
Primitives...

Cryptographic Primitives

Cryptographic primitives are well-established, low-level cryptographic algorithms that are frequently used to build cryptographic protocols for computer security systems.

From: https://en.wikipedia.org/wiki/Cryptographic_primitive (accessed 2018-08-21)

A few common Cryptographic Primitives

Limited time; there are more! Below are core primitives with[selected examples] in brackets:

1. **Symmetric Key Cryptographic Algorithms:** use the same key for encryption and decryption. [AES]
2. **Asymmetric (public-key) Cryptographic Algorithms:** use pairs of keys - one for encryption, one for decryption. [RSA]
3. **Cryptographically Secure One-way Hash Functions:** transmutes arbitrary data to a fixed-length series of bits, such that it is irreversible and not subject to faults like collisions. [SHA-256]
4. **Authentication:** the process of *confirming* the identity of an entity by establishing the truth of a provided attribute. [TOTP]
5. **Digital Signature Algorithms:** validates the authenticity of data. [DSA, RSA]
6. **Cryptographically Secure Random Number Generators:** generates "random" numbers which meet the properties needed for secure cryptography, i.e. they are not deterministic. [PRNG in Linux kernel w/ entropy inputs, see: https://eprint.iacr.org/2012/251.pdf]

Combining cryptographic primitives allows for the creation of ***cryptographic systems***. An example of a cryptographic system is TLS, which is used when visiting secure websites.

Cryptographic Primitives

This is a *very* brief introduction to *core* cryptographic primitives and is not comprehensive. Investigate *all* the cryptographic primitives thoroughly to understand their application in the field.

Cryptographic Primitives

Symmetric Encryption

Cryptographic Primitives: Symmetric Encryption Algorithms

A *symmetric encryption algorithm* uses a single, shared key for both encryption and decryption. AES, DES, 3DES, and RC4 are all examples of symmetric algorithms.

Cryptographic Primitives: Symmetric Encryption Algorithms

- Tend to be faster than public key algorithms (i.e. TLS after handshake)
- Can operate in block or stream modes
- A block cipher *may* operate like a stream cipher, i.e. CTR mode…

Cryptographic Primitives

Asymmetric Encryption

Cryptographic Primitives: Asymmetric (Public Key) Encryption Algorithms

An *asymmetric encryption algorithm* uses one key for encryption, and another key for decryption. These keys are referred to as a "Key Pair".

Cryptographic Primitives: Asymmetric (Public Key) Encryption Algorithms

- The *public key* may be shared freely, and may be used to encrypt messages.
- The *private key* must remain secret, and is used to decrypt messages encrypted with the public key.

Cryptographic Primitives: Asymmetric (Public Key) Encryption Algorithms

- Algorithms typically based on the properties of mathematical problems that do not have efficient solutions… that is, no efficient solution by *today's* capabilities.

Cryptographic Primitives: Asymmetric (Public Key) Encryption Algorithms

- Capabilities vary by cryptosystem…
  - RSA: Encryption and signatures, key exchange
  - Diffie-Hellman (DH): Key exchange
  - And many many more…

Cryptographic Primitives

One-Way Hashes

Cryptographic hash algorithms are a hash function with properties making them useful in cryptography…

Cryptographic Hash Algorithms

- Deterministic; a mathematical function
- One-way: Can't get back to input from output w/o a brute force.
- No collisions: No two messages should ever have the same output.

Cryptographic Hash Algorithms

- Commonly used in…
  - Digital signature algorithms
  - Message digests and checksums
  - Message authentication codes (MAC)

Cryptographic Hash Algorithms

- Examples…
  - MD5 (broken)
  - SHA-1 (broken)
  - SHA-2 (SHA-256 and SHA-224)
  - Others…. RIPEMD-160, etc.

Cryptographic Primitives

Authentication & Related Techniques

Authentication (and related techniques)

- Authentication is the process of verifying the identity of an entity
  - A broad definition
    - Can be applied to physical attributes *and* digital attributes.

Authentication (and related techniques)

- Can be applied to physical attributes *and* digital attributes.
  - The store checks my ID when I use my credit card.

- Can be applied to physical attributes *and* digital attributes.
  - The web browser checks that the certificate was issued to the domain, and that it's issued by a trusted CA.

Authentication (and related techniques)

- Can be applied to physical attributes *and* digital attributes.
  - The retina scanner in the data center allows authorized personnel into the cage.

- Can be applied to physical ~~attributes~~ *tokens* (something you have).
  - A *TOTP token* like Google Authenticator could be used as a ***second factor*** of authentication.

- Can be applied to *something you know*.
  - A website asks users to select a password only they know
  - An ATM requires a PIN# to use the card.

Authentication (and related techniques)

- Can be used to authenticate a message
  - Using Message Authentication Code algorithms, like
    - HMAC
      - Data integrity + authenticity

- An HMAC can be applied programmatically to do things like…
  - Authenticating cookie values
  - Authenticating session IDs
  - Protecting APIs via signed requests

Authentication (and related techniques)

- A factor is a layer of authentication
  - Multiple factors enhance posture
    - Something you *have* is a factor
    - Something you *are* is a factor
    - Something you *know* is a factor

Cryptographic Primitives

Digital Signatures

Digital Signatures

- A mathematical algorithm which authenticates a digital message ("document")
  - Could be used to implement "electronic signatures"

Digital Signatures

- Based on asymmetric cryptography primitives
  - Non-repudiation w/o private key disclosure
  - Confirms *integrity*, not altered.

Digital Signatures

- Common algorithms
  - RSA
  - DSA, ECDSA
  - ElGamal
- Used in systems like GPG, SSH, etc.

Digital Signatures

- Not the same as an HMAC
  - Radically different properties
  - Different methods of operation
  - Digital signatures rely on asymmetric cryptography!


Examining a
Crypto-System

Examining a Crypto System

- Something challenging...
  - TLS is both important to understand as well as complex.
    - An enhancement of SSL v3
      - Not compatible with SSL v3

Examining a Crypto System

- SSL/TLS versions
  - SSL v1: Broken, never released
  - SSL v2: Broken, publicly released
  - SSL v3: Fixes issues with v2, broken…

Examining a Crypto System

- SSL/TLS versions
  - TLS v1.0: Weak, MITM issues…
  - TLS v1.1: OK; allows old algorithms
  - TLS v1.2: Good - introduces GCM mode, good AES support

- SSL/TLS versions
  - TLS v1.3: Excellent
    - Downgrade protection
    - Enhanced performance *and*
    - Full handshake is signed

Examining a Crypto System

- Pro Tips...
  - Just because a version of a protocol is old, you may still need to find ways to securely support it when needed. Examples: Older mobile devices in common use!

- Pro Tips...
  - You may be able to make older versions of TLS more secure based on what protocol options you configure server side.

Examining a Crypto System

- Pro Tips...
  - Visit resources like Qualys SSL Labs: https://www.ssllabs.com/ssltest/ to test various configurations.

Examining a Crypto System

- Numerous cryptographic primitives:
  - Asymmetric encryption
    - Key exchange
    - Digital certificates
    - Digital signatures

Examining a Crypto System

- Numerous cryptographic primitives:
  - Symmetric encryption
    - Once keys are exchanged, symmetric encryption like AES is used for speed.

Examining a Crypto System

- Numerous cryptographic primitives:
  - Symmetric encryption
    - HMAC may be used for block and stream ciphers in some modes

Examining a Crypto System

- Numerous cryptographic primitives:
  - Symmetric encryption
    - Authenticated Encryption (AEAD) may be used for stream ciphers

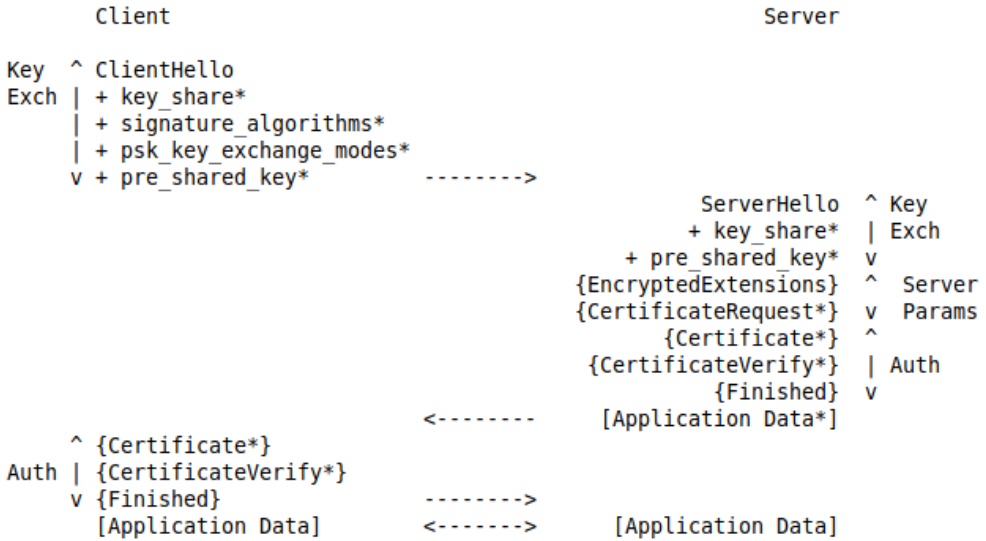
Examining a Crypto System

- Numerous cryptographic primitives:
  - Authenticated Encryption as used in TLS produces a MAC from the plaintext, and then includes the MAC in the ciphertext.

Examining a Crypto System

- Other crypto-systems include:
  - Bitcoin and other digital ledgers
  - GPG
  - SSH
- Try reading the RFCs and doing your own analysis!

The TLS negotiation process (TLS v3)




SEGMENT TWO


OWASP's
Top 10

OWASP Top 10

- OWASP is the "Open Web Application Security Project"
- A well known and established non-profit membership organization dedicated to web security.

OWASP Top 10

- In addition to other useful software projects like ZAP and WebGoat, they provide a list of "Top 10" security issues prevalent in web applications.

OWASP Top 10

- Understand these vulnerabilities as well as your biographer would understand your life history… they are some of the most common ways attackers gain *persistence*!

OWASP's TOP 10 Web Vulnerabilities

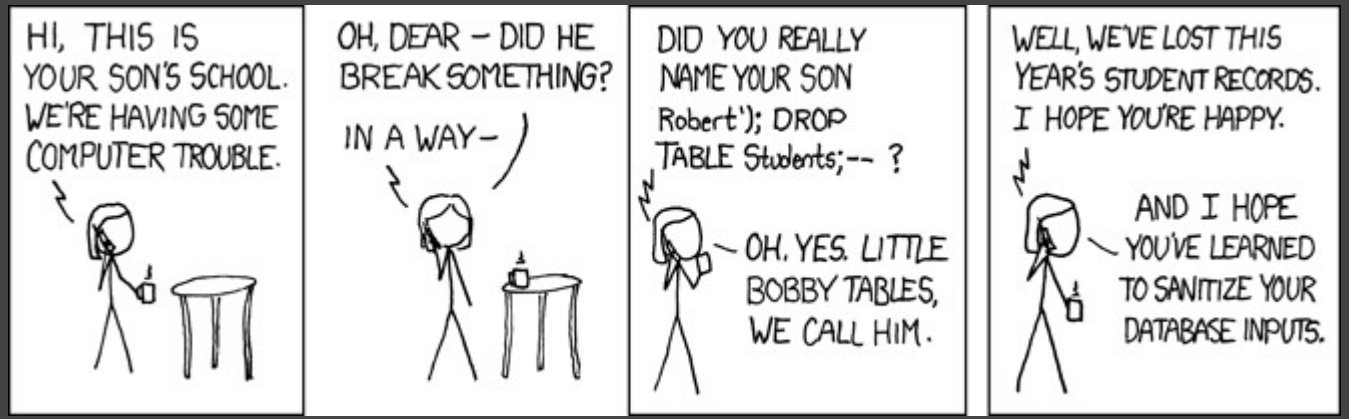
1. Injection Attacks
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Controls
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components w/ Known Vulns
10. Insufficient Logging & Monitoring

OWASP Top 10: Injection (A1:2017)

- Injection flaws
  - Occur when untrusted input is not properly sanitized, and executed as code. Most common variant publicly discussed is "SQL injection".

OWASP Top 10: Injection (A1:2017)

- Injection flaws



Source: https://xkcd.com/327/

OWASP Top 10: Injection (A1:2017)

- Injection flaws
  - Not just SQL; impacts most interpreted languages. XSS can also be considered a form of an injection attack, but it occurs client side.

OWASP Top 10: Injection (A1:2017)

- Injection flaws - red flags
  - "Exec" calls - that is, the dynamic interpretation of input - are one of several dangerous callouts which significantly increase risk.

- Injection flaws - red flags
  - Python example - what does this code do?

```
myexec = 'print "super safe code"'
exec(myexec)
```

OWASP Top 10: Injection (A1:2017)

- Injection flaws - red flags
  - Python example - what does this code do?

```
mycode = request.POST.get("usercode", "")
exec('print "' + mycode + '"')
```

OWASP Top 10: Injection (A1:2017)

- Injection flaws - red flags
  - Non-Parameterized SQL queries...

```
awesomesql = "SELECT *
FROM username WHERE uid="
+ request.POST.get["uid"]
```

OWASP Top 10: Injection (A1:2017)

- Injection flaws - red flags
  - Any use of exec
  - Any spawning of processes, etc. with untrusted input as parameters
    - Don't do it!!!!!!!!!!!!!!!!!

OWASP Top 10: Broken Authentication (A2:2017)

- Broken Authentication
  - Not checking session validity
    - Not checking session validity consistently can result in authentication bypass.

OWASP Top 10: Broken Authentication (A2:2017)

- Broken Authentication
  - Not rotating session IDs
    - Session fixation attack risk
    - Sequential/deterministic/guesss-able session IDs

OWASP Top 10: Broken Authentication (A2:2017)

- Broken Authentication
  - Default credentials
  - Lack of failed login protection
    - Allows for brute force or credential stuffing

OWASP Top 10: Broken Authentication (A2:2017)

- Broken Authentication
  - Insecure credential storage
    - Passwords should be salted, peppered, hashed - not stored plaintext. Proven algorithms like bcrypt exist and are ubiquitous!!

OWASP Top 10: Broken Authentication (A2:2017)

- Broken Authentication
  - Session ID in URL or query segment
    - Results in logging at proxies
    - Can result in session takeover

OWASP Top 10: Broken Authentication (A2:2017)

- Broken Authentication
  - Failure to remove session after logout
  - Failure to log out *all* sessions after password change

OWASP Top 10: Sensitive Data Exposure (A3:2017)

- Sensitive Information Disclosure
  - Insecure credential storage
    - Use something like scrypt, bcrypt, PBKDF2 or salted+peppered+hashed output.

OWASP Top 10: Sensitive Data Exposure (A3:2017)

- Sensitive Information Disclosure
  - Sensitive data stored in plain text
    - Consider using a data dictionary
    - Business can determine classification of each data entity

OWASP Top 10: Sensitive Data Exposure (A3:2017)

- Sensitive Information Disclosure
  - Plain-text data transport
  - Storing more than necessary and/or for longer time than needed.
    - If it doesn't exist, I can't steal it!

OWASP Top 10: XML External Entities (A4:2017)

- XML has native functionality which allows external entities to be sourced during XML processing.
  - An XXE may be easily weaponized as an XSRF attack for pivot and recon.

OWASP Top 10: XML External Entities (A4:2017)

- What does this code do?

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</foo>
```

OWASP Top 10: XML External Entities (A4:2017)

- What does this code do?

```
<!ENTITY xxe SYSTEM "https://192.168.1.1/private" >]>
```

  - FYI - This is an XSRF attack in addition to an XXE!

OWASP Top 10: XML External Entities (A4:2017)

- What does this code do?

```
<!ENTITY xxe SYSTEM "file:///dev/random" >]>
```

OWASP Top 10: Broken Access Controls (A5:2017)

- Broken Access Controls
  - Not checking object access
    - Example: Bob changes the invoice ID parameter from 1 to 2 and sees a different customer's data

OWASP Top 10: Broken Access Controls (A5:2017)

- Broken Access Controls
  - Privilege Escalation
    - Example: Alice changes a cookie value from User to Admin, and gets superuser access.

OWASP Top 10: Broken Access Controls (A5:2017)

- Broken Access Controls
  - To prevent, be consistent with access controls!
    - Check access at every request for each object

OWASP Top 10: Broken Access Controls (A5:2017)

- Broken Access Controls
  - To prevent, be consistent with access controls!
    - Deny by default for all but public resources

OWASP Top 10: Broken Access Controls (A5:2017)

- Broken Access Controls
  - To prevent, be consistent with access controls!
    - Configure the web server properly

OWASP Top 10: Broken Access Controls (A5:2017)

- Broken Access Controls
  - To prevent, be consistent with access controls!
    - Don't deploy SCM artifacts to public hosts (i.e. exclude .git!)

OWASP Top 10: Broken Access Controls (A5:2017)

- Broken Access Controls
  - To prevent, be consistent with access controls!
    - Sign API calls to avoid manipulation

OWASP Top 10: Broken Access Controls (A5:2017)

- Broken Access Controls
  - To prevent, be consistent with access controls!
    - Apply rate limiting to make attack tooling more challenging.

OWASP Top 10: Security Misconfiguration (A6:2017)

- Security misconfigurations may result in vulnerabilities
  - Minimize footprint - disable or uninstall unused features
  - Default accounts removed/locked

OWASP Top 10: Security Misconfiguration (A6:2017)

- Security misconfigurations may result in vulnerabilities
  - Configure web servers to be secure
  - Block access to non-production artifacts

OWASP Top 10: Security Misconfiguration (A6:2017)

- Security misconfigurations may result in vulnerabilities
  - Segmented architecture
    - Make pivoting hard
  - Automate testing, deployment
    - Devops & Devsecops = Friend

OWASP Top 10: Security Misconfiguration (A6:2017)

- Security misconfigurations may result in vulnerabilities
  - Use tools like Ansible, Terraform, etc. to manage local and cloud deployments *consistently*.

OWASP Top 10: Cross-Site Scripting (XSS) (A7:2017)

- XSS = Cross Site Scripting
  - A form of an injection attack
  - Usually results in code execution on the client
    - The basis of historical worms like JS.Yamanner@m and others

OWASP Top 10: Cross-Site Scripting (XSS) (A7:2017)

- XSS = Cross Site Scripting
  - Three kinds
    - Reflected XSS
    - Stored XSS
    - DOM XSS

OWASP Top 10: Cross-Site Scripting (XSS) (A7:2017)

- XSS = Cross Site Scripting
  - Reflected XSS
    - A web page "reflects" user input back into output. Malicious output allows code execution client side.

OWASP Top 10: Cross-Site Scripting (XSS) (A7:2017)

- XSS = Cross Site Scripting
  - Stored XSS
    - An application stores unsanitized user input, and then outputs it on subsequent requests without sanitization. This may result in client-side execution.

OWASP Top 10: Cross-Site Scripting (XSS) (A7:2017)

- XSS = Cross Site Scripting
  - DOM XSS
    - A client-side Javascript application passes unsanitized user input to a Javascript API, resulting in client-side code execution within the context of the DOM. HTML5 web storage is an example sensitive area.

OWASP Top 10: Insecure Deserialization (A8:2017)

- Insecure Deserialization
  - Many languages natively support serialization and deserialization
  - Convenient for the developer
  - ***Very*** vulnerable w/ untrusted input

OWASP Top 10: Insecure Deserialization (A8:2017)

- Insecure Deserialization
  - From object manipulation to full-blown RCE, depending on platform
  - Don't do it!!!!!!!!!!!!!!!!!!!!!!!!

OWASP Top 10: Insecure Deserialization (A8:2017)

- Insecure Deserialization
  - Countermeasures:
    - Signing, using HMAC et al, to prevent manipulation or tampering of client-side data

OWASP Top 10: Insecure Deserialization (A8:2017)

- Insecure Deserialization
  - Countermeasures:
    - Don't use native serialization; use common formats like JSON, protobuf, Avro, etc.

OWASP Top 10: Insecure Deserialization (A8:2017)

- Insecure Deserialization
  - Countermeasures:
    - Don't rely on client-side data for security enforcement

OWASP Top 10: Insecure Deserialization (A8:2017)

- Insecure Deserialization
  - Countermeasures:
    - Sandbox dangerous deserialization if it's absolutely necessary

OWASP Top 10: Components with Known Vulnerabilities (A9:2017)

- Using Known-Vulnerable Components
  - Outdated libraries
  - Legacy frameworks
  - Ancient hardware
  - Out of date operating systems

OWASP Top 10: Components with Known Vulnerabilities (A9:2017)

- Using Known-Vulnerable Components
  - There must be an ongoing plan to monitor for vulnerabilities in components, and to apply controls around them as needed.

OWASP Top 10: Components with Known Vulnerabilities (A9:2017)

- Using Known-Vulnerable Components
  - Maintain an inventory of all assets
  - Understand what components run on each asset
  - Understand the risk around each

OWASP Top 10: Components with Known Vulnerabilities (A9:2017)

- Using Known-Vulnerable Components
  - Move checks to the left for build pipelines - find vulnerable components before pushing to production.

OWASP Top 10: Components with Known Vulnerabilities (A9:2017)

- Using Known-Vulnerable Components
  - Don't forget that most containers use their own user space. This user space must be maintained like any other system, and isn't patched with the host OS.

OWASP Top 10: Components with Known Vulnerabilities (A9:2017)

- Using Known-Vulnerable Components
  - Consider immutable infrastructure where possible, performing a fully automated image build with newest non-vulnerable component versions on each deployment.

OWASP Top 10: Components with Known Vulnerabilities (A10:2017)

- Insufficient Logging & Monitoring
  - If you can't see it, you can't say something…

- Insufficient Logging & Monitoring
  - Red Flags:
    - No logging at all
    - App logs, but no OS logs, etc.
    - Full auth logging disabled

OWASP Top 10: Components with Known Vulnerabilities (A10:2017)

- Insufficient Logging & Monitoring
  - Red Flags:
    - Logged, but never monitored
    - Logs only stored locally
    - No real time logging/delayed logs

- Insufficient Logging & Monitoring
  - Best Practices:
    - Log all authentication attempts and sensitive operations

- Insufficient Logging & Monitoring
  - Best Practices:
    - Maintain audit logs for the application, operating system, network devices, etc.

OWASP Top 10: Components with Known Vulnerabilities (A10:2017)

- Insufficient Logging & Monitoring
  - Best Practices:
    - Consider tamper-evident logging (i.e. logs with built in digests, etc.)
      - AWS is a good example:

https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-log-file-validation-intro.html

OWASP Top 10: Components with Known Vulnerabilities (A10:2017)

- Insufficient Logging & Monitoring
  - Best Practices:
    - Logs offloaded to centralized log analysis system/host
      - Local logs = may be tampered with trivially

- Insufficient Logging & Monitoring
  - Best Practices:
    - Monitoring systems tail logs and alert on suspicious events in near real time.

Thank you.