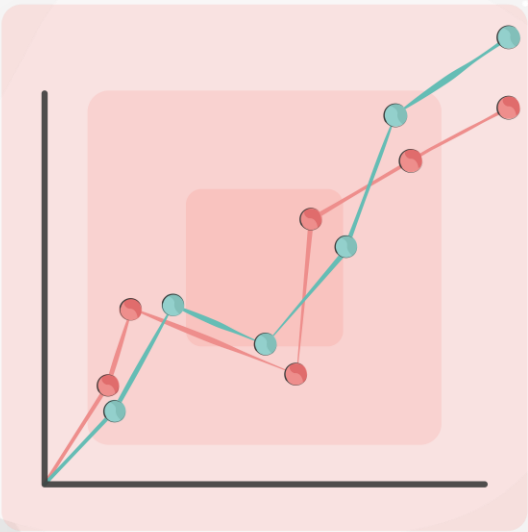
Level 5 Data Engineer
Module 4 Topic 3

Designing secure architectures

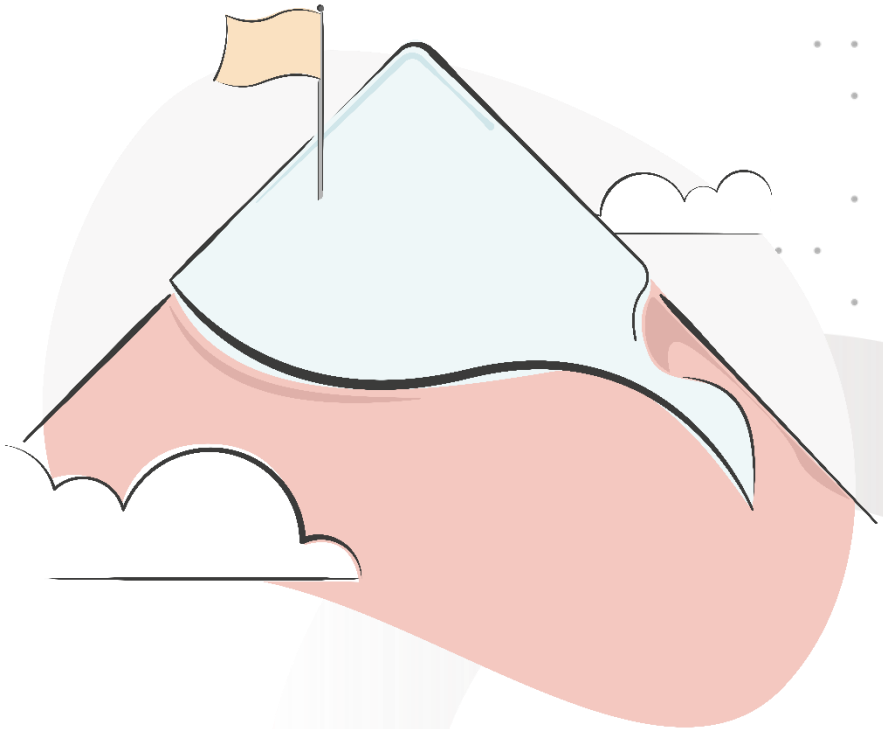
Welcome to today's webinar.



Session aim and objectives

This webinar supports the following learning outcomes:

- Implement functionality of data products and data pipelines in a way that protects data at rest and data in transit using a zero-trust approach
- Make use of advanced security techniques to safeguard data products and data pipelines against unauthorised access and security breaches.
- Practise visual methods of designing secure architectures through diagrams and flowcharts to improve data product and data pipeline communications

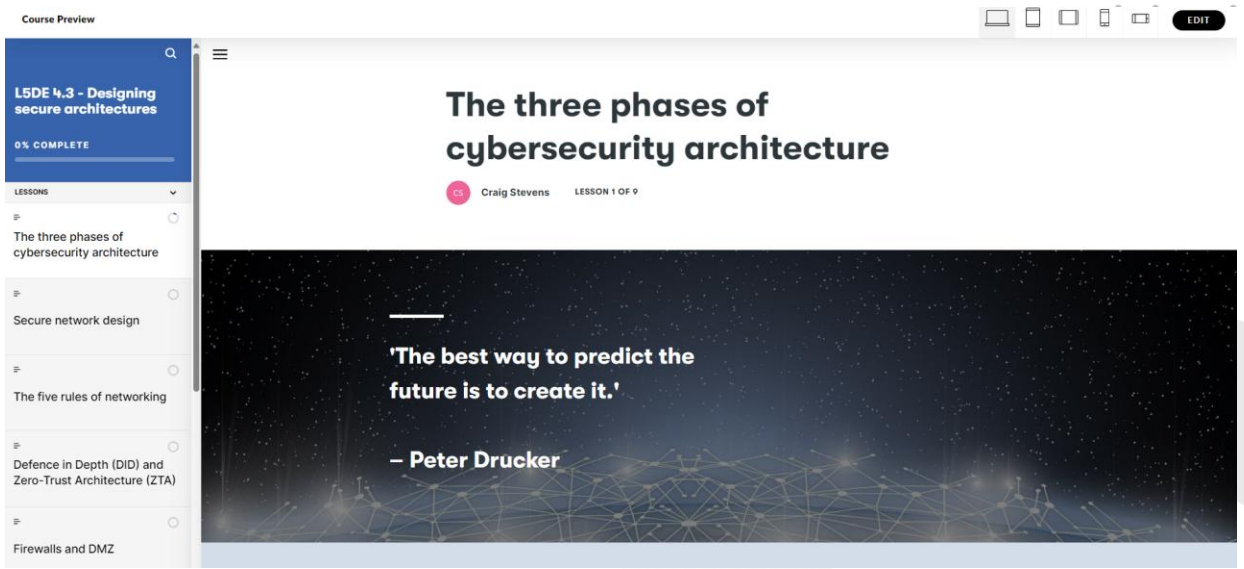


BPP

Recap of e-learning

Are you happy with your learning?

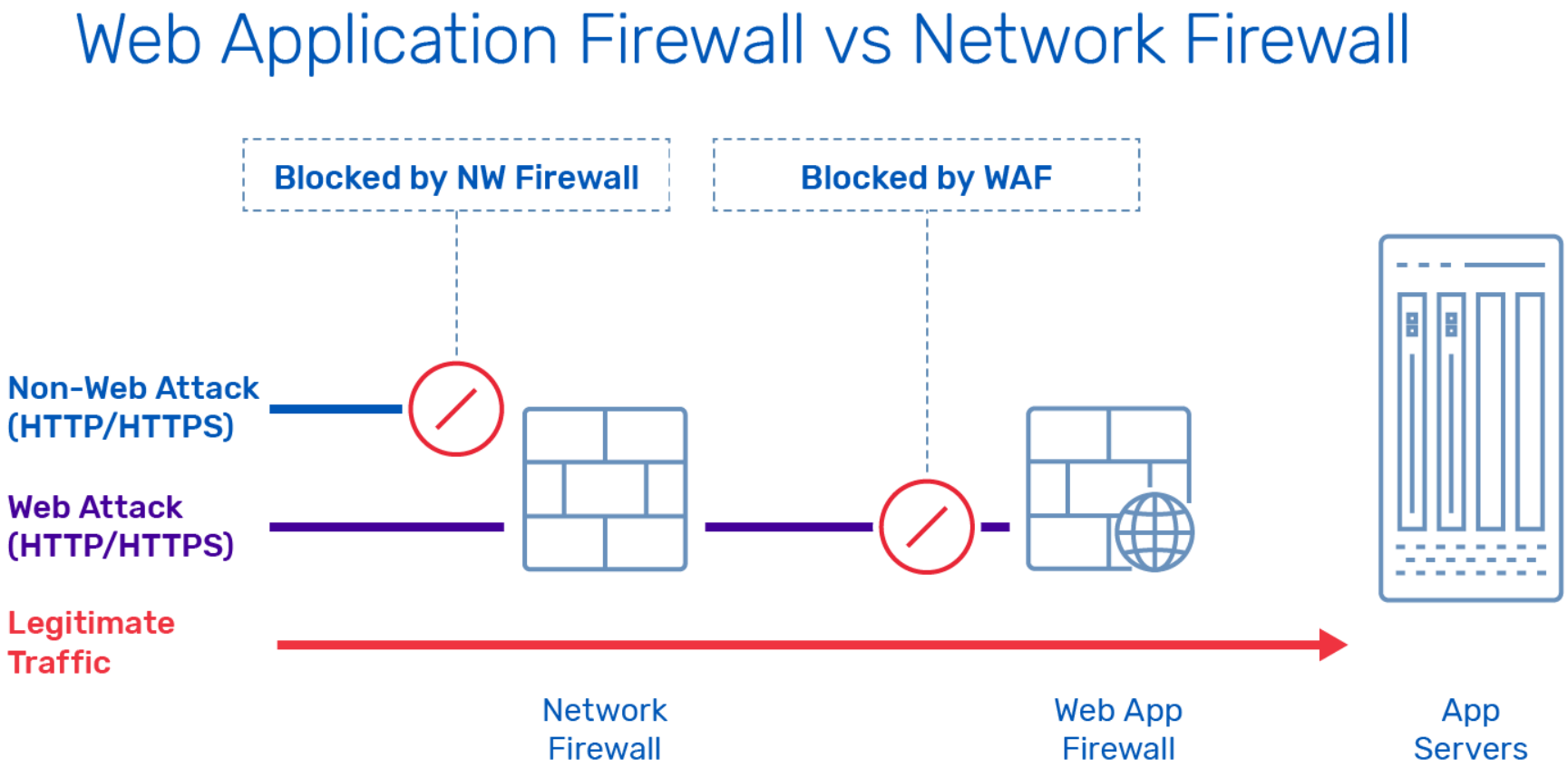
- What are the three phases of cybersecurity architecture?
- What are the five rules of networking?
- *“Implicit Trust Is Never Assumed, Always Verified”* – what do we call that?



A screenshot of topic 3 e-learning

Recap of e-learning

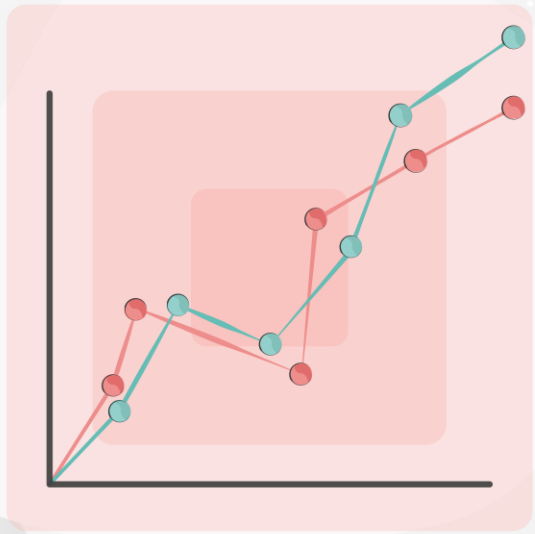
Are you happy with your learning?



Recap

can you explain the difference between authentication and authorisation?

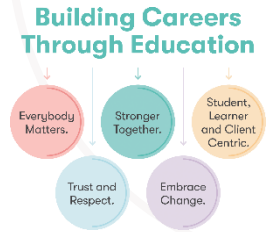
Secure by design



Useful definitions

Let’s start with the basics

- **Service**
- **Service account**
- **Externally-exposed enterprise assets**



BPP

Secure by design

What does this mean…?

- Building systems with security as a foundational and integral component from the outset, rather than as an afterthought.
- Anticipate threats and vulnerabilities during the design phase.
- Security isn’t just a phase; it's a continuous mindset.
- Proactive approach.
- Reduces long-term costs (fixing vulnerabilities later is more expensive).
- Increases trust with stakeholders and users.
- Protects user data and system integrity.

BPP

Secure by design

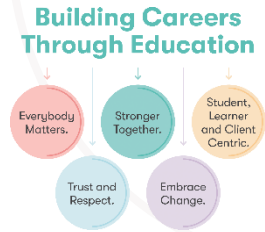
How can a house be “secure by design”?

A house built on a robust foundation with integrated alarm systems, locks, and secure windows, rather than adding locks and alarms after the first burglary has already happened.

More useful definitions

Here are some more that can be useful

- **Threat Modelling**
- **Redundancy and Resilience**
- **Security Controls**



BPP

Security control frameworks

Why…?

- Security frameworks = building regulations for digital systems
- Secure architecture protects data, access, and operations
- Without controls, systems are unstable and easily compromised

BPP

A Holistic and Evolving Discipline

- Build systems to resist attacks and failures by design
- Adapt architectures for emerging technologies (e.g. Cloud, IoT)
- Stay sharp - continuous learning is a security necessity

BPP

Two security architecture principles

Simplicity

- The simpler the system, the less can go wrong
- Minimising inconsistencies
- Easy to grasp

Restrictiveness

- Minimising access is key
- Communication should be inhibited
- Boundaries should be tight

BPP

Discussion

Voting Software

Accuvote TS

- 30k+ lines of C++

PRUI (Yee *et al*)

- Less than 300 lines of Python
- http://zesty.ca/voting/

Which one would you trust to handle voting securely?

Submit your responses to the chat!

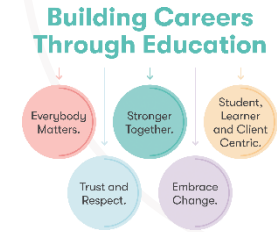
BPP

Patterns – invaluable to architects

- Patterns are reusable, proven design strategies
- Boost efficiency, reliability, and consistency
- A roadmap for building secure, scalable systems

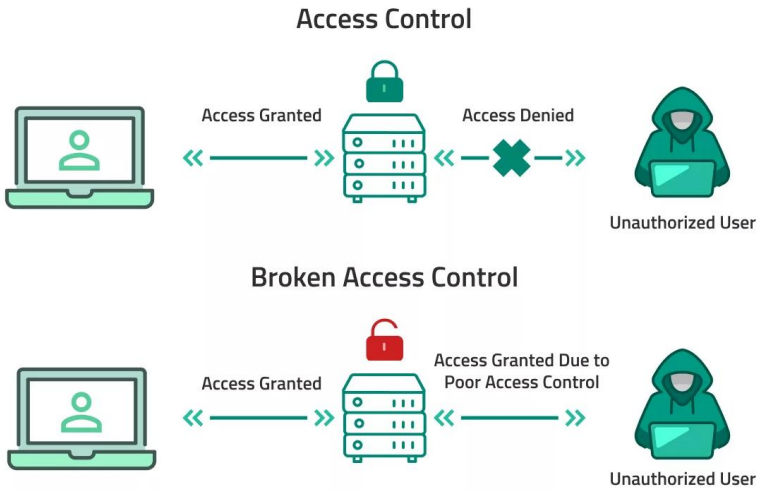
BPP

Pattern examples



Principle of Least Privilege

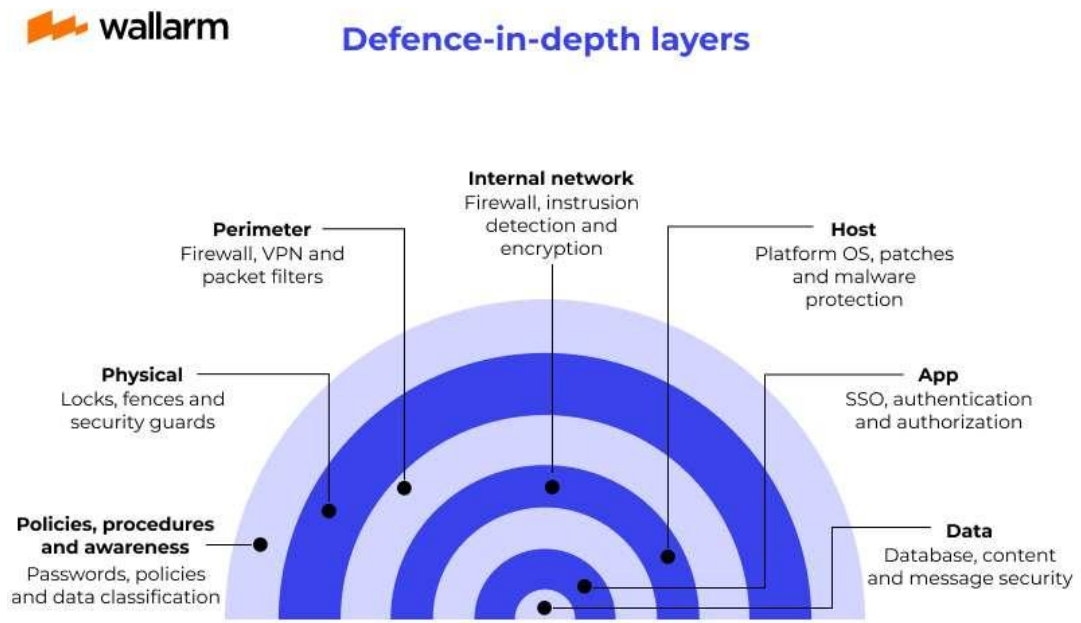
Grant only the access needed - no more



Source: SUCURi

Defence in Depth

Layer protections to reduce risk and exposure



Source: wallarm

BPP

Economy of mechanism

- Keeping mechanisms **simple** (“economical”)
- Simplicity refers to **all dimensions**: design, implementation, operation, interaction with other components, even in specification.
- The **toolkit** philosophy of the UNIX system is an example; each tool is designed and implemented to perform a single task. The tools are then put together. Think **modular** design (compare: OOP with SoC).
- This allows the checking of each component, and then their interfaces. It is conceptually much less complex than examining the unit as a whole. (Think integration vs **unit** testing).

BPP

Fail-Safe Defaults

- Access should be explicitly granted - not assumed
- Default to DENY, then allow by exception
- Failures must not create security gaps

BPP

Complete Mediation

- Every access to every object must always be checked
- *“Check every time”*
- *“Assume permissions may change in real time”*

BPP

Open Designs

- Prevents the “security by obscurity” anti-pattern
- Principles and methods of design are publicly known.
- Implementation details are not secret
- Hiding methods or data does not secure your system
- Example – GSM Algorithms designed in secret
  - A3/A8: reverse engineered (‘98), broken 3 hours later
  - A5/2: reverse engineered (‘99), broken 5 hours later
  - A5/1: reverse engineered (‘99), broken 1 year later

BPP

Separation of Privilege

- Requires at least 2 actors or components to implement a secure action
- E.g., a proposer and an approver
- You cannot approve your own proposal

Formally: *Where feasible, a protection mechanism that requires two keys to unlock it is more robust and flexible than one that allows access to the presenter of only a single key.*

BPP

Example: root / admin account

- Most operating systems use admin account
- Any privileged action requires admin privileges
- All-or-nothing access

Why is this bad?

BPP

Least Common Mechanism

Minimise the amount of mechanism common to more than one user and depended on by all users.

- *Every shared mechanism (especially one involving shared variables) represents a potential information path between users*

Is this a good principle?

- Isolation prevents communication, and communication with something - another process or a resource - is necessary for a breach of security.

BPP

Psychological acceptability

It is essential that the human interface be designed for ease of use so that users routinely and automatically accept the protection mechanisms correctly.

Security mechanisms should not add to difficulty of accessing resource:

- Ease of installation, configuration, use
- Avoid frustration and non-compliance

BPP

Some practical security principles

Principle	Explanation	Example
Secure Communication	Protect data in transit by encrypting and authenticating all communications	Data sent from a PostgreSQL database to S3 is encrypted using HTTPS and authenticated with TLS certificates
Secure Configuration	Configure systems safely by disabling defaults and removing unused components	A virtual machine is hardened by removing unnecessary services, disabling root login, and enforcing permissions
Access Control	Manage who can access what through roles, permissions, and authentication	A junior engineer receives read-only access to one folder via IAM roles and is required to use MFA
Monitoring and Logging	Track activity across systems and review logs for anomalies or incidents	CloudTrail logs all API activity, with alerts set up for unauthorised access and logs forwarded to a SIEM
Patch Management	Keep systems up to date by applying fixes for known vulnerabilities	A critical Apache vulnerability is patched across production via automated tools after testing in staging

BPP

Anti-patterns (i.e. what ***not*** to do)

Anti-patterns are common responses to recurring problems that are usually ineffective and risk being counterproductive.

- ***Awareness*:** Identifying anti-patterns helps avoid pitfalls in system design.
- ***Course Correction*:** Provides an opportunity to rectify design flaws early.
- ***Education*:** Acts as a teaching tool, enabling learning from past mistakes.

BPP

Activity discussion

Part 1

In breakout rooms, pick one pattern that you have seen implemented, or could implement in one of the systems that you work with.

- Discuss the pros and cons of implementing that pattern. Could there be any unintended consequences of applying it?

Part 2

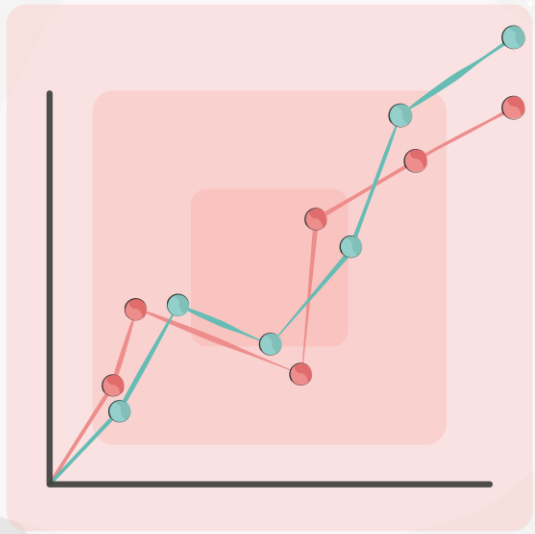
'Security by obscurity' is a common anti-pattern.

- Discuss what do you think is meant by that?
- Using your own research, find examples of that anti-pattern in action.

Submit your responses to the chat!

BPP

Secure by design



Confidentiality

- **Confidentiality – protect data from unauthorised access or exposure**
- **Use encryption protocols like SSL/TLS, SSH, VPNs, and secure management tools**
- **Design with secure communication as a baseline, not a bonus**

BPP

SSL/TLS: Securing Data in Transit

- Encrypts data for confidentiality and integrity over networks
- Mitigates threats like man-in-the-middle attacks
- Use latest TLS version & enable Perfect Forward Secrecy (PFS)

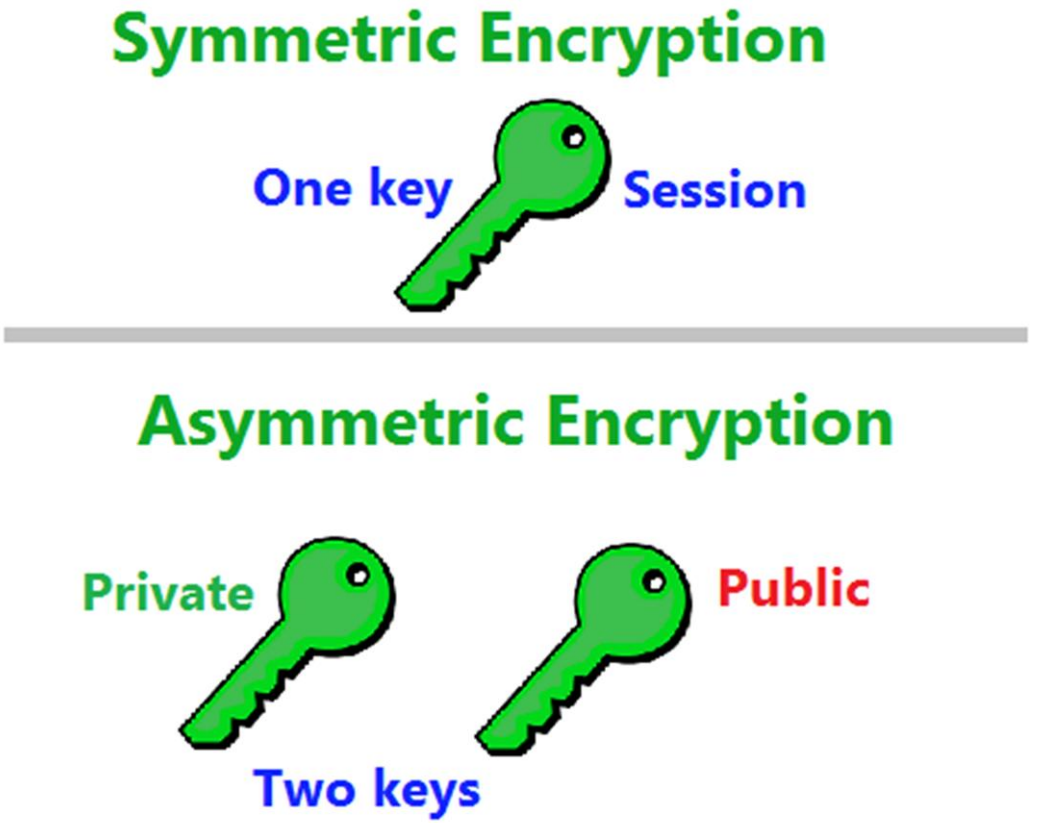
Source: WebSitePulse

BPP

SSL/TLS Encryption

Does this use symmetric or asymmetric encryption?

- Both
- Asymmetric encryption is necessary to verify the others identity
- and then symmetric encryption gets used because it's faster.

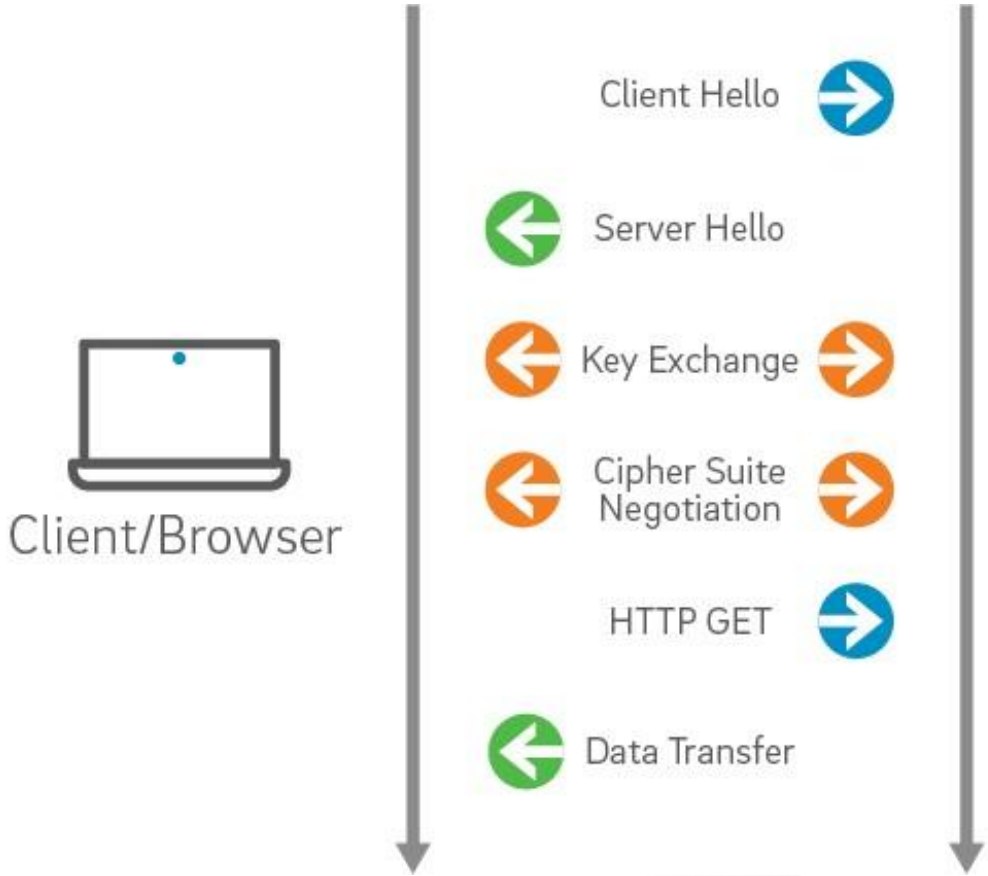


BPP

HTTPS (or HTTP Secure)

Asymmetric encryption – confidentiality

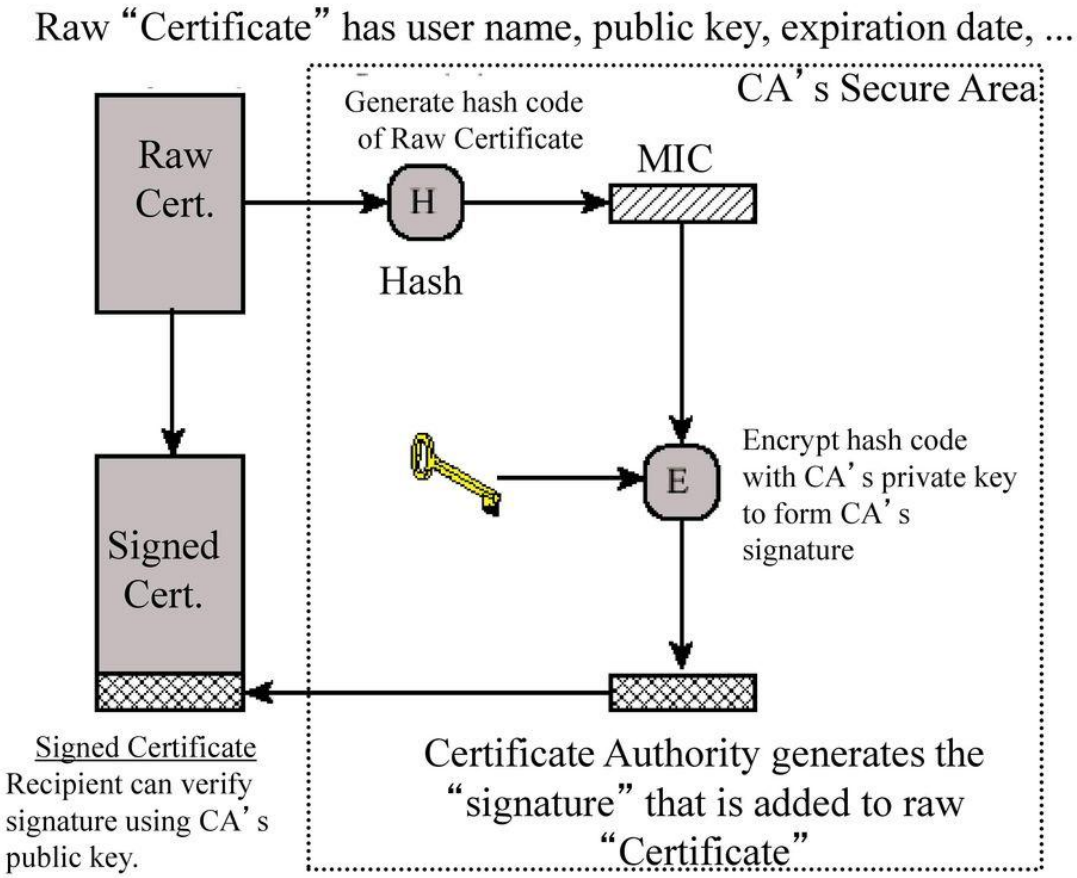
HTTPS (Hypertext Transfer Protocol Secure) is the protocol used in the communication, which combines HTTP with SSL/TLS to provide encrypted transmission of data.



X.509 Certificates

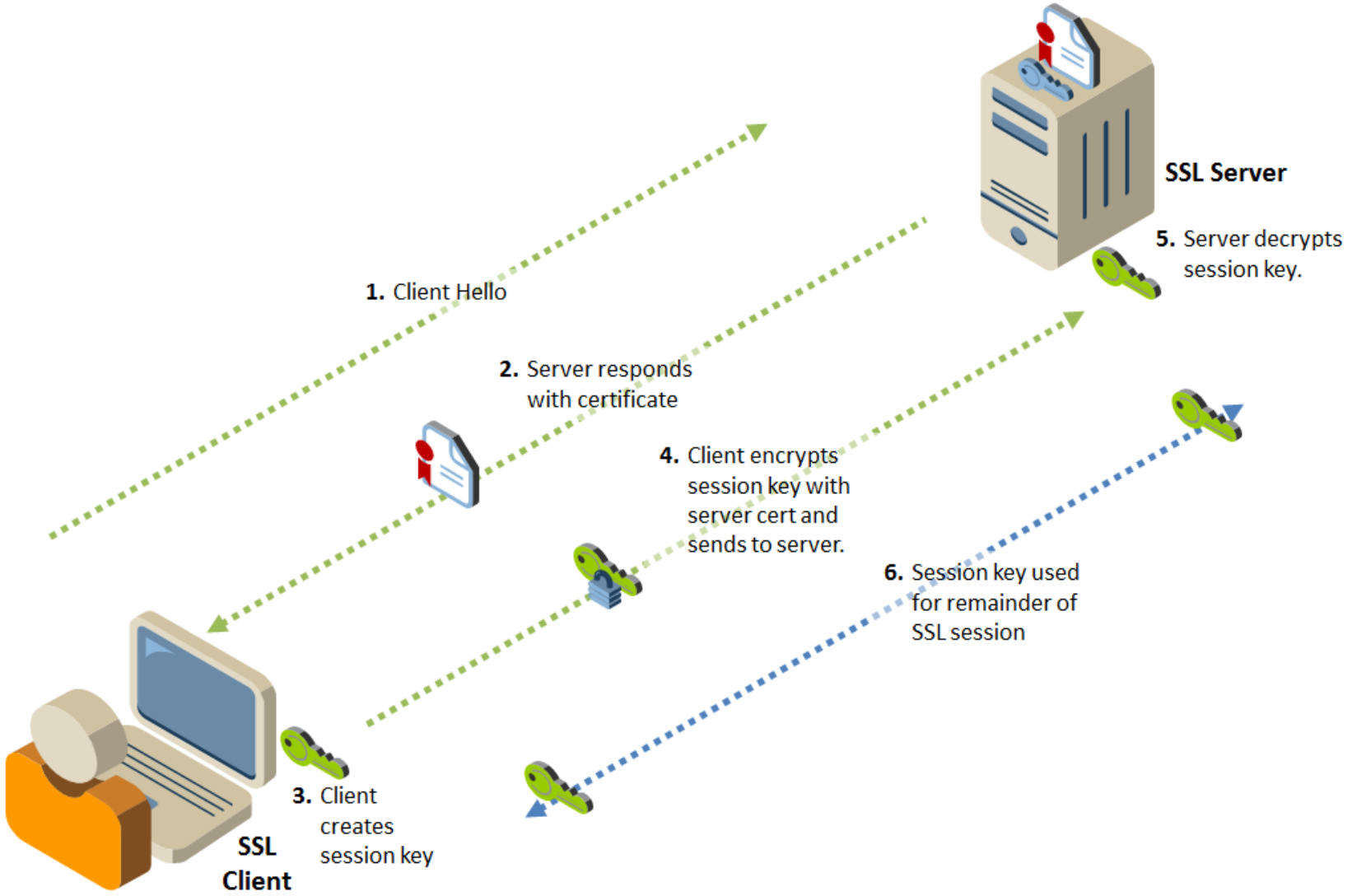
What you need to know…

- BSSL is by far the largest use of X.509 certificates, many people use the terms interchangeably.
- They're not the same however
- A "SSL Certificate" is a X.509 Certificate with Extended Key Usage: Server Authentication



Summary of SSL handshake

Portions relevant to key establishment



Downgrade attacks

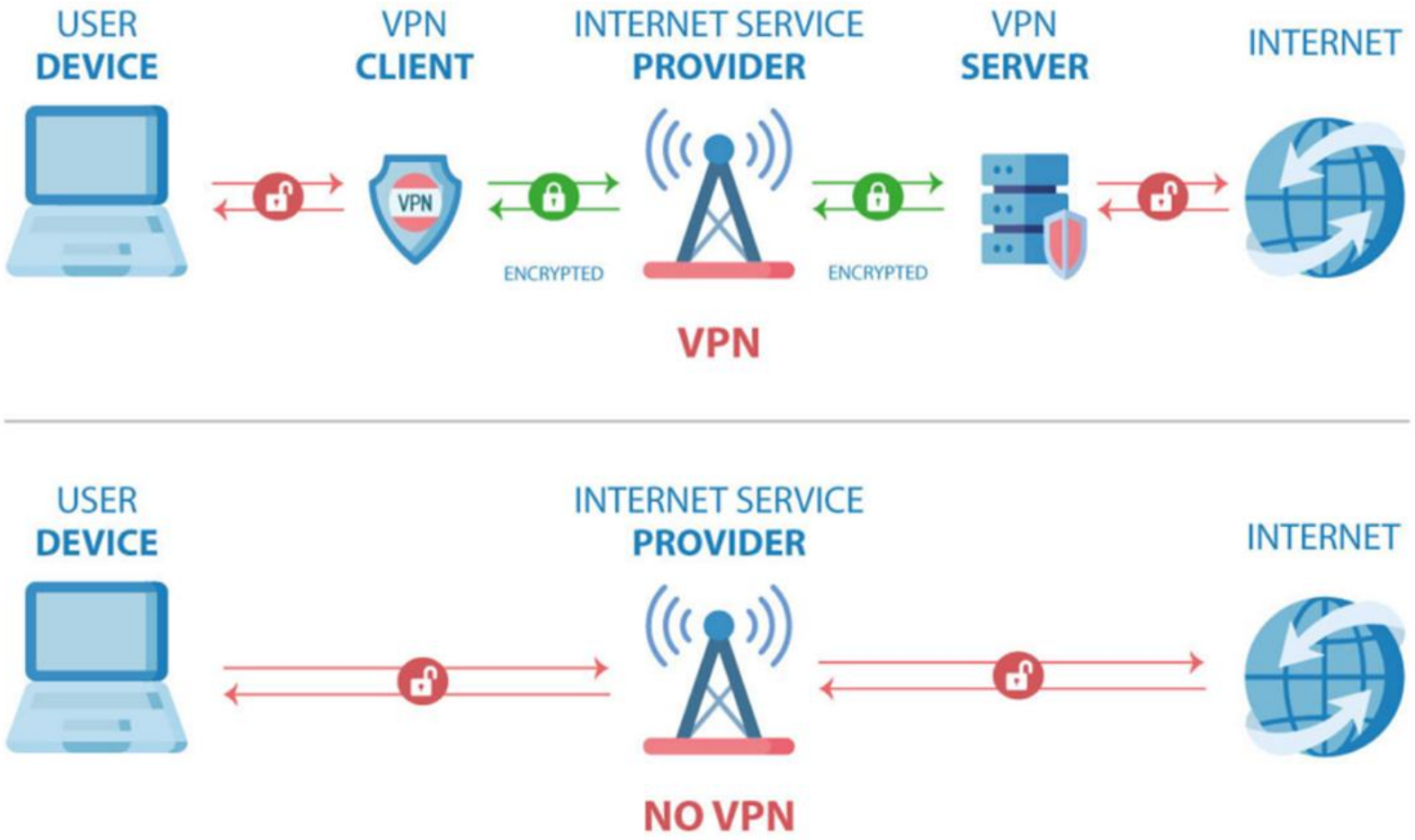
***Description*:** Forces a connection to use older, less secure versions of protocols, making them vulnerable.

***Mitigation*:**

- Use latest versions of TLS.
- Implement TLS_FALLBACK_SCSV to prevent forced downgrades.
- Monitor server logs for repeated connection attempts.

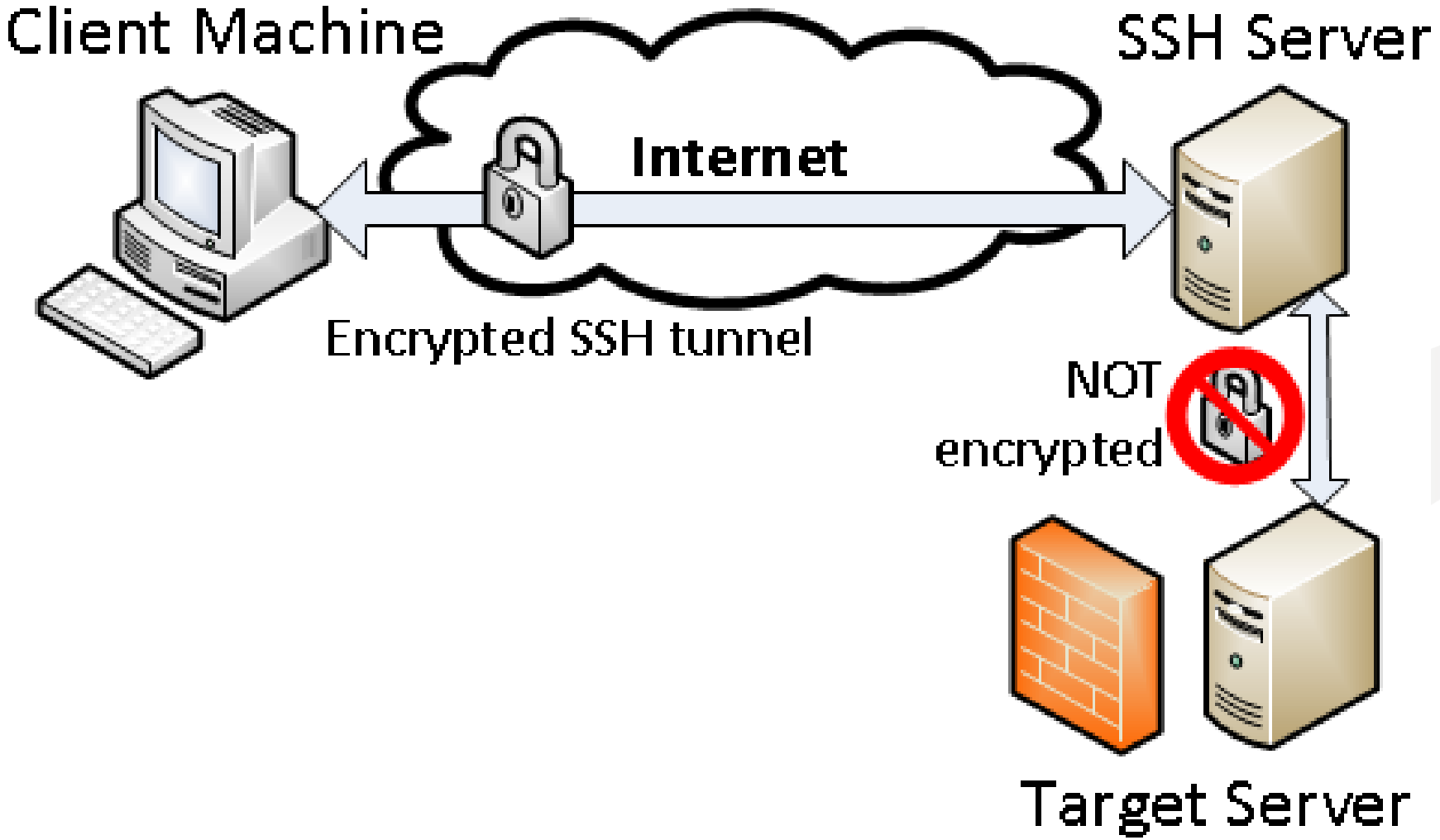
BPP

Virtual Protected Networks (VPNs)



BPP

Client and Machine Architecture

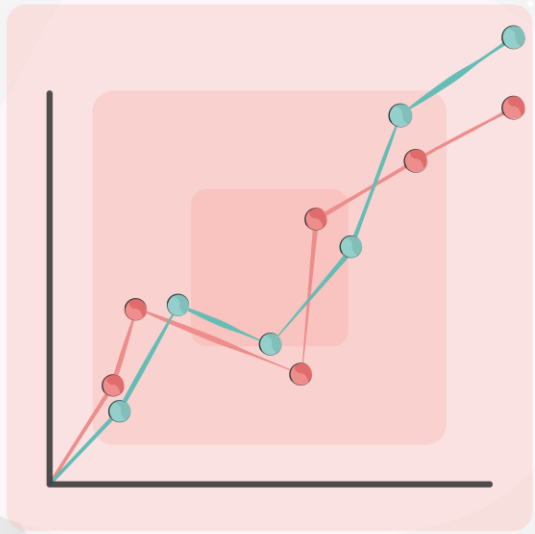


General best practices

- Regularly update software and libraries.
- Use strong, unique certificates and keep private keys secure.
- Monitor for anomalies and suspicious activity.

BPP

Vulnerabilities



OWASP TOP 5 vulnerabilities

1 - Broken Access Control

- Developers and administrators should follow the Principle of Least Privilege here.
- Access controls should be applied to APIs, and authorization checked for every request.
- Regular security audits and code reviews are a must to identify and fix access control issues, and multi-factor authentication should be enforced to limit unauthorised access.

BPP

2 - Cryptographic Failures

- Also known as ‘sensitive data exposure’
- Implement regular security testing (including code reviews and vulnerability assessments) to identify and fix cryptographic weaknesses
- Use latest versions of secure cryptographic libraries too.

BPP

3 - Injection

SQL but also other types (LDAP, XML, …)
Example:

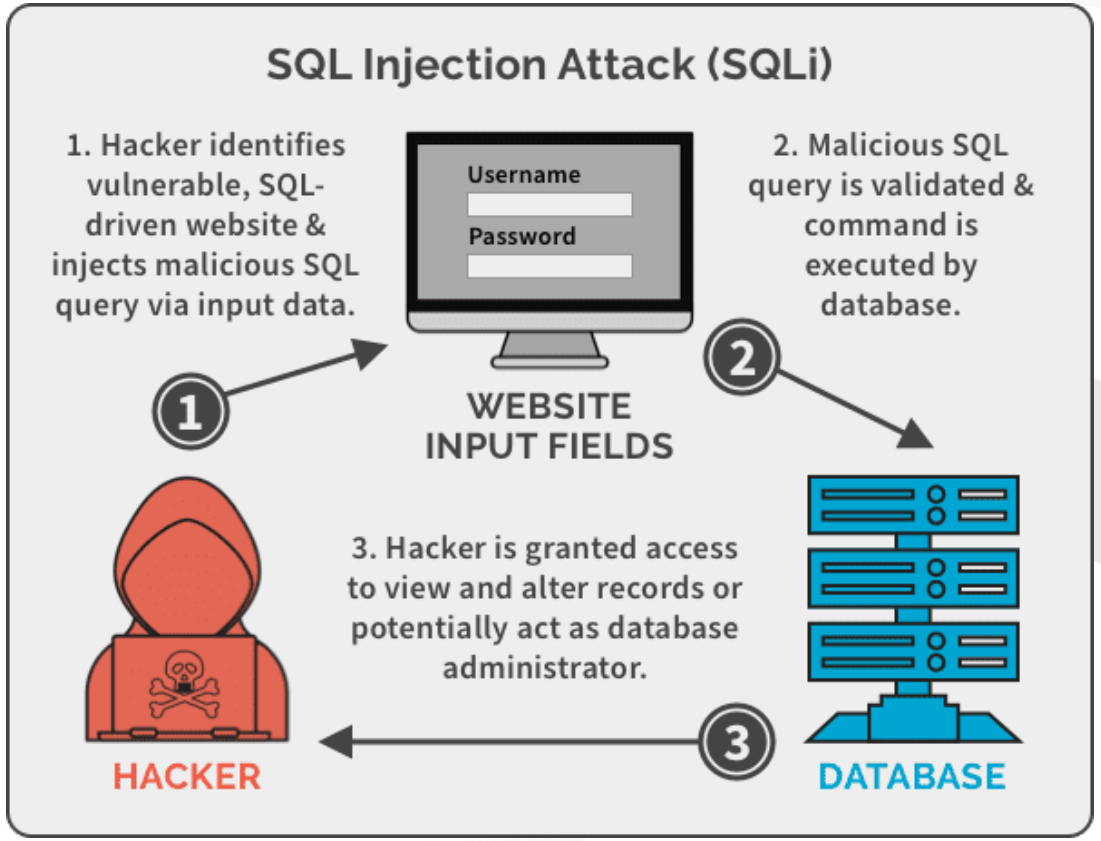
- A login form takes user input without validation:

SELECT * FROM users WHERE username = 'input' AND password = 'input';

- An attacker enters: ' OR 1=1 -- as the username
- This modifies the query to:

SELECT * FROM users WHERE username = '' OR 1=1 -- ' AND password = '';

- Since 1=1 is always **true**, the attacker gains **unauthorised access**.



4 - Insecure Design

- An example of insecure design is an app that produces overly detailed error messages.

- If it reports on error conditions in too much detail and offers diagnostic clues about the application environment, or other associated data, it could be revealing potentially useful information to attackers.

BPP

5 - Security Misconfiguration

- Unpatched vulnerabilities
- Default configurations
- Unused pages
- Unprotected files and directories
- Unnecessary services
- Use of vulnerable XML files

BPP

Activity discussion

Discuss: Which vulnerabilities could be the most critical ones in your line of work?

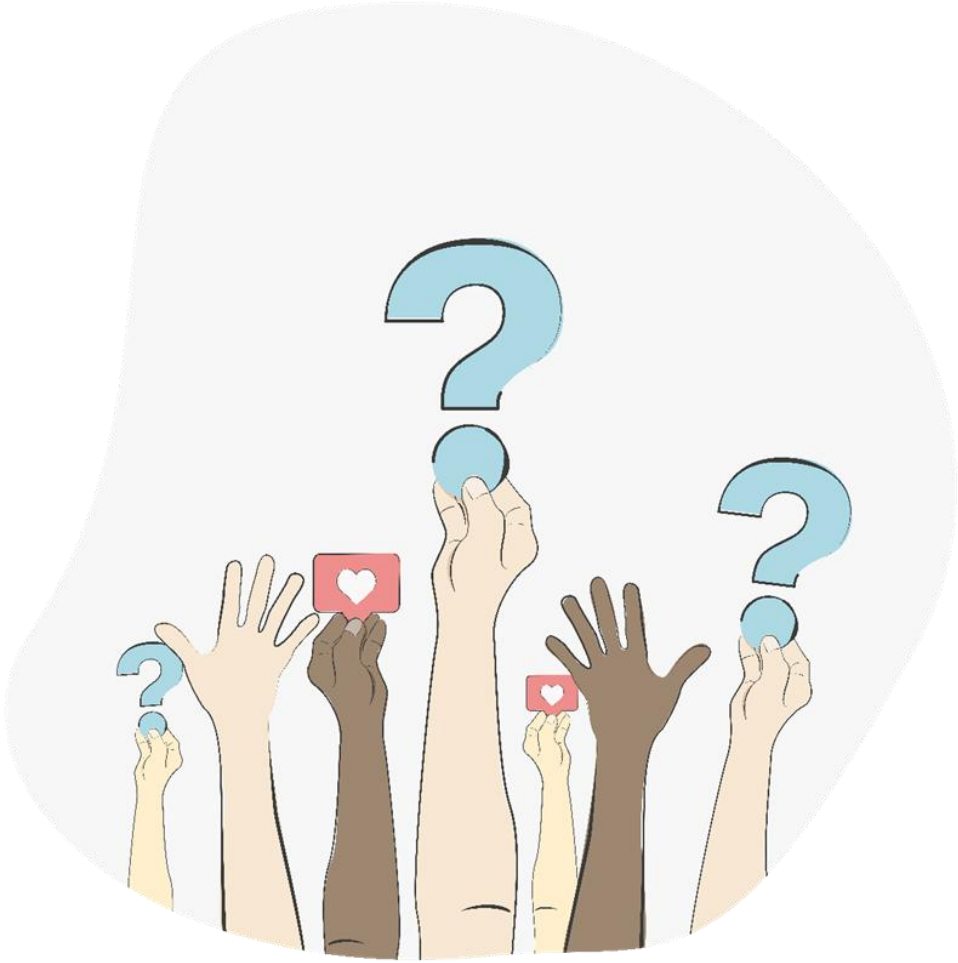
Submit your responses to the chat!

BPP

Session wrap-up

Building Careers Through Education

Everybody Matters. Stronger Together. Student, Learner and Client Centric. Trust and Respect. Embrace Change.



Any questions or feedback?

BPP