Web Application Vulnerabilities and TestREx

Daniel Ricardo dos Santos
Stanislav Dashevskyi

Outline

- Introduction to web application vulnerabilities
- TestREx project
- TestREx demo - writing automated exploits
- TestREx code

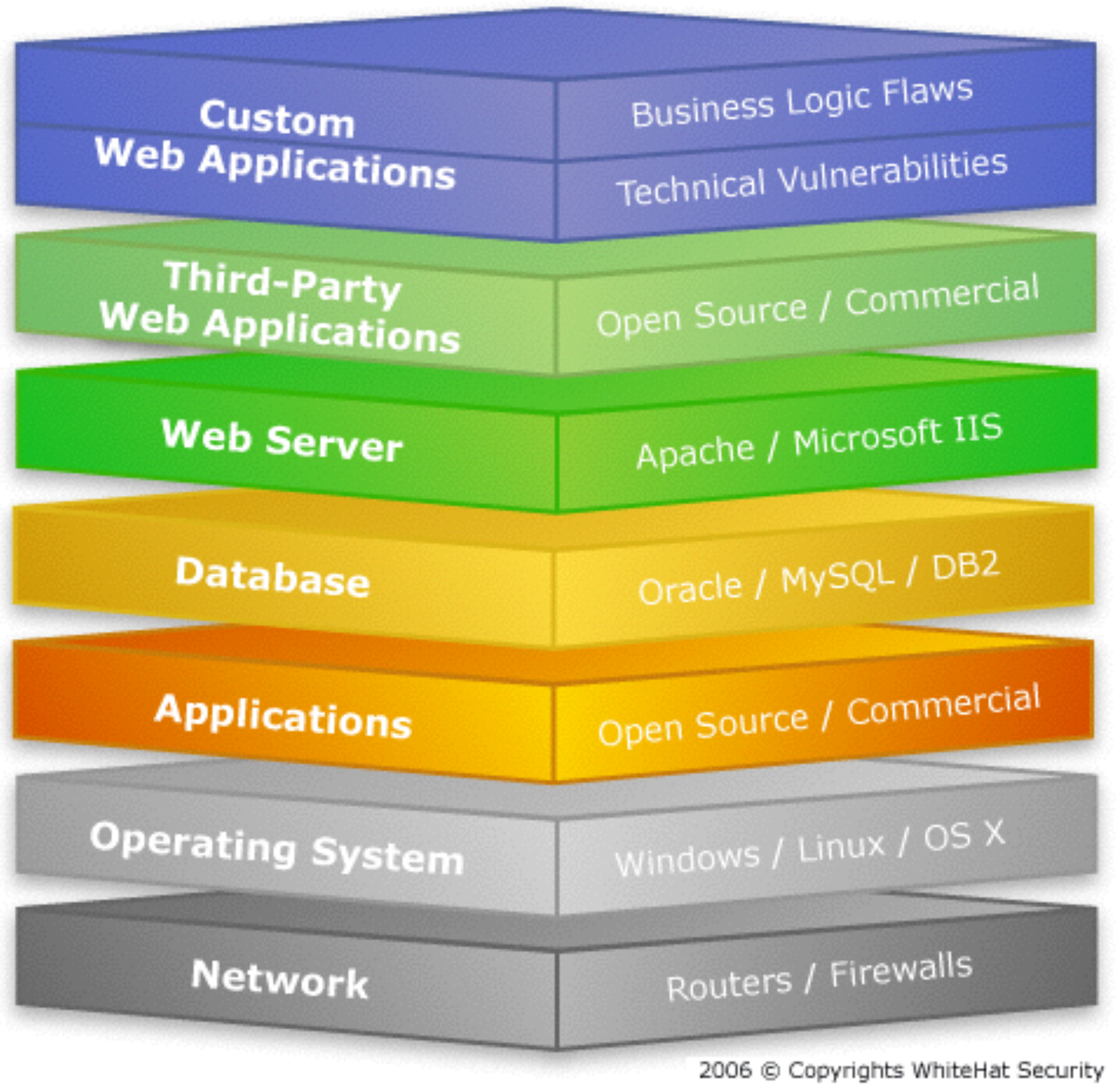
Vulnerability

- A weakness of an asset that can be exploited by one or more threats (ISO27005)
- Three elements: the system must be susceptible, the attacker must have access to the flaw and the attacker must have the capability to exploit it
- There are many kinds of vulnerabilities, some more dangerous, some less (also many ways of classifying vulns)

Exploit

- A piece of software, chunk of data or sequence of commands that takes advantage of a vulnerability
- The complexity of the exploit usually depends on the complexity of the vulnerability being exploited

Vulnerability Stack

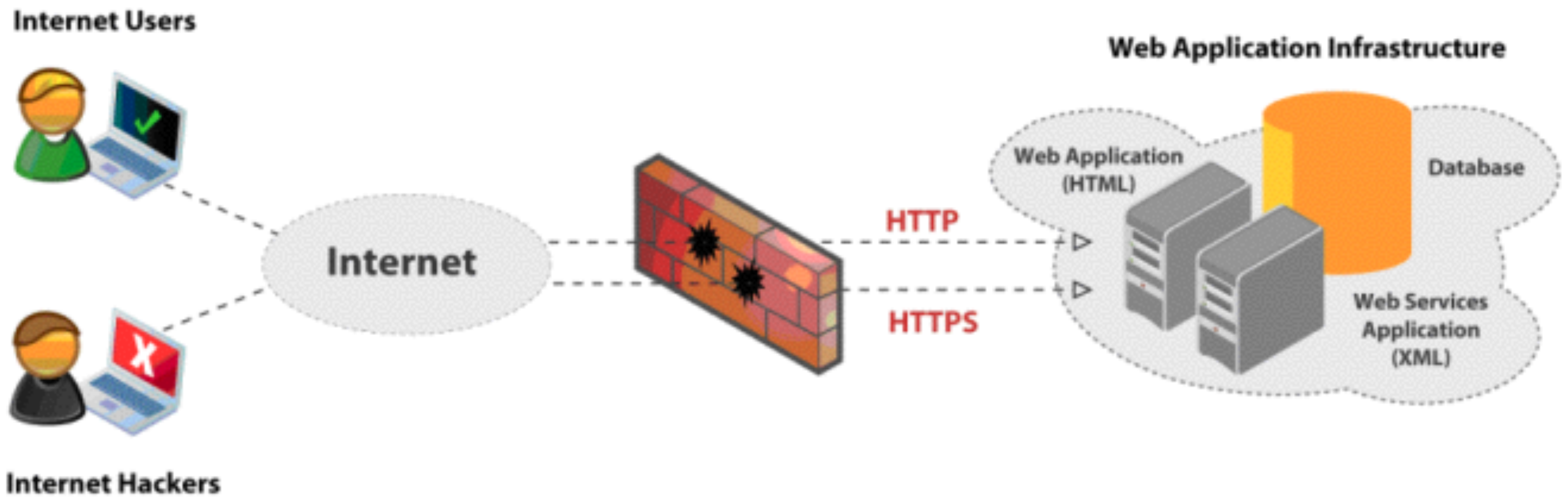


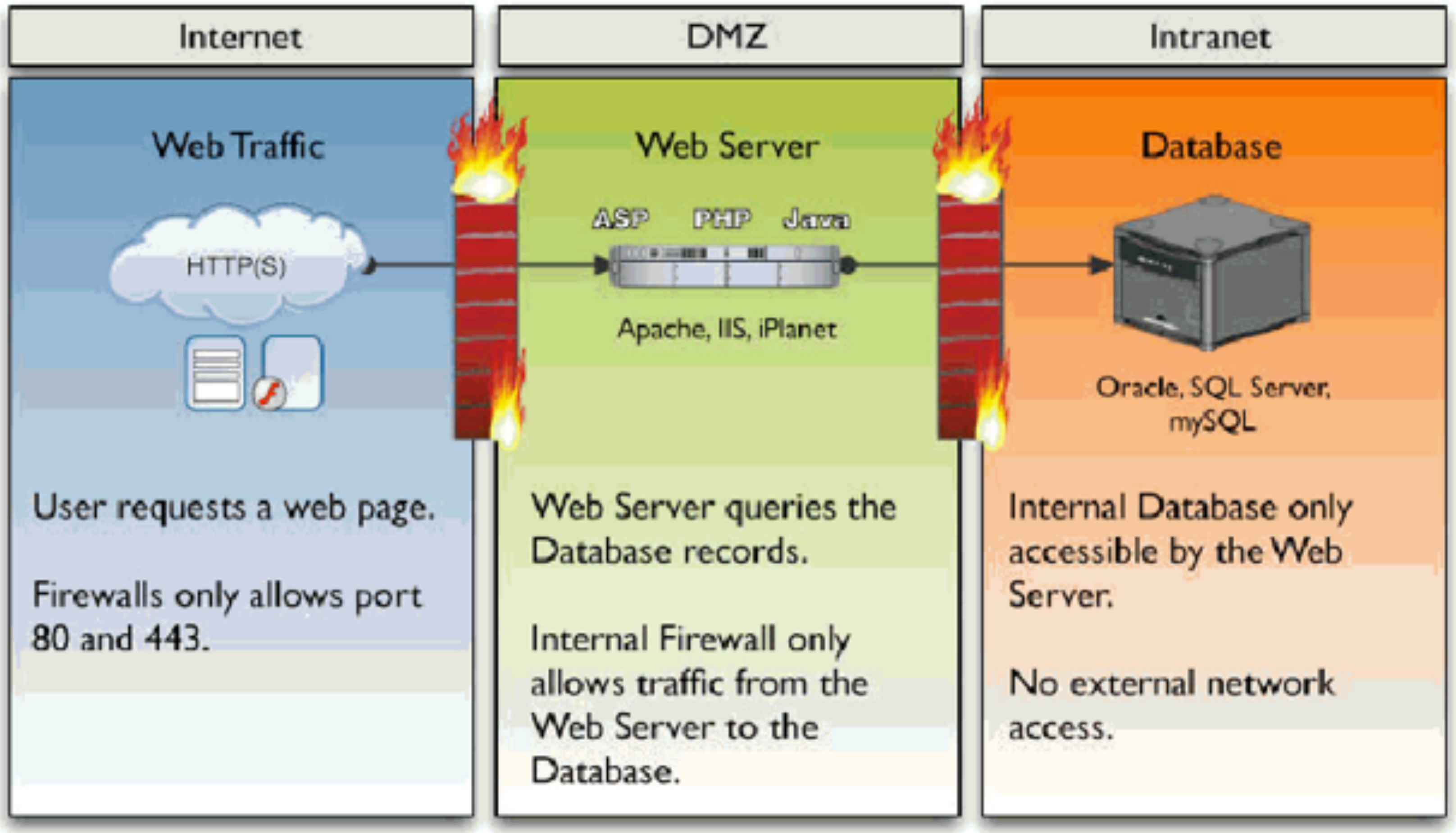
Web application exploits

- Web applications are used for everything nowadays

- They are “easy" targets:
  - They are exposed to the public and the attacker
  - They are accessible through firewalls
  - The vulnerabilities and exploits are usually less complex than on the other levels

- The severity of the exploits can range from defacing to total ownership of the target server




Internet
DMZ
Intranet
Web Traffic
HTTP(S)
Web Server
ASP PHP Java
Apache, IIS, iPlanet
Database
Oracle, SQL Server, mySQL
User requests a web page.
Firewalls only allows port 80 and 443.
Web Server queries the Database records.
Internal Firewall only allows traffic from the Web Server to the Database.
Internal Database only accessible by the Web Server.
No external network access.

Layer 1-6 security solutions are ineffective for web security

OWASP Top 10

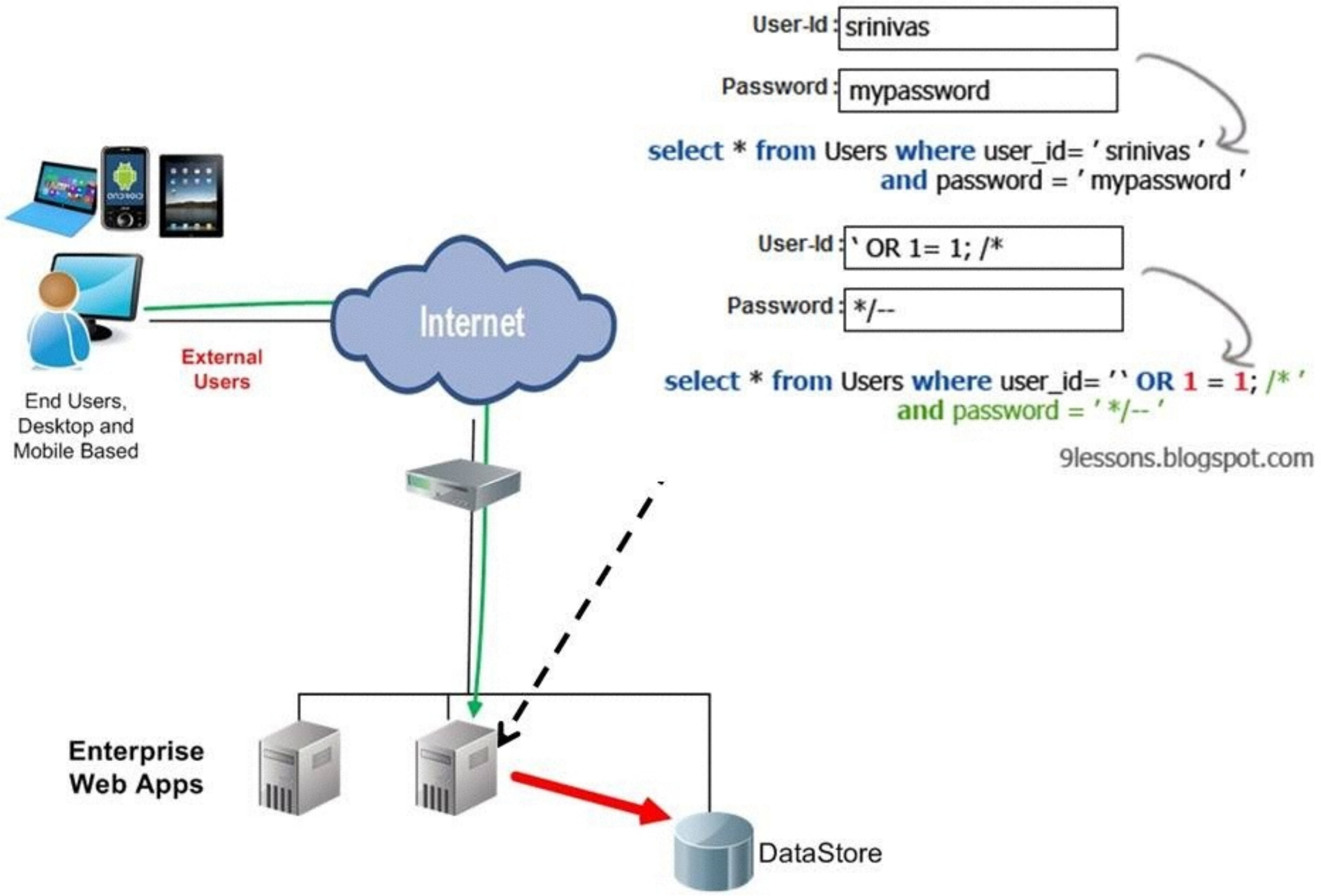
- The Open Web Application Security Project is a community dedicated to web application security
  - Builders (developers)
  - Defenders (security experts)
  - Breakers (hackers)

- OWASP publishes a lot of educational material and tools

- The Top 10 is a list of the most critical web application security flaws

A1 – Injection	Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2 – Broken Authentication and Session Management	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
A3 – Cross-Site Scripting (XSS)	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A4 – Insecure Direct Object References	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
A5 – Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
A6 – Sensitive Data Exposure	Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
A7 – Missing Function Level Access Control	Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
A8 - Cross-Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
A9 - Using Components with Known Vulnerabilities	Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
A10 – Unvalidated Redirects and Forwards	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

SQL Injection.



More information

- Books
  - Web Application Hacker's Handbook (D. Stuttard and M. Pinto)
  - Browser Hacker's Handbook (W. Alcorn et al.)
  - The Tangled Web (Michal Zalewski)

- Web:
  - owasp.org
  - exploit-db.com
  - cve.mitre.org

- Tools:
  - WebGoat (https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project)
  - NodeGoat (http://nodegoat.herokuapp.com/tutorial)
  - Burp Proxy (http://portswigger.net/burp/)
  - w3af (http://w3af.org/)
  - sqlmap (http://sqlmap.org/)

Demo