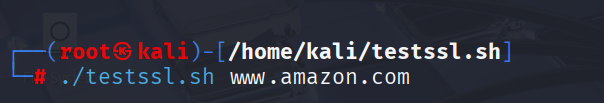
Testssl.sh is a free, command-line tool designed to check a server's service on any port for its support of TLS/SSL ciphers, protocols, and to identify potential cryptographic flaws.

Here we are performing some test cases on a website using testssl.sh cmd tool, moreover we are trying to break it down in a simpler form to our understand using the test cases below.

* Check SSL Version, Algorithms, Key length
* Check for Digital Certificate Validity (Duration, Signature and CN)
* Check credentials only delivered over HTTPS
* Check that the login form is delivered over HTTPS
* Check session tokens only delivered over HTTPS
* Check if HTTP Strict Transport Security (HSTS)

**TARGET WEBSITE :** Amazon.com

Here the prefix ./ before an executable filename like testssl.sh is used to specify that the shell should look for the executable in the current directory. We need to add the target address to the command to run the tool on an website to gather information. As shown in the snapshot below



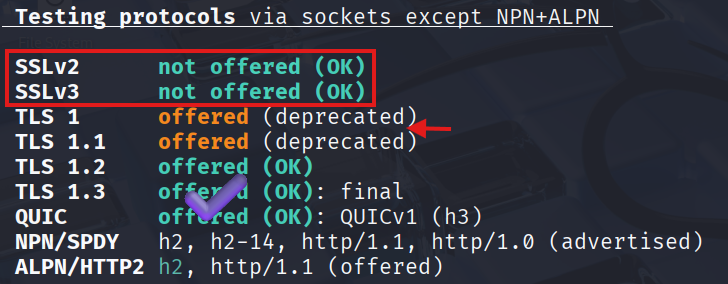
**W**e are checking for SSL Version, Algorithms, Key length.

**SSL Version :** SSLv2 and SSLv3 are not offered, which means the server doesn’t support any SSL versions

The server is configured to support **TLS versions only**, specifically:

**TLS 1.0** and **TLS 1.1** (deprecated — should be disabled)

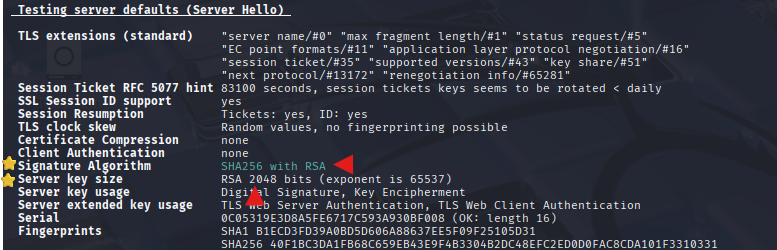
**TLS 1.2** and **TLS 1.3** (secure and recommended)



**Algorithms & Key length** : The algorithm used in the TLS certificate is **SHA256,** The public key length shown in the certificate is **2048 bits**, which refers to the RSA key size.

* **SHA256**: This is the hash function used to create a fixed-size digest of the certificate data. It ensures integrity—if any part of the certificate is altered, the hash won't match.
* **RSA Encryption**: Refers to the public key algorithm used for the digital signature and key exchange. It’s asymmetric, meaning it uses separate keys for encryption and decryption.

Together, **SHA256withRSA** is a secure and widely trusted signature algorithm that combines strong hashing with proven encryption. It's been a standard for high-security environments as shown in snapshot below.



**W**e are Checking for Digital Certificate Validity (Duration, Signature and CN) :

**Duration (Validity Period) : Start Date:** July 22, 2025, at 00:00 UTC

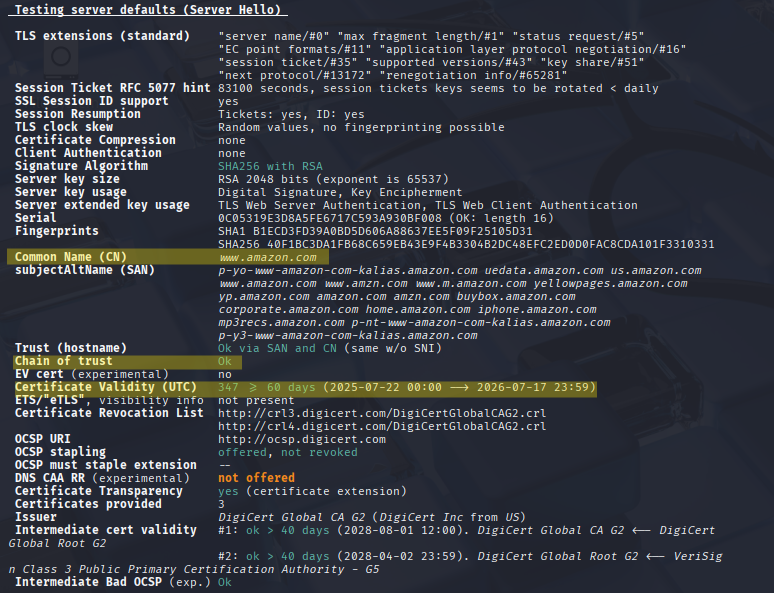
**End Date:** July 17, 2026, at 23:59 UTC

**Validity Duration:** **360 days** — a standard span for many public certificates to maintain security hygiene.

**Signature Algorithm :** **SHA256 with RSA Encryption -** This combination ensures both integrity (via SHA256 hashing) and strong asymmetric encryption (via RSA). It’s a well-established and secure choice for TLS certs.

**Common Name (CN) :**  [www.amazon.com](http://www.amazon.com) This is the primary domain the certificate is issued for. Useful when validating host identity during TLS handshakes.

As shown below in snapshot.



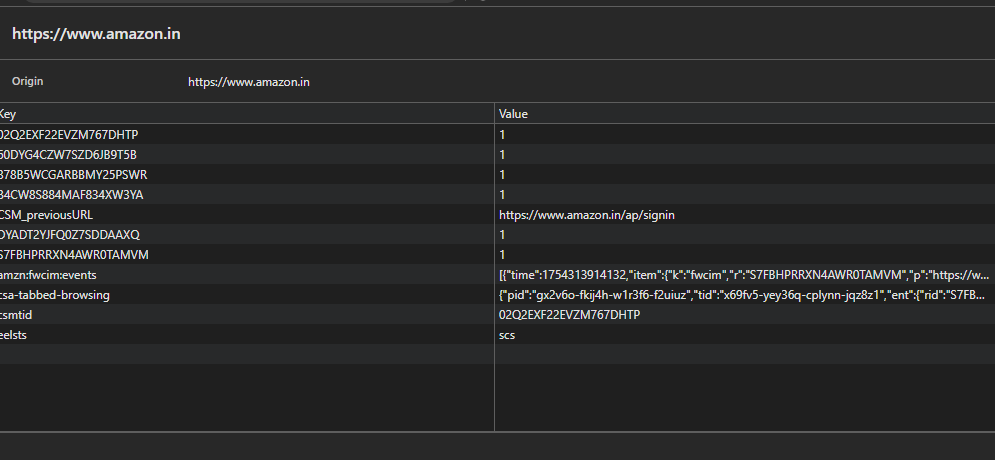
**W**e are checking credentials only delivered over HTTPS :

Using burpsuite we intercepted the traffic to confirm credential set on HTTP or HTTPS here we can see the username in plain text format which in HTTP but the password is encrypted HTTPS form, The HSTS (HTTP Strict Transport Security) for the website is 549 days=47474747 s, includeSubDomains, preload.



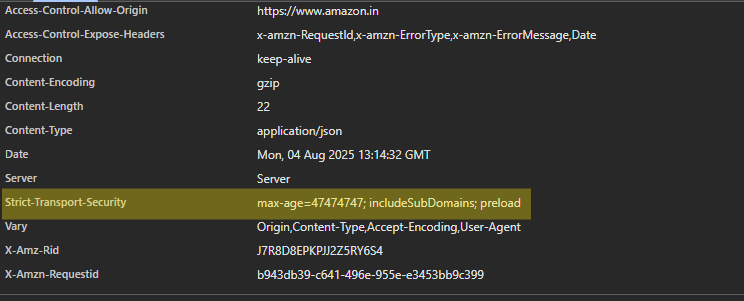
**W**e are Checking for session tokens only delivered over HTTPS :

A session token is a unique, often randomly generated, string of characters used to maintain the state of a user's interaction with a web application or service over a period of time. It acts as a temporary credential, allowing the server to recognize and authenticate a user's subsequent requests without requiring repeated login credentials for each action. As we can see in the below snapshot all the sessions or delivered on https.



**W**e Checking for HTTP Strict Transport Security (HSTS) :

HTTP Strict Transport Security (HSTS), a security mechanism that enforces HTTPS connections for a website, preventing downgrade attacks and other security vulnerabilities. It's not a separate protocol, but rather a header that instructs browsers to only use HTTPS when communicating with a specific website.



Vulenrablities found :

Here we SSL detected some vulnerabilities, some are potential some are potential but not ok state need run manual tests on the vulnerabilities to check weather they are true or false.

Screenshot 2025-08-04 164548Screenshot 2025-08-04 164629Screenshot 2025-08-04 164655