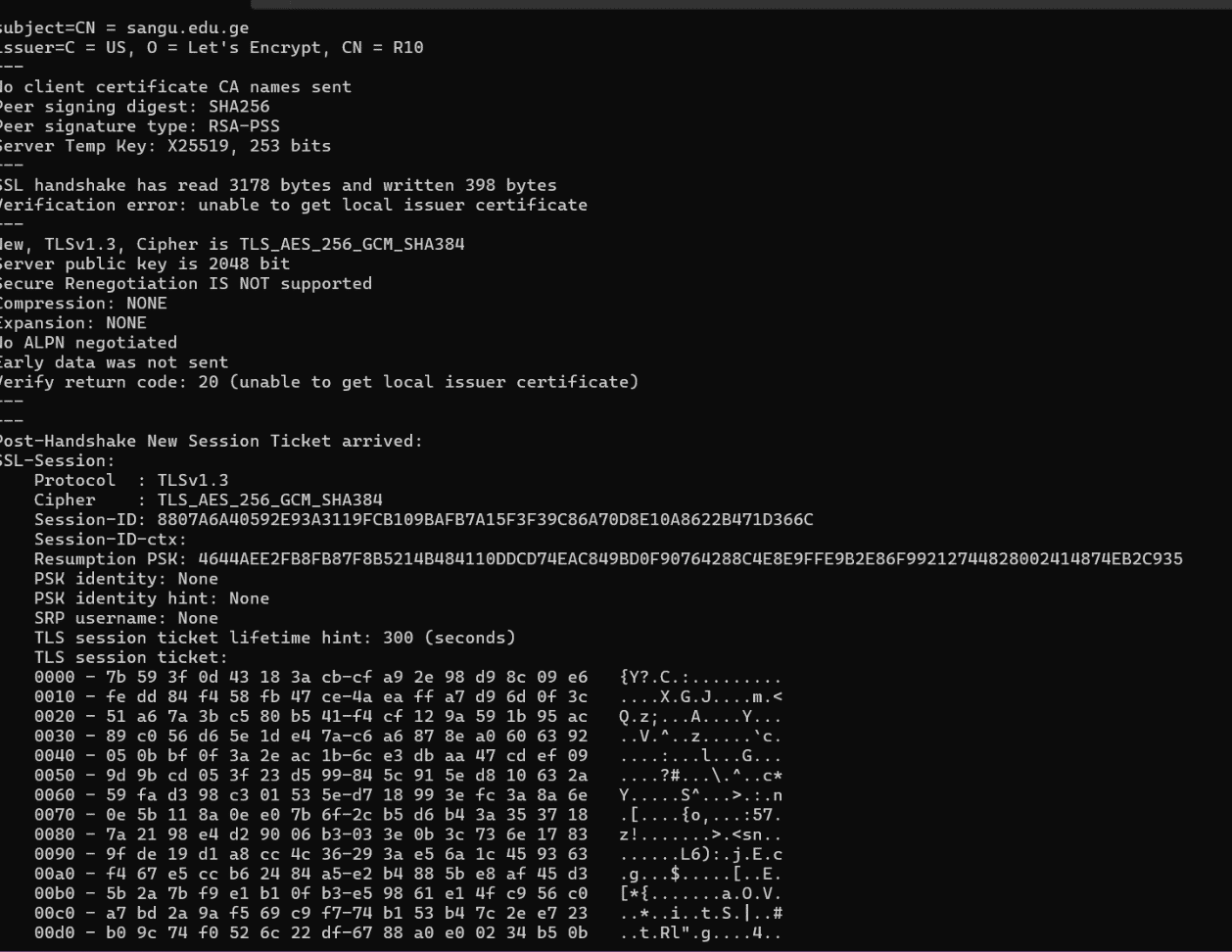
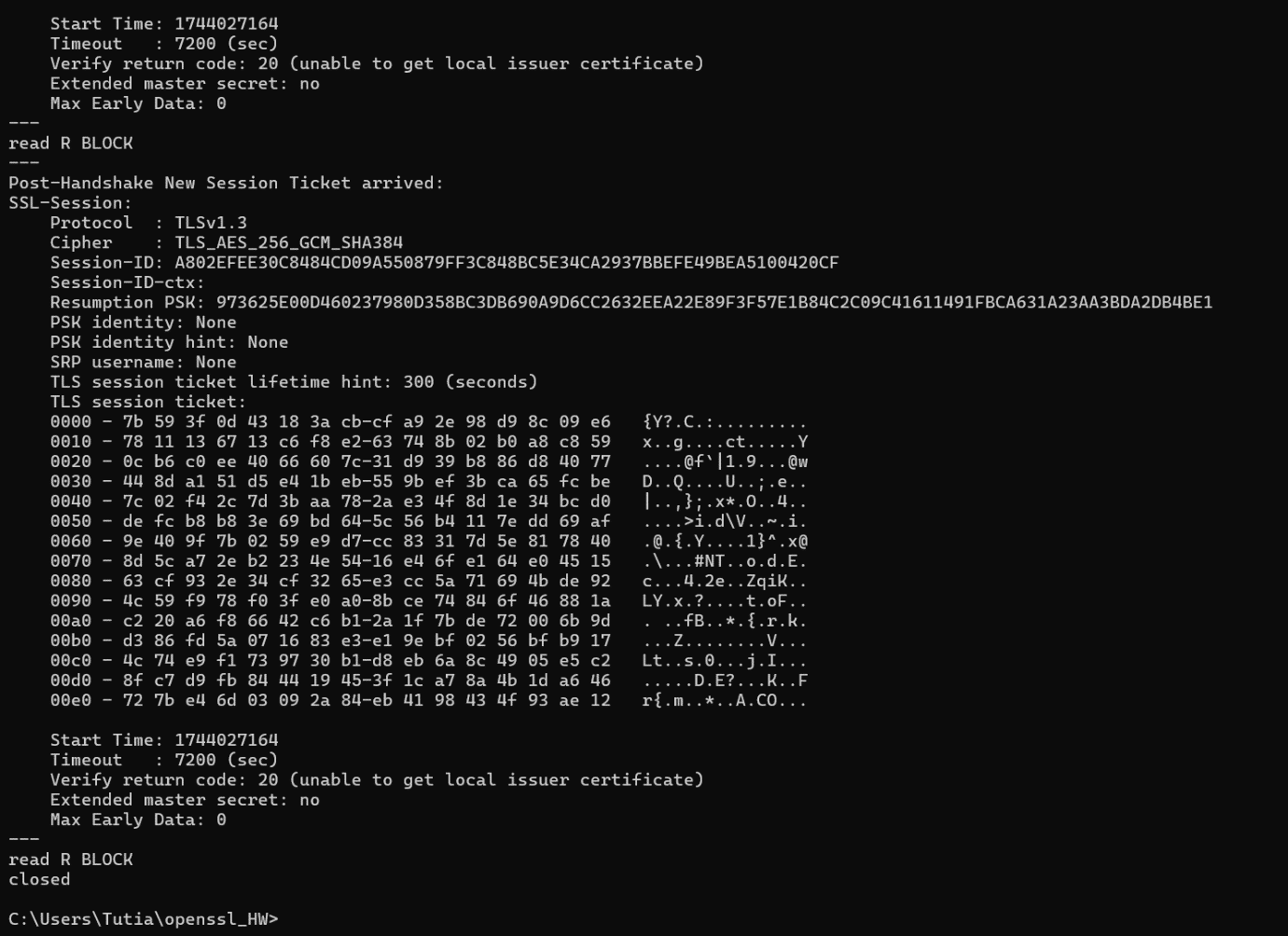
Inspection any HTTPS website with openssl s\_client

1. 



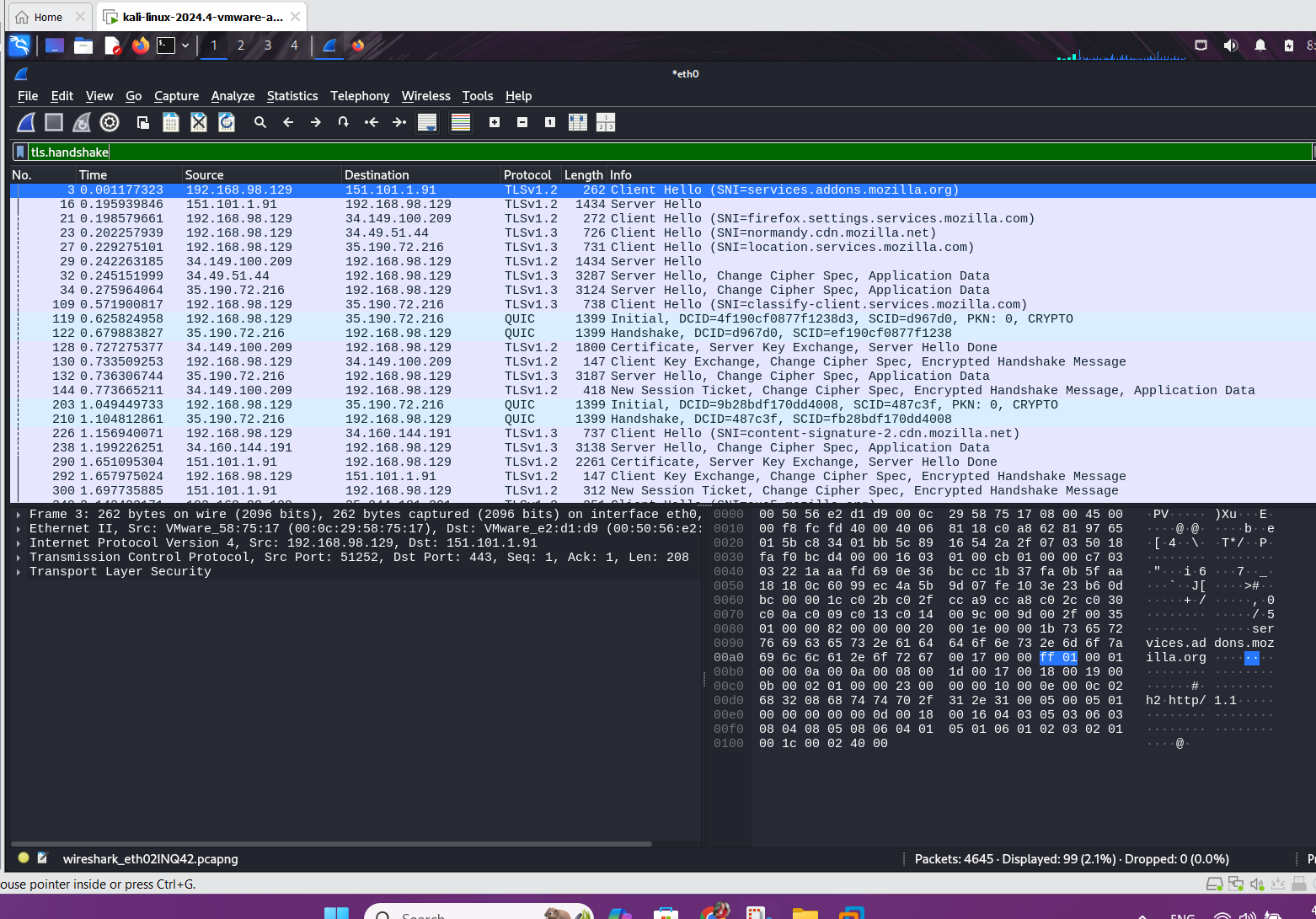


TLS version: TLS 1.3

Cipher suite: TLS\_AES\_256\_GCM\_SHA384

the server sent the certificate for sangu.edu.ge, issued by Let’s Encrypt (R10). The certificate is signed with RSA-SHA256, and the server uses a 2048-bit RSA key for encryption. A temporary key using X25519 (Elliptic Curve Diffie-Hellman) was also used for key exchange, ensuring forward secrecy.

1. Capture TLS Handshake with Wireshark



I visited google.com in a browser while running Wireshark on my Kali Linux virtual machine. In the captured traffic, we can observe the TLS handshake between the client and the server (over port 443. The handshake begins with a Client Hello message sent by the client using the TLSv1.2 protocol. This packet includes supported cipher suites, the TLS version, and the Server Name Indication, which in this case points to services.addons.mozilla.org. The server responds with a Server Hello, selecting a cipher suite and sending its digital certificate along with key exchange parameters. This is followed by the Server Key Exchange and Server Hello Done messages. The client then replies with a Client Key Exchange, Change Cipher Spec, and an Encrypted Handshake Message, completing the secure session setup. This series of packets demonstrates how the TLS handshake securely establishes encrypted communication between client and server. After the handshake, all further communication is encrypted, ensuring the confidentiality and integrity of the data exchanged.

**TLS Handshake Explanation**

The TLS handshake is a critical process that establishes a secure connection between a client and a server over the internet. It begins when the client sends a "Client Hello" message containing the supported TLS versions, cipher suites, and a randomly generated number. The server responds with a "Server Hello," selecting the TLS version and cipher suite to use, and sends its digital certificate to authenticate itself. This certificate includes the server's public key and is signed by a trusted Certificate Authority (CA). Key exchange then takes place using algorithms like ECDHE, which ensures forward secrecy. Once the keys are exchanged and verified, both parties generate session keys, exchange a final handshake message, and transition to encrypted communication.

**MITM Protection Mechanisms**

TLS is designed with robust mechanisms to protect against Man-in-the-Middle (MITM) attacks. The use of digital certificates issued by trusted Certificate Authorities ensures the authenticity of the server and helps verify that the client is communicating with the intended destination. During the handshake, public key cryptography ensures that only the server can decrypt data encrypted with its public key. Additionally, modern TLS implementations often use ephemeral key exchange algorithms, such as ECDHE, which provide forward secrecy—ensuring that even if encryption keys are compromised in the future, past sessions cannot be decrypted. These features together make it extremely difficult for attackers to intercept or alter communications undetected.

**TLS Application in Websites**

TLS is the foundation of secure communication on the web and is applied through HTTPS (Hypertext Transfer Protocol Secure). When users access websites over HTTPS, TLS encrypts the data exchanged between the browser and the web server, preventing eavesdropping and tampering by third parties. It secures sensitive information such as login credentials, personal data, and payment details, ensuring confidentiality and integrity. TLS also provides authentication through digital certificates, allowing users to verify that they are connecting to a legitimate website. In modern web browsers, a padlock icon in the address bar indicates that the site is using HTTPS and that the connection is secured with TLS.