UANL

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FCFM

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

Reporte Técnico

Sofía Martínez Cisneros

Juan Angel Rodriguez Bulnes

Angel David Morales Palomo

Introducción

El proyecto hecho y presentado por nosotros consiste en la implementación, ejecución y análisis de un sistema tipo cliente-servidor que fue desarrollado con C++, el propósito de este proyecto fue simular un payload diseñado para ver cómo era una autenticación segura comunicada con HTTP, manejando token y análisis de comportamiento de binarios en ambientes controlados como lo son las Máquinas Virtuales. El sistema en si este compuesto por un servidor que fue ejecutado en esta práctica que hicimos en una maquina virtual con Kali Linux y un cliente simulado entrando desde una maquina virtual con Windows 10, estas 2 maquinas virtuales interactuaron mediante solicitudes de HTTP comunes, sin cifrado, pero validando tokens JWT que fueron generados al ingresar usuarios y contraseñas correctas.

La finalidad de este proyecto no fue producir algún software malicioso, mas bien emular de maneta controlada como un payload funcional puede operar para facilitar el análisis estático y dinámico en un ambiente controlado.

Payload

Como tal, el proyecto genera un payload cuyo funcionamiento esta enfocado a una autenticación entre cliente y servidor. El servidor se encarga de recibir las solicitudes HTTP por el puesto 8080 especificando previo junto a la IP del servidor desde el cliente junto con las credenciales, una ves hecho esto las valida con un listado ya escrito localmente y en caso de que tanto el usuario como la contraseña no sean correctas una o la otra, genera o no un token JWT, que posterior a eso puede ser validado o reutilizado por un cliente para poder acceder a diversas rutas que son protegidas. El cliente en la maquina con Windows 10 es un programa algo sencillo y simple, la única función que tiene es enviar las credenciales y procesar la respuesta que da el servidor de vuelta.

Como se especifico en actividades anteriores y en el proyecto. El payload se hizo en un entorno de red aislado, las credenciales utilizadas son ficticias y ningún componente de alguno de los sistemas fue diseñado para que pudiera interactuar de alguna manera con equipos host o externos. El payload no oculta su presencia en si, tampoco tiene persistencia, no realiza acciones evasivas, no se aprovecha de ninguna vulnerabilidad, no modifica nada del sistema operativo y mucho menos afecta la integridad de alguno de los datos del host. El cumplimiento ético de este proyecto fue prioridad para mi equipo y para mi durante su desarrollo, nos aseguramos que permanezca en un marco seguro, transparente y controlado por completo.

Implementación y Diseño

El sistema en si se basa en la comunicación directa con un cliente en C++ y un servidor que, en nuestro caso, fue implementado con Linux.

El servidor usa “cpp-httplib” como un motor y “jwt-cpp” como su biblioteca principal a la hora de crear y validar los tokens JWT generados. Los datos estructurados los maneja la librería “json.hpp” y hace que permita el intercambio de información en el formato “JSON” par que sea legible entre los 2 dispositivos (cliente, servidor).

El cliente que es ejecutado en Windows 10, establece la conexión HTTP con el servidor utilizando las librerías y enviando una solicitud tipo POST hacia el “/login” que funciona como un endpoint. El server recibe los datos, los compara y si coinciden responde con un token JWT que el cliente imprime en su terminal.

El diseño de este proyecto fue creado para evitar un mal manejo manual de memoria o el uso de buffers sin límites. El flujo de trabajo se mantuvo sencillo para que el usuario que vea el código fuente pueda ver el panorama más completo como los elementos más importantes que son la autenticación, generación de tokens, comunicación HTTP y almacenamiento temporal

Pruebas

Mi equipo y yo hicimos unas pruebas de practica dentro del entorno controlado de red que mencione anteriormente. El server fue ejecutado en una maquina Linux como también mencione anteriormente, esta fue configurada con la dirección IP: 192.168.10.1 y el cliente que se probo en la maquina con Windows 10 fue configurada con la dirección IP: 192.168.10.2. Una vez que se establecieron estas ip en cada máquina, hicimos pruebas con pings para ver si se podían ver en la red y confirmamos que sí, la comunicación se estableció mediante una red tipo Host-Only en VirtualBox para que no hubiera problema a la hora de poder verse y estuvieran en el mismo segmento de red, aparte de que esto los ponían en aislamiento.

Al compilar los códigos .exe hicimos varias pruebas y error, por el tema de las bibliotecas y lo que ocupaba tener en el mismo directorio. Para esto hicimos uso de g++ junto con varias librerías como pthread, ssl y crypto, esto sirvió para el server, en cambio para el cliente tuvimos que usar MinGW por sus bibliotecas y por que aparte nos daba accesibilidad de compilar el código, los enlazamos con la biblioteca de sockets que tiene Windows “ws2_32”.

Después de esto hicimos varias pruebas con las autenticaciones exitosas y fallidas, inspeccionando el trafico mediante wireshark y validando los tokens recibidos.

Análisis estático del código

En el análisis estático examinamos cada módulo de nuestro proyecto, se revisaron los archivos “main_servidor.cpp”, “ServidorJWT.cpp”, “Usuarios.cpp”, así también como “ClienteJWT.cpp” de parte del cliente. El uso de “std::string” en lugar de buffers manuales reducio significativamente los riesgos que podía haber de errores de memoria y también el uso de JWT.

Al revisar estos elementos pudimos identificar algunas posibles mejoras, como por ejemplo las contraseñas que son almacenadas en texto plano presentan un riesgo obvio si quisiéramos trasladar este proyecto a un entorno de producción, el uso de HTTP sin cifrado permite ver el contenido que tienen todas las solicitudes sin ninguna dificultad. Por lo tanto, el análisis confirmo que no existen partes del código que fueron diseñadas para ocultar alguna acción de manipulación, procesos u operaciones que puedan afectar el sistema.

Análisis dinámico

Como tal en análisis dinámico vimos el comportamiento del payload durante toda la ejecución. El servidor, al iniciarse, al abrir el puerto 8080 y al permanecer al a espera de solicitudes. No genera procesos adicionales que modifiquen el entorno y no intenta obtener permisos de ningún tipo y el cliente se ejecuta como una aplicación de consola, que establece la conexión HTTP y finaliza después de recibir la respuesta.

Durante toda la captura del tráfico, se observo que toda la comunicación fluye de forma clara y muy predecible, el cliente envía sus credenciales y el servidor responde con el token firmado. El token aparece también dentro de el archivo adjunto “tokens.txt”, ahí podemos ver todos los tokens que genera facilitando así la inspección.

Riesgos y Mitigaciones

Aunque en si todo el proyecto es benigno y se ejecuta solo en un entorno controlado y aislado, es importante la consideración de los riesgos del diseño, como por ejemplo el uso de contraseñas en texto plano que representa un riesgo en cualquier escenario real, por lo que la mitigación de este problema seria hacer hash con un almacenamiento interno seguro. La falta de cifrado también se le podría considerar otra problemática, ya que en las comunicaciones entre servidor y cliente pueden interceptarse credenciales y tokens, lo que podría solucionarse cambiando de HTTP a HTTPS y alguno que otro certificado auto firmado.

Ahora, el almacenar token en algún disco duro es útil en cuestión de análisis, pero podría presentarse un riesgo en escenarios más estrictos donde quizás es mejor mantener tokens en solo memoria.

Conclusión

En conclusión, el proyecto termino demostrando que de forma exitosa se puede diseñar un payload seguro dentro de un entorno controlado como en esta práctica, la comunicación entre el cliente y el servidor se logro de manera estable la autenticación funciono como se esperaba y el análisis estático y dinámico nos ayudo a identificar tanto las fortalezas como posibles mejoras.

La ejecución de entre 2 sistemas operativos nos ayudo a diferenciar el manejo de bibliotecas y red entre ambas.

Como trabajo futuro seria tal vez recomendable el ampliar la funcionalidad del servidor cambiando a HTTP, almacenando contraseñas mediante hash, implementar alguno que otro rol y permiso de usuarios y mejorar los tokens, también otra consideración a futuro podría ser la integración de herramientas que ayuden con monitoreo, módulos de logging, detección de anomalías, etc.