MySSH: Arhitectură Securizată Client-Server pentru Execuție de Comenzi la Distanță

Grosu Mihnea-Constantin

Facultatea de Informatică, Iași
grosumihnea88@gmail.com

Abstract. Acest document prezintă proiectarea și implementarea MySSH, o aplicație de tip terminal la distanță (Remote Shell). Aplicația se distinge prin mecanisme avansate de securitate: criptarea traficului, auditarea forensică a comenzilor ("Black Box") și guvernanța resurselor pentru prevenirea atacurilor de tip DoS local. Arhitectura utilizează primitive POSIX de nivel jos pentru a gestiona execuția concurentă și manipularea fluxurilor de date.

Keywords: Remote Shell, TCP/IP, OpenSSL, Resource Governance, Forensic Auditing.

1 Introducere

Obiectivul principal al proiectului MySSH este dezvoltarea unei soluții robuste pentru administrarea sistemelor la distanță, oferind o alternativă educațională la protocolul consacrat SSH. Viziunea proiectului pune accent pe trei piloni: securitate (prin criptare și audit), stabilitate (prin limitarea consumului de resurse) și funcționalitate avansată (suport pentru operatori shell precum `|` sau `>`).

Spre deosebire de soluțiile rudimentare (ex. Telnet), MySSH implementează un protocol binar propriu care încapsulează comenzile și rezultatele, prevenind interceptarea datelor sensibile. De asemenea, serverul este proiectat să reziste la comenzi malițioase sau scrise greșit care ar putea destabiliza sistemul gazdă.

2 Tehnologii Aplicate

Pentru implementare au fost selectate următoarele tehnologii, având la bază cerințele de performanță și control asupra sistemului de operare:

- **Limbajul C++ (Standard C++17):** Ales pentru accesul direct la API-ul POSIX și gestionarea eficientă a memoriei.
- **Protocolul TCP/IP:** Utilizat pentru stratul de transport datorită garanției livrării pachetelor și ordonării acestora.
- **Concurență prin Procese (fork):** Serverul utilizează un model *multi-process*, asigurând izolarea sesiunilor.

- **Resource Governance (setrlimit):** O funcționalitate distinctivă a MySSH este utilizarea primitivei `setrlimit` pentru a impune limite stricte proceselor copil (ex: număr maxim de procese, memorie alocată), prevenind atacurile de tip "Fork Bomb".
- **Stocare Persistentă (Flat-file):** Gestiunea utilizatorilor și a jurnalelor de audit se realizează local (fișiere text/hash), evitând dependența de baze de date SQL complexe.
- **Primitive POSIX (pipe, dup2, execvp):** Nucleul execuției comenzilor folosește un parser propriu recursiv pentru înlănțuirea comenzilor (piping).
- **OpenSSL:** Utilizată pentru implementarea criptării simetrice (AES-256).

3 Structura Aplicației

Aplicația urmează o arhitectură clasică Client-Server, îmbogățită cu module de securitate și audit.

3.1 Componentele Serverului

1. **Network Listener:** Bucla principală care acceptă conexiuni TCP.
2. **Protocol Handler:** Deserializarea pachetelor binare conform structurii TLV.
3. **Auth Manager:** Verifică credențialele utilizatorului.
4. **Forensic Logger (Black Box):** Un modul care interceptează toate comenzile primite și rezultatele generate înainte de a le trimite clientului. Acestea sunt scrise într-un jurnal imuabil cu timestamp, asigurând trasabilitatea acțiunilor.
5. **Command Executor (Core Engine):** Componenta care interpretează string-ul comenzii, construiește arborele de execuție (`&&`, `||`, `|`) și aplică limitele de resurse înainte de execuție.

3.2 Componentele Clientului

Clientul este responsabil de preluarea input-ului de la utilizator, criptarea acestuia și afișarea rezultatelor primite asincron de la server.

4 Protocolul de Comunicare

Protocolul MySSH este unul binar, orientat pe mesaje, operând deasupra TCP. Fiecare mesaj respectă formatul TLV (Type-Length-Value).

4.1 Structura Header-ului

Header-ul are o dimensiune fixă de 5 bytes (padding anulat prin `#pragma pack(1)`):

- **Payload Length (4 bytes):** Lungimea datelor (Network Byte Order).
- **Message Type (1 byte):** Codul operațiunii.

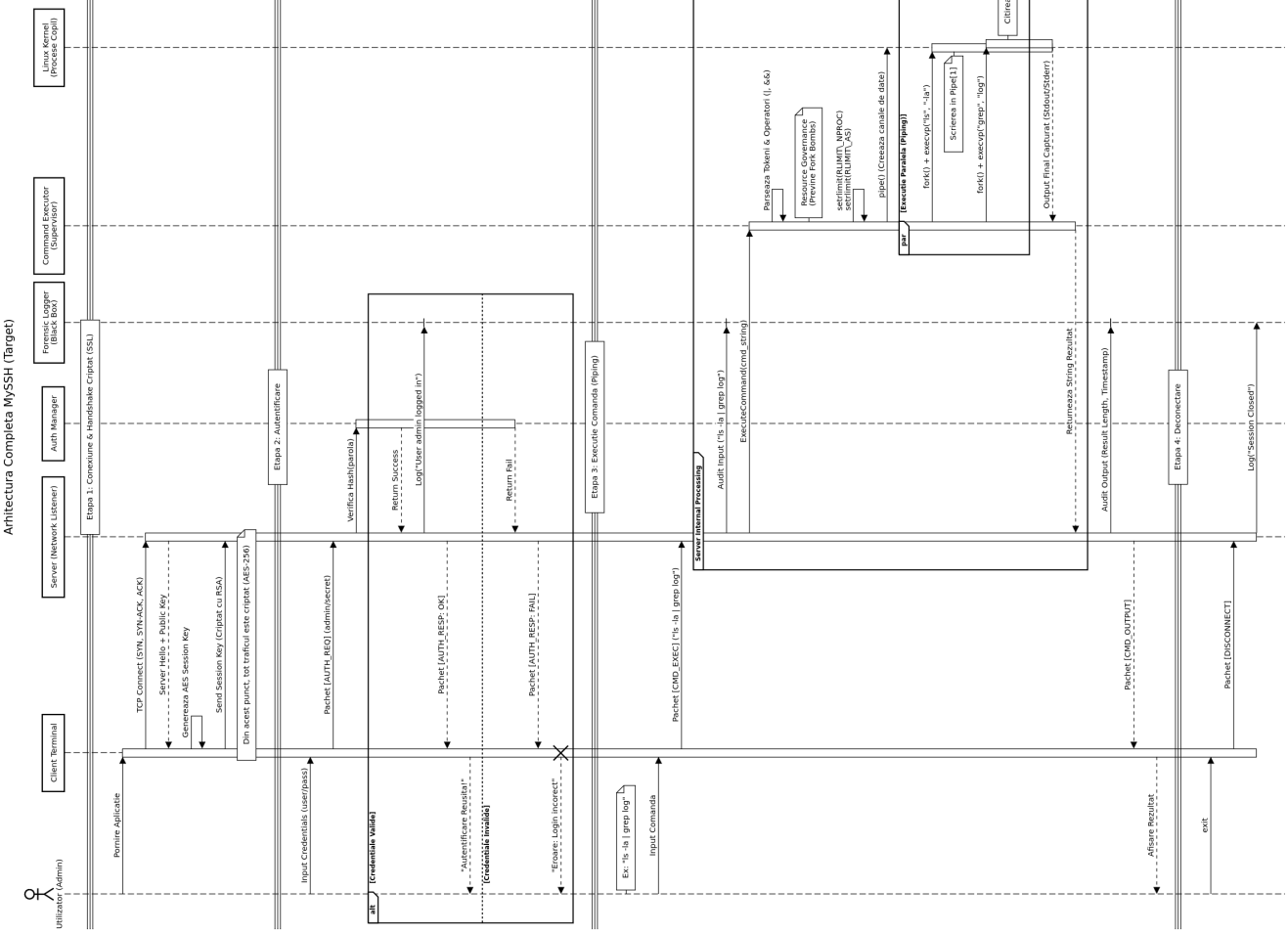

Fig. 1. Diagrama de Arhitectură MySSH (Target Architecture): Flux complet de la Conexiune la Execuție.

Nume Simbolic	Cod	Descriere
CMD_EXEC	0x01	Comandă shell trimisă de client.
CMD_OUTPUT	0x02	Rezultatul (stdout) trimis de server.
CMD_ERROR	0x03	Mesaj de eroare (stderr).
AUTH_REQ	0x04	Pereche user/parolă criptată.
AUTH_RESP	0x05	Răspuns autentificare (OK/FAIL).
FILE_UPLOAD	0x06	Inițiere transfer fișier.
HEARTBEAT	0x09	Verificare conexiune activă.

Table 1. Definirea tipurilor de mesaje în protocolul MySSH

4.2 Tipuri de Mesaje

5 Scenarii de Utilizare

5.1 Scenarii de Succes

1. **Autentificare și Audit:** Utilizatorul *admin* se autentifică. Serverul validează parola și deschide o sesiune de jurnalizare. Orice comandă ulterioară (ex: `whoami`) este scrisă automat în fișierul de audit al serverului înainte de a fi executată.
2. **Execuție Complexă (Piping):** Utilizatorul trimite: `cat logs.txt | grep "Error" | wc -l`. Serverul creează 3 procese interconectate. Rezultatul final este capturat și trimis clientului.
3. **Transfer de Fișier:** Comanda `upload script.sh` determină clientul să fragmenteze fișierul și să trimită pachete `FILE_DATA`. Serverul reconstruiește fișierul pe disc.

5.2 Scenarii de Eșec

1. **Tentativă de Fork Bomb (Resource Governance):** Utilizatorul (intenționat sau din greșeală) execută o comandă recursivă infinită de tip `:(){ :|:& };:`. Datorită limitelor impuse prin `setrlimit` (RLIMIT_NPROC), serverul blochează crearea de noi procese după un prag stabilit și trimite eroarea "Resource limit exceeded", protejând sistemul gazdă.
2. **Eșec Autentificare:** Utilizatorul introduce o parolă greșită. Serverul trimite `AUTH_RESP: FAIL` și închide conexiunea după 3 încercări.
3. **Comandă Inexistentă:** Utilizatorul trimite `comanda_falsa`. Executorul returnează eroare, iar serverul trimite `CMD_ERROR`.

6 Concluzii și Elemente de Originalitate

Proiectul MySSH a demonstrat fezabilitatea implementării unui shell securizat folosind primitive de sistem. Un element major de originalitate îl constituie implementarea modulelor de **Black Box Auditing** (pentru trasabilitate) și **Resource**

Governance (prevenirea epuizării resurselor prin `setrlimit`), funcționalități absente în implementările didactice standard.

Direcțiile viitoare includ implementarea unui terminal virtual complet (PTY) și compresia datelor.

References

1. Stevens, W. R., Fenner, B., & Rudoff, A. M. (2004). *Unix Network Programming, Volume 1: The Sockets Networking API*. Addison-Wesley Professional.
2. Kerrisk, M. (2010). *The Linux Programming Interface: A Linux and UNIX System Programming Handbook*. No Starch Press.
3. OpenSSL Software Foundation. (2023). *OpenSSL Documentation*. Retrieved from `https://www.openssl.org/docs/`
4. Ylonen, T., & Lonvick, C. (2006). *RFC 4251: The Secure Shell (SSH) Protocol Architecture*. IETF.