WebServer Security Assessment

Introduction

Author: Chinmay Lohani

GitHub: github.com/chinmayL27

Topic: Security assessment of opensource WebServer

URL to source: https://github.com/Theldus/wsServer

1. Identify assets

List of each asset in the webserver.

Data sent over to the WebSocket: the data sent by client like instructions, PII, confidential data (based on the application).
Connection configuration: the Client-WebSocket-Server connection needs to be stable and integral, so it is an asset.
Source Code and Binaries: the source code, binaries and executables that can be taken advantage of if not protected against malicious exploits.

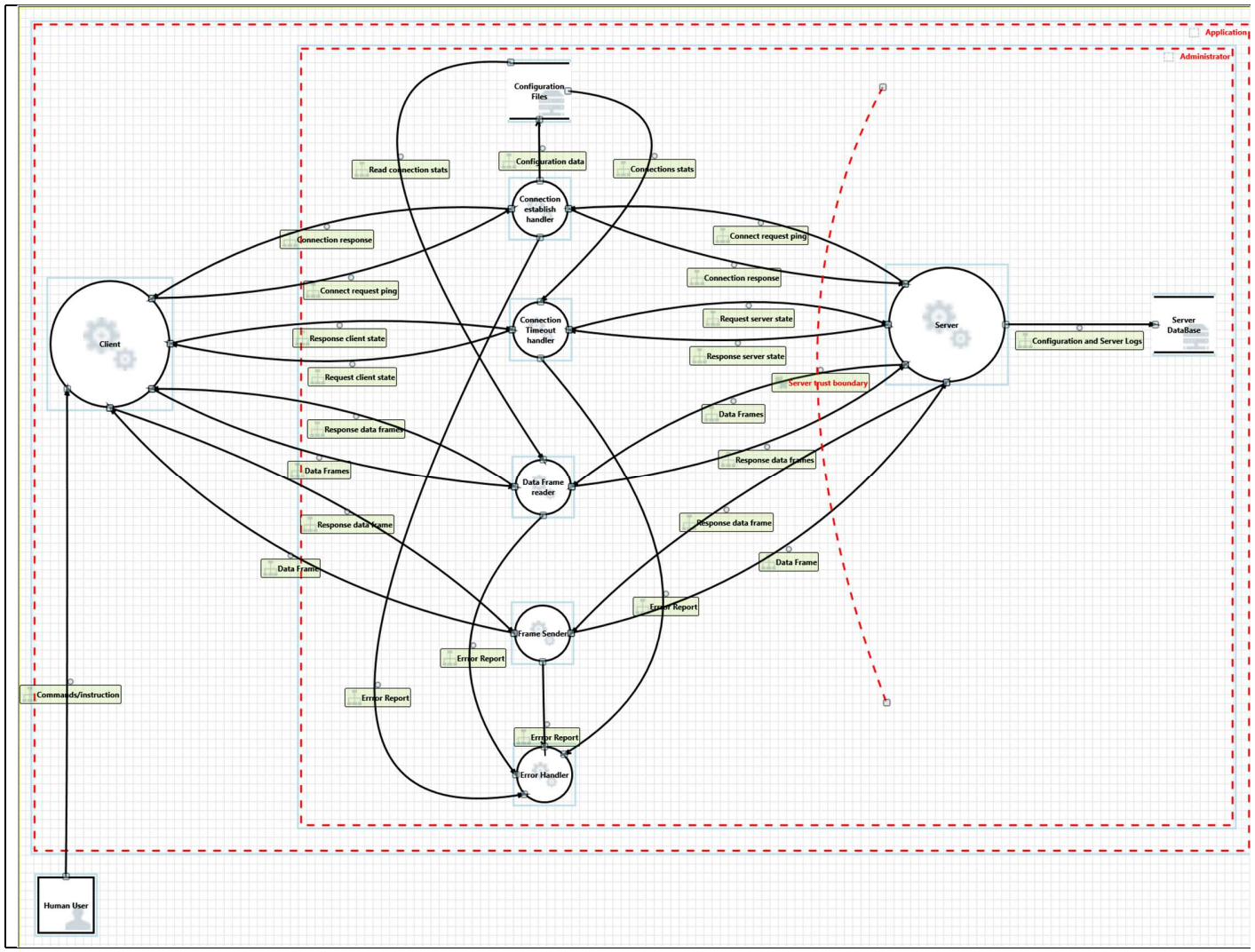
2. Architecture overview and decomposition of the application

This activity performs a functional decomposition of the system architecture and breaks the components down into functions. E.g., system functions such as "process user input" or "decrypt data store").

- Get Client address.
- Validate address.
- Get Server address.
- Establish WebSocket connection.
- Do handshake.
- Start timeout.
- Check keep-alive ping.
- Check client state.
- Check server state.
- Read Frame.
- Handle callbacks.
- Validate frame.
- Send Frame.
- Close socket.
- Handle unusual behavior/errors.

An architecture diagram (this is the high-level threat model) using the architecture components corresponding to the functions identified previously.

- Identifies trust boundaries
- Identifies data flow
- Identifies entry points


Application
Administrator
Configuration Files
Configuration data
Read connection stats
Connections stats
Connection establish handler
Connection response
Connect request ping
Connect request ping
Connection response
Client
Connection Timeout handler
Request server state
Server
Server DataBase
Response client state
Response server state
Configuration and Server Logs
Request client state
Server trust boundary
Data Frames
Response data frames
Data Frames
Data Frame reader
Response data frames
Response data frame
Response data frame
Data Frame
Data Frame
Error Report
Frame Sender
Error Report
Commands/instruction
Error Report
Error Report
Error Handler
Human User

The technologies used by the system.

- C language.
- POSIX
- TCP
- AUTOBAHN
- Doxygen

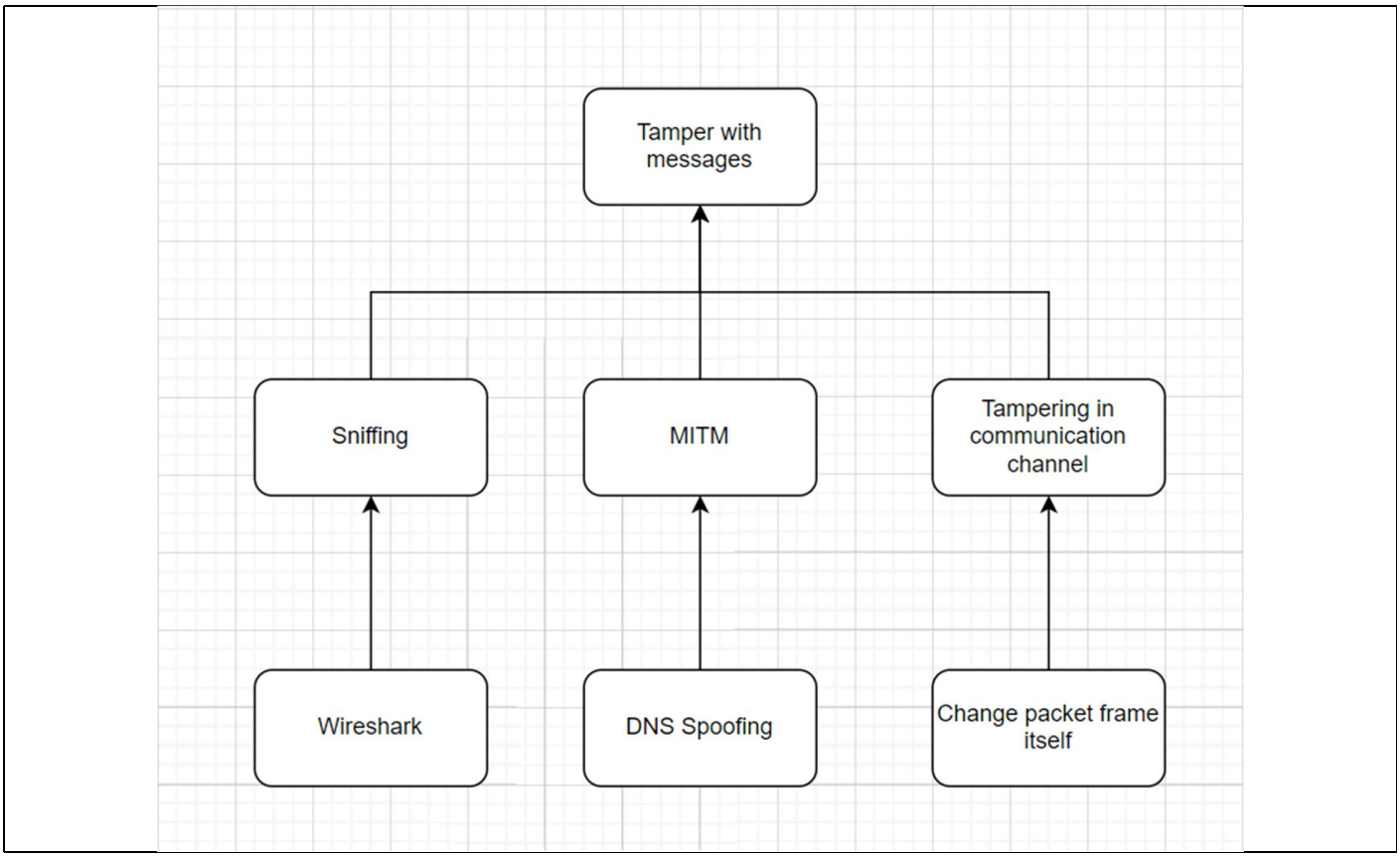
3. *Documentation of the security profile*

Security Profile.

Category	Considerations
Authentication	Is the client address authenticated before connection establishment? Is the server address authenticated before connection establishment?
Input Validation	Are the input frames validated before processing? If the format of data validated?
Access control	Is client access monitored for accessing WebSocket? Is client access monitored for accessing server?
Logging	Is the message communication logged anywhere? Are the errors and unusual behaviors logged somewhere?
Error Handling	Are the generic errors handled properly? Are the unusual/complicated errors handled and taken care of properly?
Config Management	Are the ports for client and server managed properly? Is there any update happening in configuration during runtime or by external factors?
Encryption	Are the messages or packet frames encrypted? Is there usage of certificates or any other mechanism to monitor the network?
Session Management	Are the session connections timed? (timeouts) Is the timeout duration set to appropriate value?
Resource Management	Are the race conditions handled properly? Is resource allocation properly handled?

4. *Identifying the threats*

Used the STRIDE approach, attack trees, categorized threat lists, and attack patterns to identify threats for the application.



5 relevant attack patterns.

Property	Threat Definition	Examples	Mitigations
Integrity (Buffer Overflow)	Adding more data than the buffer size to overwrite in memory.	Attacker sending oversized frames with injected code to gain unauthorized access	Bound checking and input validation at all interactions
Confidentiality (Session Hijack)	Guessing or predicting the session identification code or tokens to impersonate the process and gain access grant.	An attacker guessing an honest client process's session id and using it to impersonate as the client process and gain the client privileges.	Randomization of tokens, two-factor authorization.
Confidentiality (Man In The Middle)	Listening to packets in the communication channel.	An attacker using Wireshark or a similar tool to read the packet frames to gain the data.	Encryption of packets, or using certificates to verify.
Availability (DoS)	Flooding and overwhelming the system with multiple requests to hamper normal performance	An attacker sending multiple connection requests to the web socket to occupy the resource pool	Session Timeouts and rate limiting.
Integrity (Exploit Injection)	Inject malicious code through the packet frames to cause unusual system performance.	Attacker sending concatenated commands to cause the WebSocket to perform malicious actions.	Validating input, and process access checks.

5. Document and rate the threats

Documentation of the threats to the application.

Threat Description	**Attacker tampers with message packet that are being transmitted over communication channel.**
Threat target	Communication channel.
Risk	High
Attack techniques	M-I-T-M
Countermeasures	Message frame authentication using certificates at both client and server or use SSL.
Threat Description	**Attacker occupies the entire resource pool of the WebSocket by having multiple connections simultaneously.**
Threat target	WebSocket connection handler process.
Risk	Medium
Attack techniques	Slow-loris, rapid-connection establishment.
Countermeasures	Limit the client connections, CAPTCHA, authentication for unusual traffic.
Threat Description	**Attacker performs Buffer exploit**
Threat target	WebSocket message read handler.
Risk	Medium
Attack techniques	Improper sized buffer, shellcode injection
Countermeasures	Bound checking, non-executable stack.
Threat Description	**Attacker sends exploit to server from client**
Threat target	Server
Risk	High
Attack techniques	Sending malware packed with instructions from client.
Countermeasures	Firewall implementation.
Threat Description	**Attacker impersonates as client (spoofing attack)**
Threat target	Connection handler
Risk	Medium
Attack techniques	Guessing the identifiers of client.
Countermeasures	Randomize identifier, add nonce values to make guessing hard