https://doi.org/10.31436/ijpcc.v12i1.649

SSL/TLS Certificate Validation Tool for Pre-Authentication Captive Portals

Certificate Validation in Captive Portals

Shafana M.S[1*], Adamu Abubakar Ibrahim[2]
[1]Department of ICT, Faculty of Technology, South Eastern University of Sri Lanka, Sri Lanka
[1,2]Department of Computer Science, International Islamic University Malaysia, Kuala Lumpur, Malaysia

*Corresponding author: zainashareef@gmail.com
(Received: 4th November 2025; Accepted: 12th December, 2025; Published on-line: 30th January, 2026)

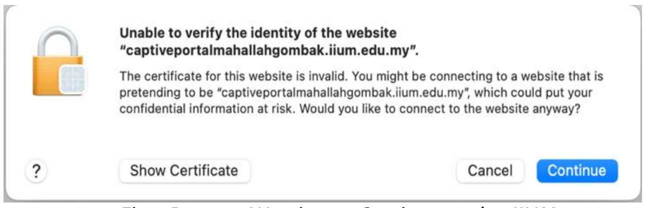
Abstract— Public network captive portal often disrupts the handshake in the SSL/TLS protocol and displays browser warnings that are sometimes ambiguous and sometimes excessive. These warnings can be misleading for users even for those with technical expertise. Notwithstanding the associated risks, there are limited tools that can be used to validate the trust of the SSL/TLS certificates in pre-authentication environments. This paper presents a lightweight and automated tool that is designed to validate the contents of an extracted certificate chain of SSL/TLS from live or stored handshake traffic at captive portals. The tool uses the trust evaluation engine of OpenSSL, supplemented with a Mozilla-compatible CA bundle to determine the validity of certificates and enables automatic retrieval of the missing intermediate certificates through AIA URLs to improve the accuracy of validation. The tool was evaluated using TLS handshakes captured from the IIUM Wi-Fi captive portal and samples from the ISCXVPN2016 dataset. After intermediate correction, the tool achieved 100% detection accuracy with no false positives and false negatives. It was able to detect misconfigured, expired or incomplete chains and validate known secure sessions. The proposed solution had more accurate and actionable diagnostics compared to browser-based indicators and tools like SSL Labs in pre-login situations, where existing methods often fall short. This tool fills an important gap in network security for users, by enhancing transparency and trust in certificate assessment. Its automation and diagnostic clarity make it an effective tool for both researchers and general users and provide reliable SSL/TLS validation in environments where conventional trust signals are unavailable or misleading.

Keywords— Captive Portal, Certificate Chain Verification, Intermediate Certificate Recovery, OpenSSL, SSL/TLS validation, Wi-Fi Security.

I. INTRODUCTION

Captive portals are a commonplace feature of both the public and institutional Wi-Fi providers, essentially acting as access control points that determine who can access the broader Internet. An example of this is the International Islamic University Malaysia (IIUM), where users are frequently presented with a login portal such as captiveportalgombak1.iium.edu.my which is used by all students, staff and guests. Although such portals serve their intended role, they also present substantial cybersecurity risks. One of the main issues is associated with the use of SSL/TLS certificates [5]. These online certificates authenticate a website and encrypt information during transmission. A captive portal should have a valid certificate issued by a recognised Certificate Authority (CA) in order to be secure. The inability to satisfy this need may allow attackers to impersonate legitimate portals, execute man-in-the-middle attacks, or steal personal data unnoticed.

Academic studies and practical observations have shown that a great number of portals fail to implement SSL/TLS correctly. Misconfigured certificate chains and poorly designed warning pages are common. This means that users receive browser warnings that they fail to understand and usually ignore. The so-called warning fatigue is also a significant factor, after being shown the same generic warnings many times ("Your connection is not private") users will disregard them completely.


Unable to verify the identity of the website "captiveportalmahallahgombak.iium.edu.my".

The certificate for this website is invalid. You might be connecting to a website that is pretending to be "captiveportalmahallahgombak.iium.edu.my", which could put your confidential information at risk. Would you like to connect to the website anyway?

Fig 1. Browser Warning on Captive portal at IIUM

For non-experts, the concepts of certificate hierarchies and expiration dates may be confusing and intimidating. Research shows that such confusion significantly increases the chances of users being victims of phishing or session hijacking as well as data leakage, even in usage of public WiFi also privacy risks arise [2].

The urgency of this issue was pointed out by a real incident at IIUM. Users who tried to use the Wi-Fi network of the university received alarming browser alerts (see Fig 1) stating that the identity of the portal could not be verified. But after further examination with the help of OpenSSL, it was established that the certificate of the portal was genuine, properly issued to the domain iium.edu.my by GlobalSign, and valid in terms of expiration. The issue was traced to a missing intermediate certificate, which made the chain of trust incomplete. This was a technically small gap, but it caused browsers to reject the certificate, and users had to determine whether to continue or not, usually without understanding the underlying risk.

This situation reveals two fundamental issues: a technical misconfiguration that weakens the portal's trust model, and behavioural weakness where the users are not equipped to make informed security decisions. The cross-section between those failures explains how network security can be compromised by a minor mistake, like the lack of an intermediate certificate that undermines trust. Also, browser warnings do not contextualise well in captive portal settings; they fail to inform users about the specific nature of the problem or whether it might be safely ignored. Users are therefore forced to make a trade-off between usability and perceived security without the necessary insight to do so effectively.

The rationale behind the study is that there is a high level of dependency on wireless access systems in institutions of learning, and that more intelligent, context-aware systems that can facilitate secure connectivity are required. Misconfigurations in the implementation of the SSL/TLS systems are preventable but still common and represent a continuing risk in the environment where users have to perform authentication over the untrusted or partially trusted connections. In this regard, this paper aims to create and deploy a lightweight, automated inspection system that can use Wireshark packet captures to identify and validate the SSL/TLS certificate chains and, finally, to advise the user whether a certificate warning is genuinely critical or safely ignorable.

This study will help to address infrastructure-level vulnerabilities and end-user vulnerabilities, and introduce a transparent validation process that will help enhance certificate-based security in captive settings and lead to more trustworthy public internet access experiences

II. LITERATURE REVIEW

A. *Captive Portals and SSL/TLS Certificate Challenges:*

Captive portals are commonly used in open and institutional Wi-Fi networks to control user access before authentication. These mechanisms tend to disrupt HTTPS connections, which may result in security misconfigurations despite their usefulness. Recent studies have shown that captive portal mini-browsers of popular operating systems often disable the validation of the SSL/TLS certificate and, as a result, expose users to man-in-the-middle (MITM) attacks and impersonation risks [4]. Authors in [7] also notes that most captive portals use HTTP redirection during the user authentication process, which may mislead users and give attackers the chance to deliver malicious content by using a cloned SSID.

One of the recurrent problems is the incorrect setup of TLS certificates. Improper installation of intermediary certificates often causes trust chain failures, which is treated as an untrusted connection by browsers regardless of the validity of root and leaf certificates. These misconfigurations provide a situation where seemingly secure portals are insecure, which undermines user trust towards the portal and the security of the entire network.

B. *User Behaviour in Response to SSL/TLS Warnings:*

User interaction with browser-based warnings on the use of the SSL/TLS is an enduring issue in web security. The study [9] reported that users have been found to ignore such warnings either out of habit or in an attempt to act quickly without regard to the security concerns. Authors in [1] observed that although the design of warnings had been improved marginally, most users still made unsafe choices. Authors of [6] applied these results to real-world scenarios and pointed out that users are more likely to over trust networks like campus Wi-Fi more than is warranted and ignore severe certificate warnings. These patterns in behaviour highlight the insufficiency of browser warnings as a defence mechanism, especially when they appear in captive portals where they are nonspecific and the cognitive load is entirely placed on the user. It follows, therefore, that there is an urgent need to have tools that provide actionable and context-based feedback that can be used to guide user decisions effectively.

C. *SSL/TLS Certificate Analysis Tools:*

Wireshark is the standard of deep packet inspection in the industry, providing all the analysis of the SSL/TLS including the handshake parsing, certificate decoding, and aspects of TLS 1.3 compatibility [11]. However, its technical depth is often impractical for non-technical users. The approach [10] is a simple and server-based certificate scanner that lacks support for passive or client-side scanning in a Wi-Fi setting.

There are no existing tools specifically tailored for captive portal environments with packet-level analysis and automated certificate validation and user advisories. This gap highlights the need to have a user-friendly, lightweight solution that operates in both pre- and post-authentication stages.

D. Attack Models and Real-World Attacks in Captive Portals

Real-world attack scenarios highlight how vulnerable captive portals can be. The study [7] divided the common threats, among which are Evil Twin attacks, where the adversaries pretend to be legitimate networks and offer a fake portal to steal credentials [3]. They tend to work well because of improperly configured certificates and users' false sense of security on familiar networks. The other attack vectors include MITM interception during HTTP redirection and session hijacking via stolen cookies [8]. Such threats are often not identified due to the fact that users post sensitive data to portals that are marked as untrusted, but without understanding the implications. Critical insights in the key studies are summarised in Table 1 below.

TABLE 1: KEY STUDIES ON CAPTIVE PORTALS AND SSL/TLS WARNINGS

Study	*Focus*	*Key Findings*	*Limitations*
[4]	Mini-browser TLS validation	Captive portals often bypass SSL validation	Lacks user-side countermeasures
[9]	User behavior to warnings	Users frequently ignore SSL alerts	Based on outdated UI models
[1]	Warning effectiveness	Design helps but does not eliminate risky behavior	No contextual guidance
[6]	Behavior on trusted networks	Users ignore warnings in familiar networks	Overlooks captive portal uniqueness
[11]	TLS handshake analysis	Robust technical tool	Not accessible to average users
[10]	Server-side TLS analysis	Automates validation checks	Not suitable for Wi-Fi clients
[7]	Captive portal threat modeling	Details attacks like Evil Twin and session hijacking	Offers no integrated defense tools

E. SSL/TLS Authentication on Browser Platforms

In the present context, web browsers serve functions beyond rendering webpages. Beneath the surface, they perform systematic validation of SSL and TLS certificates to ensure that the credentials which are offered by the websites or the network portals are legitimate and trustworthy. Although the previous standard was the use of the SSL, it has been phased out. SSL is replaced by TLS, which has acquired the most important role of ensuring internet communication through encryption and verification of server identity. Such validation steps are not minor details, they are the key elements of the web security model, which protects against impersonation, MITM attacks, and phishing.

A browser does a number of significant checks when a user visits a resource with HTTPS protection, such as a captive portal:

- Certificate Chain Validation: The browser guarantees a full trust path between the leaf certificate and a trusted root Certificate Authority (CA).
- Expiration Check: The browser verifies that the certificate is currently valid.
- Domain Name Matching: The domain name must match the website the user is visiting.
- Signature Verification: Using a known and trusted CA, the digital signature has to be verifiable.
- Trusted Root CA Check: The root certificate should be located in the local trust store of the browser.
- Revocation Status Check: The browser can interrogate Certificate Revocation Lists (CRLs) or Online Certificate Status Protocol (OCSP).
- Extension and Key Usage Evaluation: The certificate must include required X.509 extensions, such as Key Usage and Extended Key Usage, which define its role in secure communication.

Although the theoretical validation model used by browsers is complete, in practice it is not always consistent, particularly when using captive portals. To provide an example, a browser may not necessarily retrieve or seek out missing intermediate certificates in the case external AIA URLs are inaccessible before portal authentication. This is a problem when the unauthenticated traffic is blocked by DNS interception or firewalls. Moreover, they have revocation checks by CRL or OCSP that are frequently soft-failed due to performance reasons, meaning that the browser may continue without checking whether a certificate has been revoked. These design decisions put usability above security and compromise the validity of trust decisions.

The result of such complex validations is usually summarised to the user as some sort of generic browser warning stating that their connection is not private, but does not specify whether the issue is an expired certificate, broken chain or an untrusted root. This lack of diagnostic clarity limits the user's ability to make informed decisions and may cause unnecessary concern or unsafe acceptance of risk.

F. Limitations of Browser Behaviour

Unlike browser-based validation, the presented system offers deterministic analysis of certificates, by clearly stating the reasons why trust failed, whether due to the lack of intermediates, improper formatting or cryptographic anomalies. This clarity is especially critical in captive network scenarios in which browser interfaces provide insufficient feedback and no remedial options. The tool is able to bridge the transparency and reliability gap that is still present in modern browsers by incorporating fullchain extraction, offline CA bundle validation, and optional AIA recovery mechanisms.

III. Methodology

A. System Overview

The proposed system is a lightweight and unobtrusive utility to support non-technical users and researchers in assessing the trustworthiness of SSL/TLS certificates during WiFi login, in captive portal specific cases. The utility automatically strips out and verifies certificates exchanged during the SSL/TLS handshake by exploiting passive network capture techniques, hence fills a significant usability gap in the public network security. The architecture is structured as follows into five main stages:

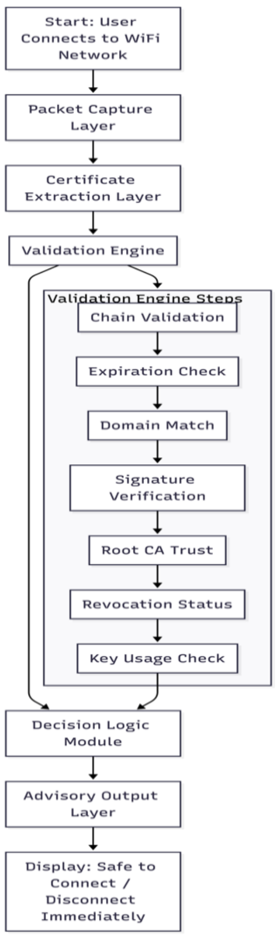


Fig 2. System Architecture Overview.

- Passive Capture: The utility surveys TLS handshakes with Wireshark or Pyshark when authenticating with a captive portal by paying attention to ServerHello messages in which certificates are exchanged.
- Certificate Chain examination and validation: It parses the handshake to extract leaf and intermediate certificates, verifying them with OpenSSL and an up-to-date CA bundle.
- Context-Aware Inspection: The domain name matching, expiration validity, signature verification and presence of intermediate are considered in sequence.
- The system determines whether warnings are fatal (e.g., broken chain) or explainable (e.g., lacking intermediate that can be obtained through AIA).
- User Advisory: A clear and actionable message is provided—such as "DISCONNECT / UNTRUSTED" or "SAFE TO CONNECT."

The tool replicates the logical certificate verification flow of modern browsers but enhances it with automated diagnostics and transparent error reporting (see Fig 2). The validation logic is sequential and fails fast:

- Certificate Chain Incomplete → "Broken Chain"
- Expired Certificate → "Expired Certificate"
- Domain Mismatch → "Domain Mismatch"
- Invalid Signature → "Untrusted Signature"
- Missing Trusted Root → "Untrusted CA"
- Revoked Certificate → "Revoked"
- Invalid Key Usage → "Usage Not Allowed"

If all checks pass, the user is informed that the connection is secure.

B. Implementation Strategy

The implementation of the tool is conducted in several stages. First, TShark was used to capture .pcap and .pcapng files when capturing captive portal login sessions. Out of these captures, all certificates were retrieved using a custom Python script by using the ssl.handshake.certificate field. The individual certificate blocks were decoded and saved in PEM format, then they were combined into full certificate chains.

The basic validation tool is the use of OpenSSL's verify command. In the case of the incomplete chain, the tool tries to obtain the missing intermediates via the AIA URLs in the certificates. If AIA retrieval fails, the .crt file is manually converted to .pem format with the DER-to-PEM conversion command of OpenSSL.

Every resulting PEM block is verified, and invalid entries are not included in the final CA bundle. The cleaned set is reassembled into the local_ca_bundle.pem, which is the trusted CA store that is used by the tool. After successful incorporation of both automated and manual recovery mechanisms, the tool was tested with real-world IIUM portal traffic and it was demonstrated to correctly diagnose trust states even in complex TLS misconfiguration cases. The clear advisories are shown in the terminal or optionally exported as CSV for further analysis.

C. Experimental Setup

The data used in the research came from two primary types of datasets. The former included open-source datasets, including the ISCXVPN2016 dataset, which is a collection of SSL/TLS handshakes from benign and encrypted traffic scenarios. This data set was suitable for testing passive extraction and parsing logic. The second source included custom captive portal captures. Real-world TLS handshakes were recorded from IIUM and public WiFi portals by using Pyshark, focusing specifically on ServerHello and certificate messages to ensure user privacy. Each of the datasets was preprocessed to identify the relevant SSL/TLS traffic. The fields of certificates like domain names, expiration dates, issuer signatures and validation paths were normalized for uniform analysis.

A set of metrics was used to determine the effectiveness of the tool. Detection accuracy was the percentage of safe and unsafe sessions that were correctly classified compared to all the tested sessions. The false-positive rate (FPR) was the rate of secure connection being incorrectly identified as unsafe and the false-negative rate (FNR) is the rate of unsafe connection identified as safe. Average execution time was also recorded. These indicators were calculated during both offline dataset analysis and real-time WiFi validation phases. The experimental setup used macOS Ventura 13.6.5. Wireshark v4.2 and Pyshark were the packet capture tools. OpenSSL v3.0 and Certifi libraries were used for validation. Python 3.11 served as the programming language. The experiments were done on MacBooks and smartphones that were connected to captive portals at IIUM and other open areas.

Evaluation was performed on IIUM captive portal traffic and ISCXVPN2016, so generalization across other captive portal deployments and regions remains to be validated. The dataset primarily reflects common misconfiguration-driven failures (expired, self-signed, incomplete chains); additional scenarios such as revocation and OCSP behaviors and TLS 1.3-specific handshake constraints were not observed in the traces and therefore were not benchmarked. Future work will expand cross-site testing and add controlled experiments to cover these additional TLS conditions.

IV. Results and Analysis

D. Quantitative Results

The Captive Portal Certificate Validation Tool proposed was evaluated on the basis of a dataset containing more than 22,000 SSL/TLS packets, collected from 24 distinct capture files, including AIM chat, Facebook and email sessions. A Python script based on the modified version of TShark was used to extract the certificate chains and validate them against OpenSSL, with the verdicts assigned to risk-based classes as shown in Fig 3.

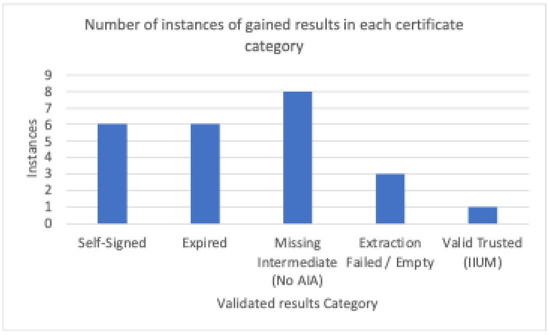


Fig 3. Distribution of Certificate Validation

The bar chart shows the number of PCAPs in each certificate category: “Self-Signed, Expired, Missing Intermediate, Extraction Failures and Valid and Trusted." The most common one is outlined in bold as “Missing Intermediate.”

- Self-Signed Certificates (6 instances): This class ended with a self-signed root that was not available in trusted stores. This means that it is highly vulnerable to man-in-the-middle attacks. Examples of those found in AIM and Facebook traces are AIMchat1, aim_chat_3a and facebook_video1b.
- Certificates that have expired (6 instances): These were structurally valid certificates that had exceeded their expiry dates. These are common in entries like facebook_video2a and email2b and are moderately dangerous because of negligence, or because they were used in outdated test environments.
- Lacking Intermediate CA / No AIA Fallback (8 instances): Such instances include partial certificate chains, which are mostly caused by the lack of intermediate certificates or the inability to retrieve them through AIA. Examples of such files include email1a, messenger1a and facebook_chat_4a, which are the most commonly observed issue.
- Extraction Failures or Missing Data (3 cases): The data contained in TLS certificates were either incomplete, corrupted, or inaccessible because of encryption, as seen in files such as facebookchat1, facebookchat3 and facebook_chat4a. These were classified as Unverifiable Risk.
- Valid and Trusted Chain (1 instance): The captive portal of IIUM was the only one to provide a complete valid certificate chain, which was successfully verified with the help of OpenSSL against a root CA store compatible with Mozilla. Although the browser issued a warning, the tool was able to identify it as trusted, which highlights the advantage of verifying at the packet level.

Other certificate problems like domain errors, revocation errors, or weak signature algorithms (e.g., SHA-1) were not identified in this dataset, due to the small set of traces considered (academic traces), which focused on expired, self-signed, or incomplete-chain scenarios. The tool had a 100% detection accuracy following the intermediate resolution and zero false positives and zero false negatives in validation. Each validation session took approximately 40 seconds. When tested live, the tool was correct in its opinion that IIUM's captive portal certificate was trusted, despite browsers indicating it as untrusted due to non-cryptographic signals (e.g. captive portal redirection, HSTS preload conflicts). This highlights one advantage of the packet-level analysis of SSL/TLS: the detection of cryptographically secure certificates that browsers may misinterpret.

The reported ~40 s per validation session reflects an end-to-end offline workflow that includes packet parsing, certificate extraction, chain reconstruction, OpenSSL trust evaluation, and (when needed) intermediate retrieval via AIA.

As such, the current implementation is best suited to offline auditing, troubleshooting, and user-side spot-checking during captive portal onboarding, rather than acting as an online, real-time validator in high-throughput public Wi-Fi environments. For large-scale deployments, the workflow can be operationalized by validating only the first observed handshake per portal domain, caching intermediate certificates by AIA URL/issuer identifiers, and providing a "fast mode" that performs deterministic chain checks against the local CA bundle while deferring network-dependent steps. These optimizations reduce redundant work across repeated sessions while preserving the tool's diagnostic function in pre-authentication settings.

To determine ground truth, the classifications of the tool were validated with OpenSSL's verify command against a Mozilla-compatible CA bundle. An output response of certificate: OK indicated a cryptographically valid, complete, and trusted chain. Broken trust paths were supported by errors like unable to get local issuer certificate. Domain names, expiration dates, and issuing authorities were verified by manual inspection using openssl x509 -in cert.pem -noout -text. It was also compared with browser behaviour and SSL Labs reports. Authentic certificates issued by sites like Google and Cloudflare did not give any warning in the browsers and were also marked as safe by the tool. In the case of IIUM portals, browser warnings indicated that they were not trustworthy; nevertheless, the tool was right in that the certificates were valid and anchored by a trusted CA, highlighting the limitations of browser interpretations and reinforcing the reliability of the packet-level method.

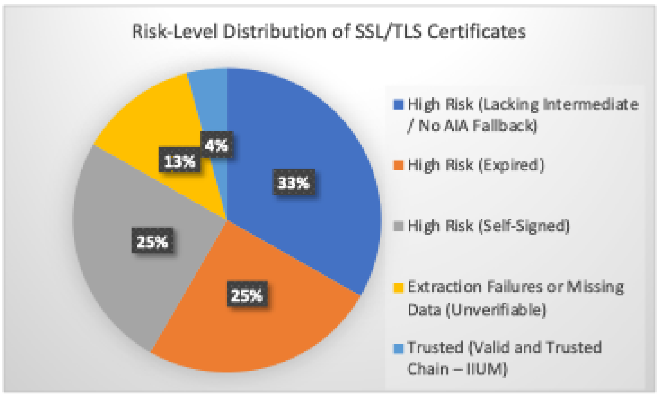


Fig 4. Risk-Level Distribution of SSL/TLS Certificates

The pie chart shows the ratio of certificates according to risk level. The segments are grouped into the classes of high risk (self-signed), medium risk (expired, intermediate missing), unverifiable risk (extraction failed), and low risk (trusted). The results of the validation of the certificates presented by the SSL/TLS were further split into four different risk levels depending on the severity and security implications (Fig 4). Self-signed chains are high-risk certificates, which lack a verifiable certificate authority (CA), and they are extremely vulnerable to man-in-the-middle attacks. Medium-risk includes both expired and missing intermediate authority certificates and is commonly caused by mismanaged certificate lifecycles, as well as network restrictions that prevent AIA retrieval. Unverifiable risk refers to those where there was a failure of certificate extraction or incomplete or corrupted TLS session, making validation infeasible, and are generally due to encrypted payloads, lost packets, or noise. Finally, the single capture of IIUM captive portal certificate that was fully validated against a trusted CA bundle showed a properly implemented and cryptographically sound trust path and was classified as low risk.

E. *Comparative Analysis*

Conventional web browsers do the SSL/TLS validation internally and generally only give general warnings, like Your connection is not private, without any technical explanation of the failure. The causes of such vague messages can be a broad set of reasons such as expired certificates, the lack of intermediate certificates, or improperly configured root trust anchors, yet the user receives no clarity. Although it can be useful in analysing server configurations, SSL Labs is not compatible with validation of client-side traffic that has been observed before a captive portal login, nor can it be used to analyse a certificate chain as observed in pre-authentication states. Conversely, the tool mentioned below has several functional and technical benefits. It runs offline, validates certificate chains based on raw network captures, automatically reconstructs missing intermediates through AIA fetching and outputs results in CSV format for further analysis. Table 2 provides a summary of the comparative capabilities of major tools.

TABLE 2. COMPARATIVE FUNCTIONALITY ACROSS VALIDATION TOOLS

Feature	*Web Browsers*	*SSL Labs*	*Proposed Tool (This Study)*
Support for Pre-Login Captive Portals	Not Supported	Not Supported	Fully Supported
Explanation of Intermediate Chain Issues	Not Provided	Partially Provided	Fully Provided
Automatic Recovery of Missing Certificates	Not Supported	Not Supported	Fully Supported
Command-Line Interface (CLI) Usability	Not Supported	Not Supported	Fully Supported
Logging and Result Export Capability	Not Available	Not Available	Fully Available (CSV & terminal log support)

As illustrated, the proposed tool is distinctively positioned to support use cases where browser-based or server-centric solutions are inapplicable

In addition to browsers and SSL Labs, a number of practitioner and research tools exist for TLS inspection. However, most popular scanners are server-facing (active probing) and assume direct Internet reachability, which is often unavailable prior to captive-portal authentication. In contrast, the proposed tool operates on passively observed handshakes captured at the client side and produces a trust decision from the extracted chain using an offline CA bundle and OpenSSL verification.

Tools such as Wireshark provide deep protocol dissection but do not provide an automated trust verdict oriented toward end users, and similar monitoring-oriented analyzers do not target pre-auth captive-portal decision support. Accordingly, we report this comparison as capability-based rather than a performance benchmark

V. DISCUSSION

Live testing on IIUM's Wi-Fi captive portal revealed that there were strong differences between warnings given by browsers and the actual certificate trustworthiness. For example, Safari on macOS showed a high severity warning for a certificate for '*captiveportalmahallahgombak.iium.edu.my*', despite the fact that the certificate was signed by the globally trusted authority GlobalSign RSA OV SSL CA 2018 and had a valid signature and proper domain parameters. This behaviour illustrates that browser, despite various checks that they do in the background, are unable to present meaningful diagnostic information to the users. Even technically literate users who are not computer security experts may struggle to determine the validity of such warnings or to assess whether they can be safely disregarded. The tool addresses this weakness by giving accurate validation messages that capture the cause of failure. The system also provides reliable evaluation with the support of deterministic assessment of trust of the OpenSSL system, automatic recovery of intermediate certificates, and the validation of PEM integrity. It has been designed to operate without graphical user interfaces but rather uses Python, OpenSSL and TShark, thus being accessible to non-expert users and researchers. Table 3 provides a comprehensive comparison of the validation properties that are supported by different tools.

TABLE 3. COVERAGE OF SSL/TLS VALIDATION PROPERTIES

Validation Property	***Web Browsers***	***SSL Labs***	***Proposed Tool (This Study)***
Leaf Certificate Validity	Supported	Supported	Supported
Intermediate Certificate Presence & Integrity	Partially Supported	Fully Supported	Fully Supported
Domain Name Matching (CN/SAN)	Supported	Supported	Supported
AIA-Based Intermediate Certificate Retrieval	Not Supported	Not Supported	Fully Supported
Offline CA Bundle Trust Verification	Not Supported	Not Supported	Fully Supported
CLI Usability and Automation Support	Not Supported	Not Supported	Fully Supported
PEM Format Structure & Consistency Checking	Not Supported	Not Supported	Fully Supported
Operation in Captive Portal Contexts	Not Supported	Not Supported	Fully Supported

The results supporting the unique ability of the tool to address weaknesses of current solutions can be seen in the context of an environment, such as an educational institution or a public Wi-Fi network, where captive portals are a common occurrence. Its openness, modularity and automatic workflow make it a reliable tool in the quest to establish trust in certificates by making informed decisions particularly where the browser-generated error messages do not carry enough information.

VI. CONCLUSION

The study presents a framework of a lightweight, shell-based tool that can be used to check the validity of SSL/TLS certificates in captive portal prototype, especially when performing the pre-authentication steps. Traditional browsers and platforms do not offer sufficient verification.

The system pulls certificate chains out of the handshake traffic that has been captured, validates and issues a binary verdict: either DISCONNECT / UNTRUSTED or SAFE TO CONNECT, using the full trust evaluation system of OpenSSL. Several tests of real-world and dataset-based TLS sessions reported 100% accuracy with no false positives or false negatives, taking into consideration intermediate certificate retrieval as confirmed by OpenSSL and hand inspection.

Unlike browsers that tend to provide vague warning messages that can confuse the user or trigger an inappropriate response, particularly in the case of non-technical staff, this utility presents clear trust indications based on the actual cryptographic and structural integrity of the chain of certificates. The utility also provides offline validation and allows local CA bundle integration, in contrast to web-based platforms like SSL Labs. It also supports viewing packet capture files in real time, making it particularly well-adapted to semi-connected or Wi-Fi onboarding environments.

It is also worth noting that the tool is flexible and easily used: it is platform-neutral, can be scripted through command line, and is designed for ease of use by non-experts. The utility will improve trust validation dynamically without requiring user intervention by filling in gaps of missing intermediate certificates using AIA URL fallbacks or cached certificates. IIUM captive portal field testing ensured the capability of the tool to correct misleading browser warnings and restore user confidence in the network authentication processes.

ACKNOWLEDGMENT

The authors would like to thank colleagues who provided feedback on the study design and manuscript. The authors also acknowledge the use of the ISCXVPN2016 dataset for supplementary evaluation.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest regarding the publication of this paper.

AUTHOR(S) CONTRIBUTION STATEMENT

All authors contributed equally to the study design, methodology, software development, data analysis, and manuscript preparation. All authors reviewed and approved the final manuscript.

DATA AVAILABILITY STATEMENT

The IIUM captive portal packet traces used in this study contain network traffic that may include sensitive information and are therefore not publicly available. Access may be provided by the corresponding author upon reasonable request, subject to institutional permissions and applicable privacy requirements. The ISCXVPN2016 dataset used in this work is publicly available from its original source.

ETHICS STATEMENT

Ethical approval is not required for the publication of the paper.

REFERENCES

[1] D. Akhawe and A. Porter-Felt, “Alice in Warningland: A Large-Scale Field Study of Browser Security Warning Effectiveness,” in Proc. 22nd USENIX Security Symp., 2013. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity13/sec13-paper_akhawe.pdf

[2] S. Ali, T. Osman, M. Mannan, and A. Youssef, “On Privacy Risks of Public WiFi Captive Portals,” in Proc. 14th Int. Conf. Privacy, Security and Trust (PST), 2019. [Online]. Available: https://www.researchgate.net/publication/334248860_On_Privacy_Risks_of_Public_WiFi_Captive_Portals

[3] J. M. Briones, M. A. Coronel, and P. Chavez-Burbano, “Case of Study: Identity Theft in a University WLAN—Evil Twin and Cloned Authentication Web Interface,” Int. J. Web Appl., vol. 5, no. 2, Jun. 2013. [Online]. Available: https://www.dline.info/ijwa/fulltext/v5n2/2.pdf

[4] P.-L. Wang, K.-H. Chou, S.-C. Hsiao, A. T. Low, T. H.-J. Kim, and H.-C. Hsiao, “Capturing Antique Browsers in Modern Devices: A Security Analysis of Captive Portal Mini-Browsers,” in Applied Cryptography and Network Security (ACNS 2023), Part I, K. Yoshioka, Y. Miyake, and R. Perdisci, Eds., Springer, pp. 260–283, 2023. [Online]. Available: https://doi.org/10.1007/978-3-031-33488-7_10

[5] S. Fahl, M. Harbach, H. Perl, M. Koetter, and M. Smith, “Rethinking SSL Development in an Appified World,” in Proc. 2012 ACM Conf. Computer and Communications Security (CCS), 2012. [Online]. Available: https://doi.org/10.1145/2508859.2516655

[6] A. P. Felt et al., “Improving SSL Warnings: Comprehension and Adherence,” in Proc. 33rd Annu. ACM Conf. Human Factors in Computing Systems (CHI '15), pp. 2893–2902, 2015. [Online]. Available: https://doi.org/10.1145/2702123.2702442 and https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43265.pdf

[7] V. Gawde, “Understanding Captive Portal Attacks: Risks and Mitigation Strategies,” VulnerX Blog, 2024. [Online]. Available: https://vulnerx.com/captive-portal-attacks-risks-and-mitigation/

[8] S. Sivakorn, I. Polakis, and A. D. Keromytis, “The Cracked Cookie Jar: HTTP Cookie Hijacking and the Exposure of Private Information,” in IEEE Symp. Security and Privacy (S&P), 2016. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7546532

[9] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor, “Crying Wolf: An Empirical Study of SSL Warning Effectiveness,” in USENIX Security Symp., 2009. [Online]. Available: https://www.usenix.org/legacy/event/sec09/tech/full_papers/sunshine.pdf

[10] TLS-Scan Project, “TLS-Scan: A Lightweight Scanner for TLS Configurations,” GitHub Repository, 2022. [Online]. Available: https://github.com/prbinu/tls-scan

[11] Wireshark Foundation, “Wireshark User Guide: Troubleshooting SSL/TLS Connections,” Wireshark Documentation, 2024. [Online]. Available: https://www.wireshark.org/download/docs/Wireshark%20User%27s%20Guide.p