JARAMOGI OGINGA ODINGA UNIVERSITY

OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATICS AND INNOVATIVE SYSTEMS

PROJECT TITLE:

SSL/TLS CERTIFICATE TRACKING IN REVEALING FORGED CERTIFICATES

PRESENTED BY:

NGARI EUSTUS MUTUGI

ADMISSION No: I 132/0877/2013

AN APPLICATION PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF BACHELOR OF SCIENCE IN COMPUTER SECURITY AND FORENSICS

SUPERVISORS

MR. ABUONJI PAUL

DECLARATION BY THE STUDENT

This project proposal is my original work and has not been presented for any award of any degree in any other college or university.

NGARI EUSTUS MUTUGI

ADM No: I 132/0877/2013

Sign…………………………. Date…………………

DECLARATION BY SUPERVISORS

This project proposal has been submitted for examination with our approval as the candidate's/University supervisors.

MR. PAUL ABUONJI

Sign…………………………… Date………………….

Dedication.

I dedicate SSL/TLS Certificate Tracking in Revealing Forged Certificates to Dad, Mum and the Browsing Secure Community who have always wanted a safer world.

Abstract.

SSL/TLS (Secure Sockets Layer/Transport Layer Security) Certificate Tracking in Revealing Forged Certificates is based on implementing two chrome based security solutions meant set to uncover and evade the threats that results from unencrypted connections or by use of a forged SSL/TLS certificates. To ensure that the client browser connects via encrypted HTTPS (Secure HyperText Transfer Protocol) connections we have come up with a way to ensure that all user requests shall be redirected to HTTPS. Also, through monitoring of SSL/TLS certificate we will be able to verify the integrity of the certificate. Moreover, there are trackers to website that handles user data or have the ability to gather client related data. These trackers if they set to connect over HTTP unencrypted connection they may compromise the security of the of the client regardless the ability of the client to connect over secure HTTPS connections. Therefore, there was a need to connect the trackers mandatory over HTTP + SSL (HTTPS) encrypted connection.

The development of the security solution was made possible by the utilization of agile methodology and in our case Scrum Methodology proved to be useful. The software development took 35 days despite the change requests made. This was made possible by the adoption of Scrum in the software development process.

Strict SSL and Cert Monitor chrome browser extensions and application respectively, provides the user with added security features in your browsing experience and greatly improves your privacy online. This is also made as transparent and automatic as possible. Development was majorly done on the chrome browser as it the leading in usage statistics to date (Dec 2016). The chrome app and extensions should be deployed to any web user to help fight against man-in-the-middle attacks that are faced daily by every internet user and many a times without his/her consent. The information provided by my chrome extensions can allow the user to easily validate the security of his/her browsing experience. My chrome app and extension have the advantage of being lightweight, informative, and simple to install.

Acknowledgement

I would like to extend my heartiest thanks with a deep sense of gratitude and respect to all those who provides me immense help and guidance during my training period.

I would like to thank my Project Leader Mr. Paul Abuonji for providing a vision about the system. I have greatly benefited from his regular critical reviews and inspiration throughout my work.

I would also like to thank my friends, who have already formed a browsing secure community for their unfailing cooperation and sparing their valuable time to assist me in my user testing and discussion on online privacy.

I would like to express my sincere thanks to our Director of ICT department, Prof. Anthony Rodrigues, the Dean of the School of Informatics and Innovative Systems, Dr. Abeka and my internal guide Mr. Paul Abuonji, who gave me an opportunity to undertake such a great challenging and innovative work. I am grateful to them for their guidance, encouragement, understanding and insightful support in the development process.

I would also like to thank Dr. Ogara for providing us with a super informative guide on how to carry out the system development process and a template documentation on how to get the job done.

I am also thankful to entire staff of School of Informatics for their constant encouragement, suggestions and moral support throughout the duration of my project.

I would also like to express my gratitude to Dr. Bernard Ngari, who provided financial support in every aspect of the project conceptualization to project articulation.

Last but not the least I would like to mention here that I am greatly indebted to each and everybody who has been associated with my project at any stage but whose name does not find a place in this acknowledgement.

With sincere regards,

NGARI EUSTUS MUTUGI

TABLE OF CONTENTS

Dedication. ii
Abstract. iii
Acknowledgement iv
List of Figures x
List of Tables xi
CHAPTER 1 1
1.0 Introduction 1
1.1 Background Information. 3
1.1.1 The SSL Protocol 3
1.1.2 The SSL Man-in-the-Middle Attack 5
1.2 Problem Statement 6
1.3 Objectives of the Study 7
1.3.1 Main Objective 7
1.3.2 Specific Objectives 7
1.4 Research Questions. 7
1.5 Scope of the Project 7
1.6 Justification 7
1.7 Assumptions. 8
1.8 Tools and Technologies used. 8
CHAPTER 2 9
2.1 Literature Review 9
2.1.1 Web Tripwires 9
2.1.2 Content Security Policy. 9
2.1.4 HTTP Strict Transport Security (HSTS) 11

2.1.5 Certificate validation by Notaries. 11
2.1.5 Certificate Transparency. 12
CHAPTER 3 13
3.1 Scrum Methodology. 13
3.1.1 Overview. 13
3.2 Scrum Artifacts. 14
3.2.1 The Product Backlog. 15
3.2.2 The Project Burn down Chart. 16
3.2.3 The Sprint Backlog. 17
3.2.4 The Sprint Burn Down Chart 18
3.2.5 Impediment list. 19
3.3 Sprint 19
3.4 The Sprint Planning Meeting. 19
3.5 The Daily Activities. 19
3.6 Sprint Review Meeting 19
3.7 Sprint Retrospect Meeting 20
3.8 Project Startup 20
3.9 Project Completion 20
CHAPTER 4 21
Software Analysis and Requirement 21
4.1 Study of the current system. 21
4.1.1 The Trouble with HTTP Strict Transport Security (HSTS) 21
4.1.2 The Trouble with Certificate Transparency. 22
4.1.3 Requirements of the new system. 22
4.1.4 Functional System Requirements 23

4.1.5 Non-Functional Requirements 24

4.1.6 Feasibility Study. 24

4.1.7 Functions of the system 25

4.1.7.1 Strict SSL use case diagram. 25

4.1.7.2 Cert Monitor use case diagram 26

4.1.7.3 Cert Monitor Sequence Diagram 26

4.1.8 Hardware and Operating system considerations. 27

4.1.9 Programming languages, libraries and chrome APIs used 28

Software Design 30

4.2 Introduction 30

4.2.1 Designs considerations during problem solving. 30

4.2.2 Rationale for choosing the design models. 30

4.2.3 Chrome Browser App Architecture. 31

Requirement 1. App Container Model. 31

Requirement 2. Programming model 32

Requirement 3. App lifecycle 32

Requirement 4 Security model. 34

4.2.4 Strict SSL Design. 34

4.2.5 Certificate Monitor Design. 36

Software Implementation. 37

4.3 Software Implementation Environment. 37

4.3.1 4-Tier Implementation architecture 37

Layer 1. The presentation tier or user services layer 38

Layer 2. The control tier or control layer. 38

Layer 3. The business services layer. 38

Layer 4. The data tier or data services layer. 38

Software Development Testing 39

4.4 Testing Plan. 39

4.4.1 The Testing Process 40

4.4.2 Requirement Traceability 40

4.4.3 Testing Schedule 41

4.4.4 Test Strategy 41

4.4.5 Unit Testing 41

4.4.6 Integration Testing 43

4.4.7 System Testing 43

4.4.8 Performance Testing 44

4.4.9 Test Cases 44

4.4.5.1 Test case for Certificate Monitor 46

4.4.5.2 Test case for Certificate Monitor App Debugging 47

4.4.5.3 Test Case for Strict SSL Extension 47

CHAPTER 5 48

Discussion and Conclusion 48

5.1 Problems that Strict SSL and Cert Monitor set to solve. 48

5.2 How the Strict SLL set to ensure HTTP + SSL connections. 48

5.3 How the Certificate Monitor set to ensure the integrity of the of the certificate. 48

5.4 Problem Encountered in the system conceptualization towards system actualization. 49

5.4.1 General problems encountered. 49

5.2.3 Challenges faced during the development of the chrome browser application. 51

5.2.4 Challenges faced during Certificate Monitor app extension 52

5.2.5 How my application and extension have meet the objectives. 52

5.3.1 Recommendations 53

5.3.2 Future Implementation. 53

5.3.3 Conclusion 53

References 54

APPENDIX. 58

6.1 UML Code Syntax for StrictSSL Use Case diagram. 58

6.2 UML code Syntax for Certificate Monitor use case diagram. 58

6.3 UML code Syntax for Top-Down Testing Methodology Diagram. 59

6.4 UML code syntax for Certificate Monitor Sequence Diagram. 59

6.5 UML code syntax for StrictSSL Activity Diagram. 60

List of Figures

Figure 1. 1 Man-In-The-Middle attack using SSL stripping 2
Figure 1. 2 Man-in-The-Middle Attack using SSL sniffing 2
Figure 1. 3 A basic SSL handshake with no certificates........ 3
Figure 1. 4 An SSL man-in-the-middle attack between the browser and the server, using a forged SSL certificate to impersonate as the server to the client. 5

Figure 3. 1 Overview of Scrum Methodology 14
Figure 3. 2 product backlog from story readiness report 15
Figure 3. 3 Product backlog burndown chart........ 16
Figure 3. 4 project burndown chart with forecast 17
Figure 3. 5 Sprint backlog From the Story readiness report 18
Figure 3. 6 Sprint backlog burn down chart 18

Figure 4. 1 Strict SSL use case Diagram 25
Figure 4. 2 Certificate Monitor Use Case Diagram 26
Figure 4. 3 Certificate Monitor Sequence Diagram........ 26
Figure 4. 4 App container Model........ 32
Figure 4. 5 App life cycle 33
Figure 4. 6 Strict SSL Design 35
Figure 4. 7 Certificate Monitor Design........ 36
Figure 4. 8 4-Tier Implementation Architecture........ 37
Figure 4. 9 Top-down Test Methodology 39

List of Tables

Table 4. 1 Programming Languages Used 28
Table 4. 2 Chrome APIs Used 29
Table 4. 3 Libraries Used 29
Table 4. 4 App Lifecycle Table 33
Table 4. 5 Top-down Testing Methodology Description 40
Table 4. 6 Features to be Tested 42
Table 4. 7 Unit Testing for Strict SSL 43
Table 4. 8 Test Case for Certificate Monitor 46
Table 4. 9 Test case for chrome app debugging 47
Table 4. 10 test case for Strict SSL 47

Table 5. 1 General Problems Faced 49
Table 5. 2 Challenges faced during the development of Strict SSL chrome browser extension .. 50
Table 5. 3 Challenges faced during the development of the chrome browser application 51
Table 5. 4 Challenges faced during Certificate Monitor 52

CHAPTER 1
INTRODUCTION

1.0 Introduction

Online transactions and communications are reliant on the connection between client and server being secure. This has led to HTTPS becoming popular with online security, due to its simple layering of HTTP on top the older SSL (Secure Sockets Layer) or the more recent TLS (Transport Layer Security) protocols. The HTTPS (Secure Hyper Text Transfer Protocol) protocol is stated to provide cryptographic capabilities to web servers and corresponding web sites (Oppliger, 2009), with the latest version of TLS standards also stating that it is designed to prevent eavesdropping, tampering, and message forgery (Rescoria, 2008). However, the HTTPS protocol, and consequently the SSL/TLS protocols, have historically been compromised by many different types of Man-in-the middle attacks (MITM) such as SSL Stripping (Moxie, 2009) and use of forged certificates in SSL sniffing (Moxie, 2009).

SSL stripping refers to removing away the SSL/TLS data from a request message. It exploits the vulnerability of multiple open ports for same application server. MITM attacker redirects all requests to the application server through unsecure ports. Therefore, the data exchanged are unencrypted and hence unsecure. SSL stripping is also achieved by a penetration testing tool (Moxie, 2009), (Moxie, 2016) called the SSL strip. This tool removes the SSL/TLS based request messages from a client request and sends it to the server. The server assumes that the client does not support SSL/TLS and hence the established an insecure connection. This is really tragic because the client communicates to the server on plain text.

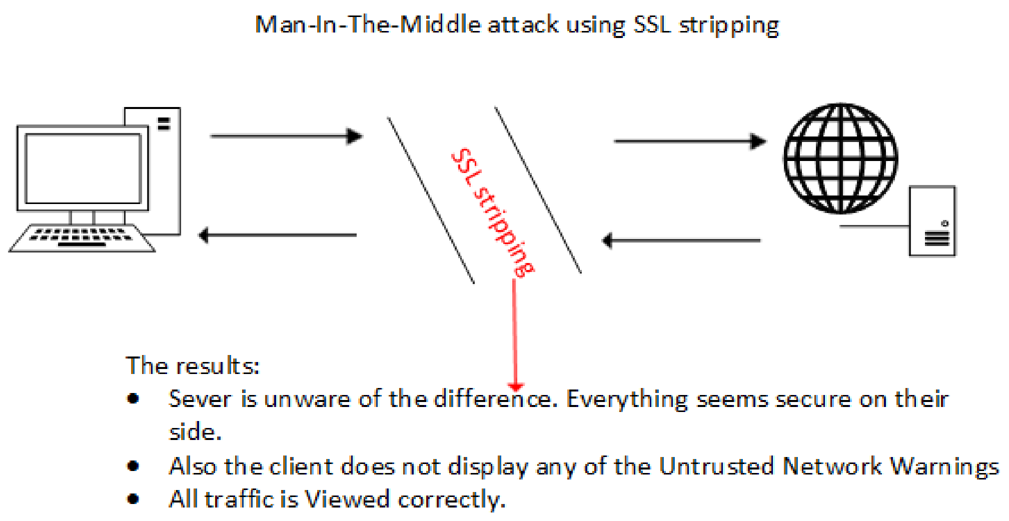

Man-In-The-Middle attack using SSL stripping
SSL stripping
The results:
• Sever is unware of the difference. Everything seems secure on their side.
• Also the client does not display any of the Untrusted Network Warnings
• All traffic is Viewed correctly.

Figure 1. 1 Man-In-The-Middle attack using SSL stripping

SSL Sniffing tries to intercept the communication channel by impersonating the server from the client's perspective and impersonating the client from the server's perspective. Numerous automated tools that can mount SSL man- in-the-middle attacks are publicly available on the Internet e.g. sslsniff (Moxie, 2011) and sslstrip (Moxie, 2011), which greatly reduce the level of technical sophistication necessary to mount such attacks.

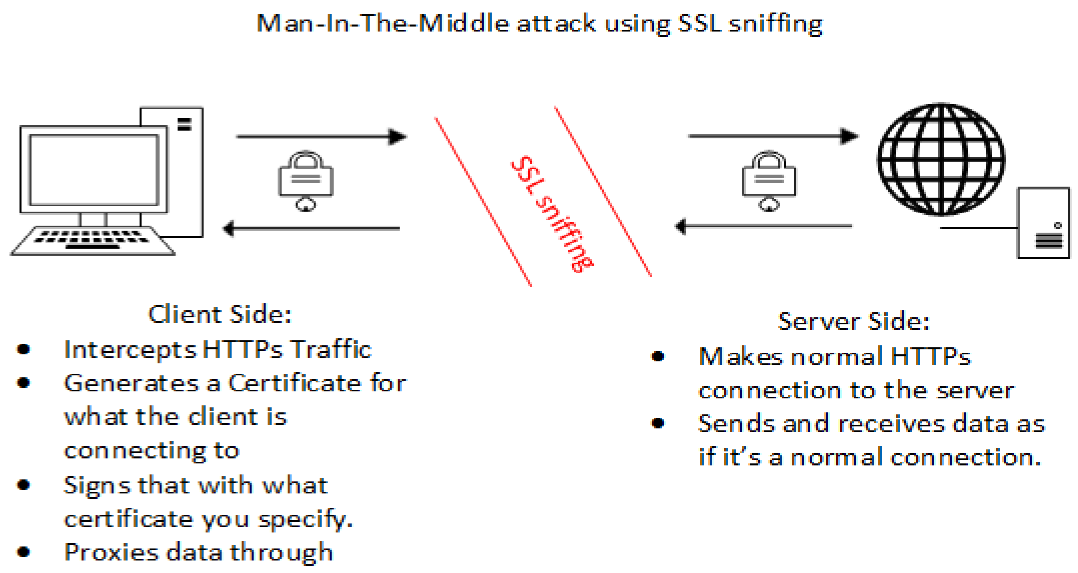

Man-In-The-Middle attack using SSL sniffing
SSL sniffing
Client Side:
• Intercepts HTTPs Traffic
• Generates a Certificate for what the client is connecting to
• Signs that with what certificate you specify.
• Proxies data through
Server Side:
• Makes normal HTTPs connection to the server
• Sends and receives data as if it's a normal connection.

Figure 1. 2 Man-in-The-Middle Attack using SSL sniffing

This is very concerning as the widespread adoption of HTTPS as best practice for securing websites, and having users becoming accustomed to the use of HTTPS, may lead to a false sense of security. In most cases, the client and server may never be aware that their security has been compromised, and that an attacker has complete control over the information crossing the network. Therefore, it is important to understand the manner in which these attacks are implemented and what can be done to mitigate them.

1.1 **Background Information**.

In this section, I provide an overview of the SSL protocol, and how the SSL man-in-the-middle attack works to circumvent encrypted connections over the Secure Hyper Text Transfer Protocol (HTTPS).

1.1.1 The SSL Protocol

The Secure Socket Layer (SSL) protocol was designed to ensure secure communications between two entities over untrusted networks. The SSL protocol provides authentication based on the X.509 public key infrastructure, protects data confidentiality using symmetric encryption, and ensures data integrity with cryptographic message digests.

To establish an SSL connection, the client and the server performs a handshake to authenticate each other as seen in the below diagram.

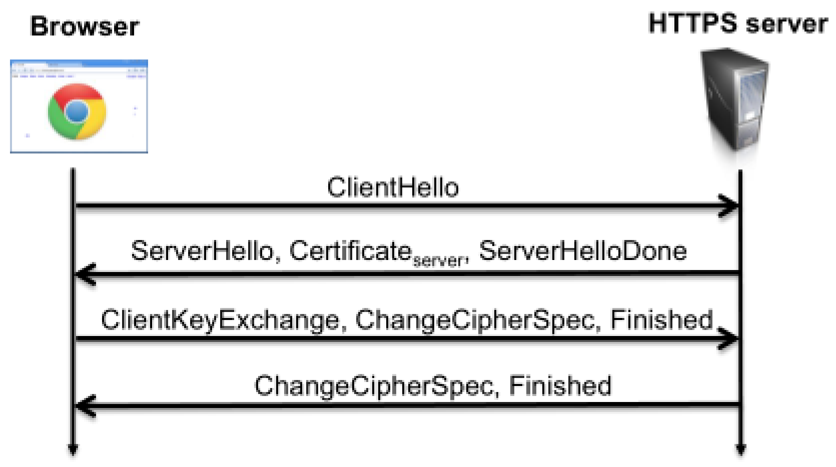

Browser
HTTPS server
ClientHello
ServerHello, Certificate$_{server}$, ServerHelloDone
ClientKeyExchange, ChangeCipherSpec, Finished
ChangeCipherSpec, Finished

Figure 1. 3 A basic SSL handshake with no certificates

1. The client sends a ClientHello message to the server, which specifies a list of supported cipher suites and a client-generated random number.

2. The server responds with the ServerHello message which contains the server-chosen cipher suite and a server-generated random number. In addition, the Certificate message contains the server's public key and hostname, digitally signed by a certificate authority, in which the client is responsible of verifying.
3. The client then encrypts the pre-master secret using the server's public key and sends the pre-master secret to the server over a ClientKeyExchange message. Both the client and server can hence derive the same session key from the pre-master secret and random numbers.
4. Finally, the client and server exchanges ChangeCipherSpec messages to notify each other that subsequent application data within the current session will be encrypted using the derived session key.

In practice, commercial SSL certificates are often signed by intermediate CAs (a delegated certificate signer), instead of directly signed by a trusted root CA (which are kept offline to reduce the risk of being compromised). Therefore, the server's Certificate message normally includes a chain of certificates, consisting of one leaf certificate (to identify the server itself), and one or more intermediate certificates (to identify the intermediate Certification Authorities). Each certificate is cryptographically signed by the entity of the next certificate in the chain, and so on. A valid certificate chain must chain up to a root Certificate Authority that is trusted by the client. Note that SSL certificates are by design transferred in plaintext since the integrity can be verified by signatures. It is critical that clients must validate every certificate in the chain. In the following section, we will explain why validating SSL server certificates is necessary.

1.1.2 The SSL Man-in-the-Middle Attack

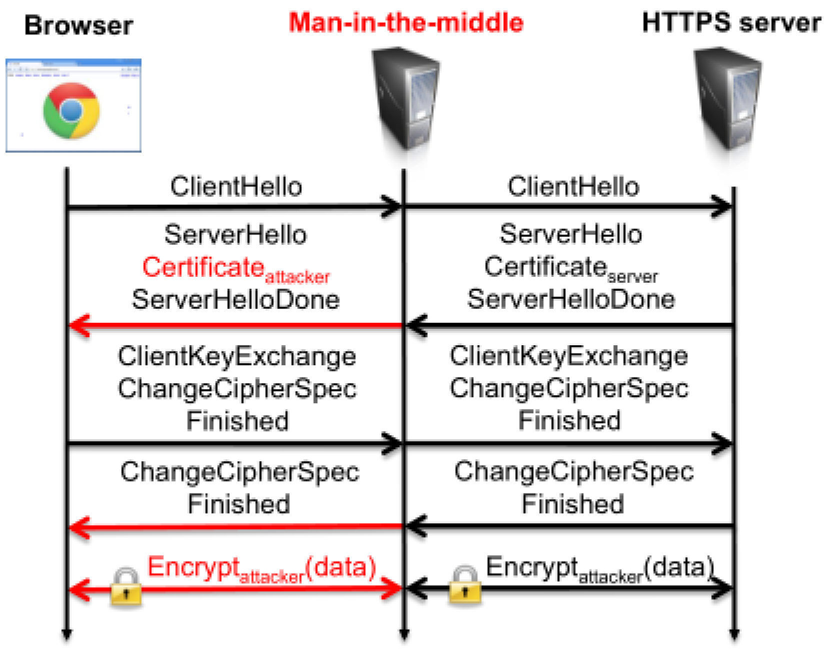


Figure 1. 4 An SSL man-in-the-middle attack between the browser and the server, using a forged SSL certificate to impersonate as the server to the client.

The SSL man-in-the-middle (MITM) attack is a form of active network interception where the attacker inserts itself into the communication channel between the victim client and the server (typically for the purpose of eavesdropping or manipulating private communications). The attacker establishes two separate SSL connections with the client and the server, and relays messages between them, in a way such that both the client and the server are unaware of the middleman. This setup enables the attacker to record all messages on the wire, and even selectively modify the transmitted data the figure above depicts an SSL man-in-the-middle attack with a forged certificate mounted between a browser and a HTTPS server. I describe the basic steps of a generic SSL man-in-the-middle attack as follows:

a. The attacker first inserts itself into the transport path between the client and the server, for example, by setting up a malicious Wi-Fi hotspot. Even on otherwise trusted networks, a local network attacker may often successfully re-route all of the client's traffic to itself using exploits like ARP poisoning, DNS spoofing, BGP hijacking, etc. The attacker could also possibly configure itself as the client's proxy server by exploiting auto configuration protocols (PAC/WPAD) (Chen, 2009). At this point, the attacker has gained control over the client's traffic, and acts as a relay server between the client and the server.

b. When the attacker detects an SSL ClientHello message being sent from the client, the attacker accurately determines that the client is initiating an SSL connection. The attacker begins the impersonation of the victim server and establishes an SSL connection with the client. Note that the attacker uses a forged SSL certificate during its SSL handshake with the client.
c. In parallel to the previous step, the attacker creates a separate SSL connection to the legitimate server, impersonating the client. Once both SSL connections are established, the attacker relays all encrypted messages between them (decrypting messages from the client, and then re-encrypting them before sending to the server). Now, the attacker can read and even modify the encrypted messages between the client and the server.

As soon as the client accepts the forged SSL certificate, the client's secrets will be encrypted with the attacker's public key, which can be decrypted by the attacker. In fact, professional attackers have proven capable of compromising CAs themselves in order to obtain valid certificates, as has occurred during the security breaches of DigiNotar (Vasco, 2011) and Comodo (Comodo, 2011). Moreover, even if the attacker does not have a trusted certificate of the victim server and uses a self-signed certificate, researchers have shown that many users ignore SSL certificate warnings presented by the browser (Sunshine, 2009). Even worse, studies have discovered that some non-browser software and native mobile applications actually contain faulty SSL certificate validation code, which silently accepts invalid certificates (Georgiev, 2012), (Muders, 2012), (Fahl, et al., 2013).

1.2 Problem Statement

Enhancing HTTPS security can be made possible by enabling consistent secure encrypted connection with the ability to evade man-the-middle (MITM) attacks such as the most devastating SSL stripping that opts the unencrypted connections or the more sophisticated certificate SSL Sniffing. With this in mind, I have developed an in-browser chrome browser application and extensions that will mitigate these adversaries. These adversaries leverage themselves on the network. With the ability to evade and terminate connections to this MITM attacks will help mitigate loss of confidentiality, Integrity and Availability of the Client-Server browser communications.

1.3 Objectives of the Study

1.3.1 Main Objective.

The main objective of this project is to enhance HTTPS security by enforcing HTTP connections to the rather appreciated HTTPS connections and revealing the integrity of the SSL/TLS certificate used on the HTTPS connections. All trackers to a website shall be mandatory for them to connect over HTTPS

1.3.2 Specific Objectives

i. To enforce HTTPS connections on depreciated HTTP connection
ii. To reveal the status of the SSL certificate on all trackers and the requested urls.
iii. To inform the user on a change in SSL certificate and inform the user.
iv. To mandatory enforce HTTPS connections on all trackers to a site.

1.4 Research Questions.

i. What are the role of HSTS (Strict Transport Security) on enforcing HTTPS connections?
ii. What are the techniques used to detect whether the client’s network connections have been tampered with?
iii. What are the techniques used to validate certificate used on HTTPS connections?
iv. What are the suggested ideas in certificate transparency?

1.5 Scope of the Project.

The scope of the project shall gear toward enforcing HTTPS connections and depreciating any HTTP or unencrypted connection. This shall also include all trackers to a site which shall be enforced to connect over HTTPS encrypted connection and reveal the status of the SSL/TLS certificate used in ensuring encrypted connections. The security solution shall also be informative to the user and simple to use. User interaction with the chrome app and extensions shall be limited to the whitelisting of only requested urls. This is because some sites do not support HTTPS connections.

1.6 Justification

SSL stripping is a much more serious attack that remains dangerous today. It is a man-in-the-middle attack that converts your HTTPS connections into HTTP connections. This attack is Devastating and difficult for users to detect. However, this can be enabled at the server by enabling strict transport security. HSTS is supported on all modern browsers. However, HSTS requires the site to opt into protection (which is a major disadvantage). This disadvantage has led to slow implementation of Strict Transport Security accounting for 95% of HTTP servers vulnerable to SSL Stripping attacks (Mutton, 2016).

Secondly, since SSL client certificates are rarely sent by normal users, it is not possible to distinguish a legitimate client from an attacker directly via the SSL handshake from the server's perspective. In order to determine whether an SSL connection is being intercepted, our fundamental approach is to observe the server's certificate from the client's perspective. Intuitively, if the client actually received a server certificate that does not exactly match the website's legitimate certificate, we would have direct evidence that the client's connection must have been tampered with.

I have decided to develop on google chrome as is the leading browser in usage and by this I will be able to reach a large number of internet users.

1.7 Assumptions.

i. Users are aware of basic security concepts such as Secure Hyper Text Transfer Protocol.
ii. Users want to avoid MITM without any noticeable penalty on browsing experience
iii. Attackers are sophisticated are always looking ways to compromise secure communications.
iv. The applications (browser, application, and extensions) are secure. This shall be done by packaging the chrome browser application and extensions with a private key.

1.8 Tools and Technologies used.

i. Google chrome Developer Tools Application
ii. Sublime text editor
iii. JavaScript Programming Language
iv. Google Chrome APIs (Application Programming Interfaces)
v. JSON (JavaScript Object Notation) is a lightweight data-interchange format.
vi. JavaScript Libraries such as moment.js, restruct.js, underscore.js etc.
vii. HTML (Hyper Text Markup Language)

CHAPTER 2
LITERATURE REVIEW

2.1 Literature Review

Several techniques have been proposed to assist websites in detecting whether the client's network connections have been tampered with. In this paper, we focus on detection methods that do not require user interaction, and do not require the installation of additional software or browser extensions. Notably, Web Tripwires (Reis, 2008) uses client-side JavaScript code to detect in-flight modifications to a web page. Several other studies (Freedman, 2007), (Jackson, 2007), (Huang, 2011), (Kreibich, 2010) have utilized Java applets to probe the client's network configurations and detect proxies that are altering the client's traffic.

2.1.1 Web Tripwires

Web Tripwires (Reis, 2008) was a technique proposed to ensure data integrity of web pages, as an alternative to HTTPS. Websites can deploy JavaScript to the client's browser that detects modifications on web pages during transmission. In their study of real world clients, over 1% of 50,000 unique IP addresses observed altered web pages. Roughly 70% of the page modifications were caused by user-installed software that injected unwanted JavaScript into web pages. They found that some ISPs (Internet Service Providers) and enterprise firewalls were also injecting ads into web pages, or benignly adding compression to the traffic. Interestingly, they spotted three instances of client-side malware that modified their web pages. Web Tripwires was mainly designed to detect modifications to unencrypted web traffic. By design, Web Tripwires does not detect passive eavesdropping (that does not modify any page content), nor does it detect SSL man-in-the-middle attacks. In comparison, our goal is to be able to detect eavesdropping on encrypted SSL connections.

2.1.2 Content Security Policy.

Content Security Policy (CSP) (Stamm, 2010) enables websites to restrict browsers to load page content, like scripts and stylesheets, only from a server-specified list of trusted sources. In addition, websites can instruct browsers to report CSP violations back to the server with the report-uri directive. Interestingly, CSP may detect untrusted scripts that are injected into the protected page, and report them to websites. Like Web Tripwires, CSP does not detect eavesdropping on SSL connections.

2.1.3 Browser Plugins.

Another technique for websites to diagnose the client's network is by using browser plugins, such as Java and Flash Player. Browser plugins may provide more network capabilities than JavaScript, including the ability to open raw network sockets and even perform DNS requests. For instance, the Illuminati (Jackson, 2007) project used Java applets to identify whether clients were connecting through proxies or NAT (Network Address Translation) devices. Jackson et al. conducted studies using both Java and Flash Player on real-world clients to find web proxy vulnerabilities, including multi-pin DNS rebinding (Jackson, 2007) and cache poisoning (Huang, 2011). The ICSI Netalyzer (Kreibich, 2010) used a signed Java applet to perform extensive tests on the client's network connectivity, such as detecting DNS manipulations. However, browser plugins have turned out to be one of the biggest security problems today (Hoffman, 2012).

The Flashback Trojan (Smith, 2012) infected over 600,000 Macs. It called the Java plugin from a web page and loaded a special Java applet that exploited a Java bug, gaining access to the system. Having Java installed increases your attack surface. Now picture a browser with multiple plugins – Java, Flash, PDF reader, QuickTime, Silverlight, Unity Web Player, RealPlayer (I'm sure some people still have that installed), and more – and you'll see just how much plugins increase your attack surface. Each plugin must be updated separately using its own update manager. While browser vendors are under heavy scrutiny to write secure code, plugin developers don't seem to have the same urge in during development, and many of them have atrocious security records.

The unfortunate thing about compromising a plugin is that you can compromise multiple platforms at once. Find a security hole in Flash and you're able to compromise nearly every browser on the planet – Internet Explorer on Windows, Safari on a Mac, Firefox on Linux – you can run wild.

My application being a chrome app is mandatory for it to follow the chrome app security model that sandboxes any activity of the app to the app and that of the chrome browser to itself i.e. a compromise of the browser may not affect the app and vice versa.

2.1.4 HTTP Strict Transport Security (HSTS)

HTTP Strict Transport Security (HSTS) (Hodges, 2012), the successor of ForceHTTPS (Barth, 2008), is a HTTP response header that allows websites to instruct browsers to make SSL connections mandatory on their site. By setting the HSTS header, websites may prevent network attackers from performing SSL stripping (Moxie, 2009). A less obvious security benefit of HSTS is that browsers simply hard-fail when seeing invalid certificates, and do not give users the option to ignore SSL errors. This feature prevents users from accepting untrusted certificates when under man-in- the-middle attacks by amateur script kiddies. However, HSTS is not designed to protect against malware or professional attackers that use forged certificates that would be accepted by the browser

2.1.5 Certificate validation by Notaries.

Also, there are some methods that has been designed to validate certificates by use of Notaries for example, Perspectives (Wendlandt, 2008) is a Firefox add-on that compares server certificates against multiple notaries (with different network vantage points) to reveal inconsistencies. Since public notaries observe certificates from diverse network perspectives, a local impersonation attack could be easily detected. Convergence (Marlinspike, 2011) extends Perspectives by anonymizing the certificate queries for improved privacy, while allowing users to configure alternative verification methods such as Domain Name System Security Extensions (DNSSEC). The DetecTor (Engert, 2013) project which extends Doublecheck (Keromytis, 2009) makes use of the distributed Tor network to serve as external notaries. Crossbear (Holz, 2012) further attempts to localize the attacker's position in the network using notaries. However, notary approaches might produce false positives when servers switch between alternative certificates, and clients may experience slower SSL connection times due to querying multiple notaries during certificate validation. Further, these pure client-side defenses have not been adopted by mainstream browsers, thus cannot protect the majority of (less tech-savvy) users.

2.1.5 Certificate Transparency.

Additionally, proposals have suggested the idea of maintaining cryptographically irreversible records of all the legitimately-issued certificates, such that mis-issued certificates can be easily discovered, while off-the-record certificates are simply rejected. Sovereign Keys (Eckersley, 2015) requires clients to query public timeline servers to validate certificates. Certificate Transparency (CT) (Laurie, 2012) removes the certificate queries from clients by bundling each certificate with an audit proof of its existence in the public log. Accountable Key Infrastructure (AKI) (Kim, 2013)further supports revocation of server and Certification Authority keys. These defenses are designed to protect against network attackers (not including malware). However, browsers need to be modified to support the mechanism, and changes (or cooperation) are needed on the CAs or servers to deliver the audit proof. Encouragingly, Chromium includes support for Certificate Transparency, a protocol defined by RFC 6962. Certificate Transparency is a way for interested parties, such as Certificate Authorities, to provide a publicly auditable record of certificate issuance, through submitting these certificates to a Certificate Transparency Log (Placeholder1). Google is making Certificate Transparency mandatory for its Chrome web browser by October 2017 (Laurie & Langley, 2012). Google software engineer Ryan Sleevi made the announcement in conjunction with the CA/Browser Forum that took place in Redmond, Washington (Sleevi, 2016).

CHAPTER 3
METHODOLOGY

3.1 Scrum Methodology

I have provided a scrum project management file that explicitly show all the development process using the agile methodology Scrum. The file is a Sprintometer (an agile project management tool) file.

Enhancing HTTPs Security against Sophisticated MITM attack Is a scope driven project that satisfies security need from the product backlog or the business needs of the product. According to Ken Schwaber (Schwaber, 2003), one of the originators of the Scrum method, scrum is a process for managing complex projects. He stresses that it isn't just limited to software development. However, software development projects have a tendency to be very complex (Brooks, 1978) and so Scrum is well suited for managing them.

3.1.1 Overview.

The Scrum method is incremental. Each increment is called a sprint and is recommended to last for four weeks. Before the sprint, there is a sprint planning meeting where the customer decides what features should be implemented in the upcoming sprint. During the sprint, the team meets daily at a short meeting called a scrum or the daily stand-up meeting. At the end of a sprint, a sprint review meeting is held where the customer gets to see what was accomplished during the sprint. The team can also hold a sprint retrospective meeting where they look at the process and tries to find out what went well and what can be improved. Schwaber uses Figure below to visualize the flow of the method. The upper circle represents the daily activities of the team members, while the lower circle represents the development activities that occur during a sprint.

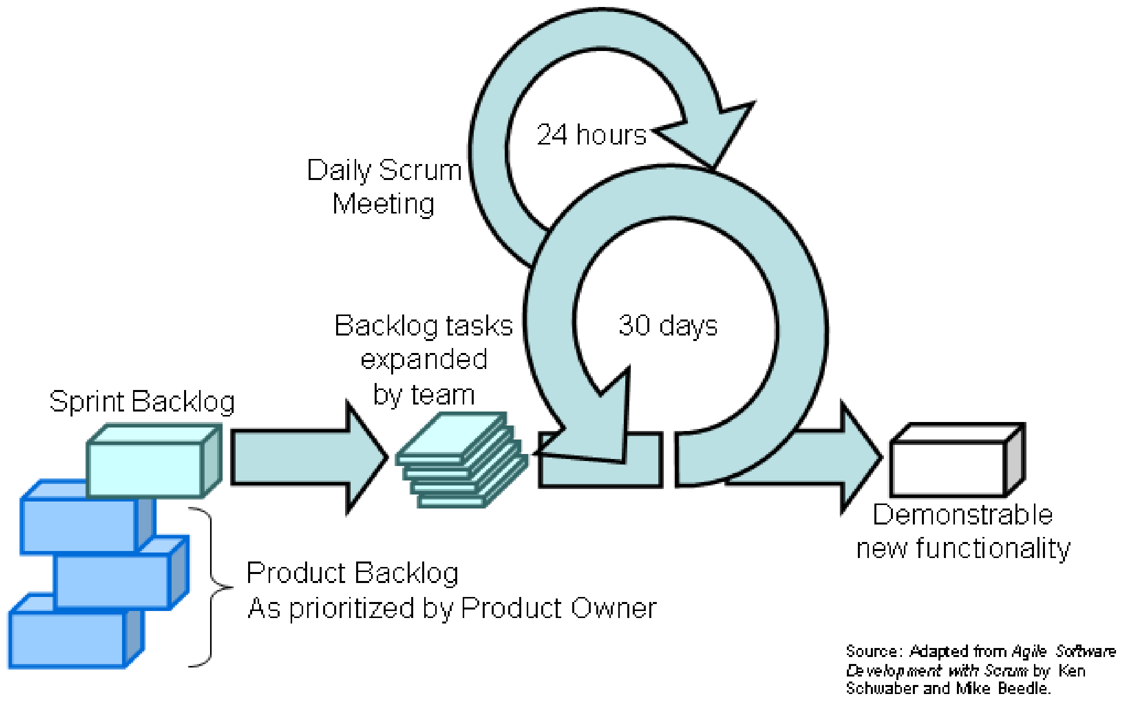

24 hours
Daily Scrum Meeting
30 days
Backlog tasks expanded by team
Sprint Backlog
Product Backlog As prioritized by Product Owner
Demonstrable new functionality
Source: Adapted from Agile Software Development with Scrum by Ken Schwaber and Mike Beedle.

Figure 3. 1 Overview of Scrum Methodology

3.2 **Scrum Artifacts.**

Since Scrum is an agile method, it follows that the formality of the project is as low as possible. This gave me the freedom to make change requests as often. However, it is considered important that the customer can see the project progress since this improves their motivation and involvement. Also, the team needs some formality to help them cooperate and focus their work.

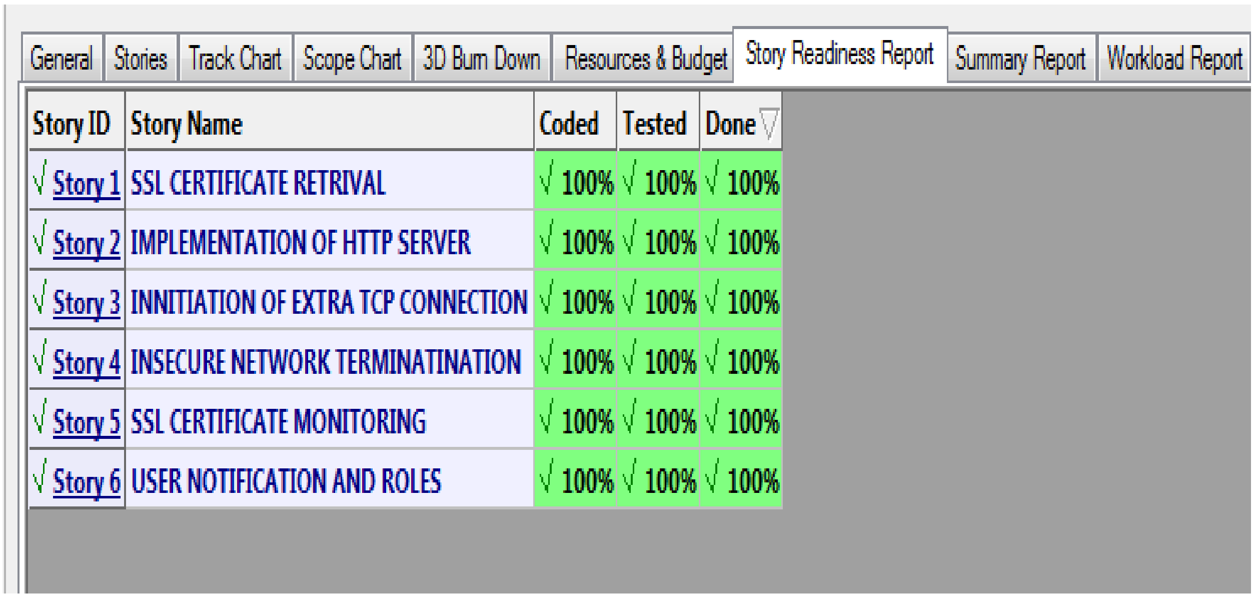
The artifacts of Scrum are

1. the product backlog
2. the project burndown chart
3. the sprint backlog
4. the sprint burndown chart
5. the impediments list.

3.2.1 **The Product Backlog.**

This could be considered equivalent to the requirements specifications but there is one big difference. Instead of a long description of each requirement, the product backlog only has a single sentence description of each requirement. This sentence should be enough to remind the customer and the developers of what the feature is.

The Product Backlog is a list of such single sentence requirements. It is the customer's responsibility to keep it prioritized and updated. The customer adds requirements to this list and then the team is responsible for estimating how long it will take to implement them.



Story ID	Story Name	Coded	Tested	Done
√ Story 1	SSL CERTIFICATE RETRIVAL	√ 100%	√ 100%	√ 100%
√ Story 2	IMPLEMENTATION OF HTTP SERVER	√ 100%	√ 100%	√ 100%
√ Story 3	INNITIATION OF EXTRA TCP CONNECTION	√ 100%	√ 100%	√ 100%
√ Story 4	INSECURE NETWORK TERMINATINATION	√ 100%	√ 100%	√ 100%
√ Story 5	SSL CERTIFICATE MONITORING	√ 100%	√ 100%	√ 100%
√ Story 6	USER NOTIFICATION AND ROLES	√ 100%	√ 100%	√ 100%

Figure 3. 2 product backlog from story readiness report

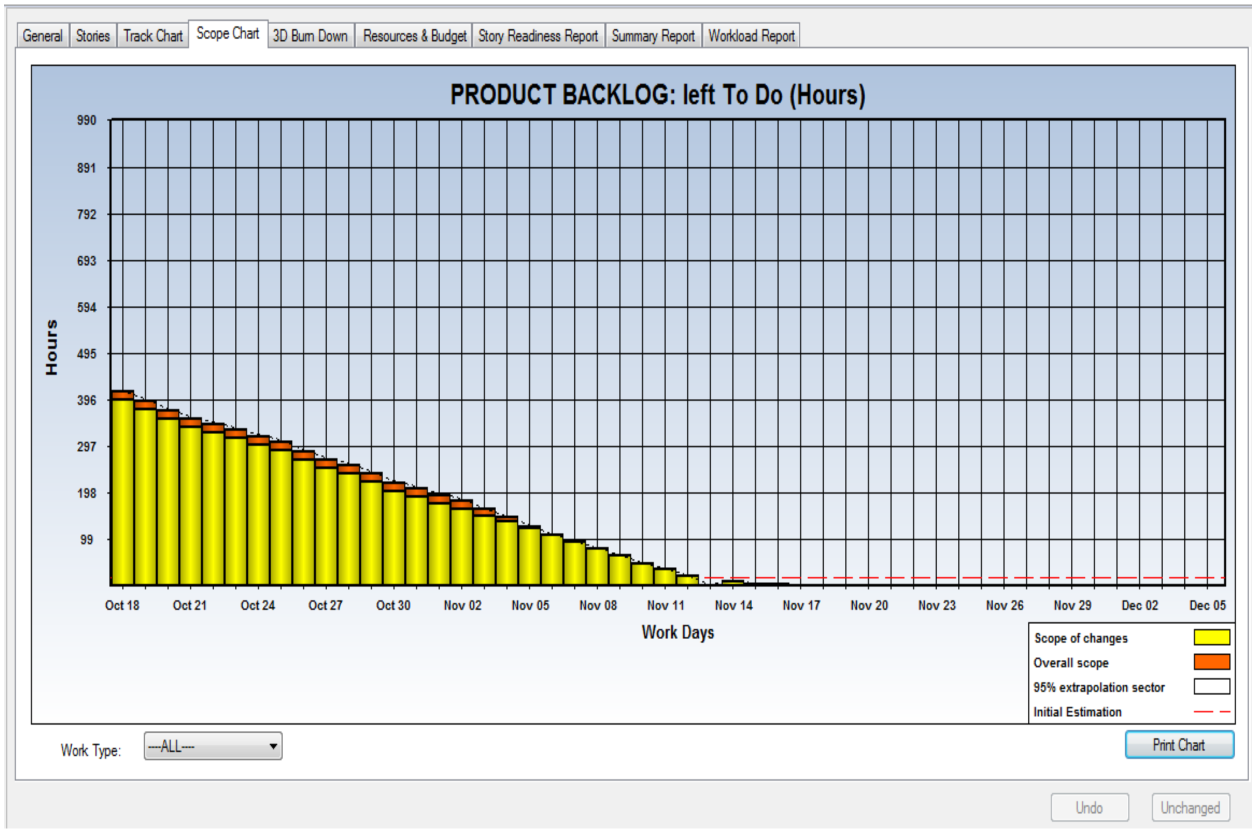


Figure 3. 3 Product backlog burndown chart

The product backlog burn down chart depicts the development process of the product backlog item throughout the development process.

3.2.2 The Project Burn down Chart.

This is a graph with the work remaining on the Product Backlog as the y axis and the time elapsed since project startup as the x axis. The graph gives a visual representation of the project speed. It can also be (and usually is) used to see when the project will be completed at the current development speed.

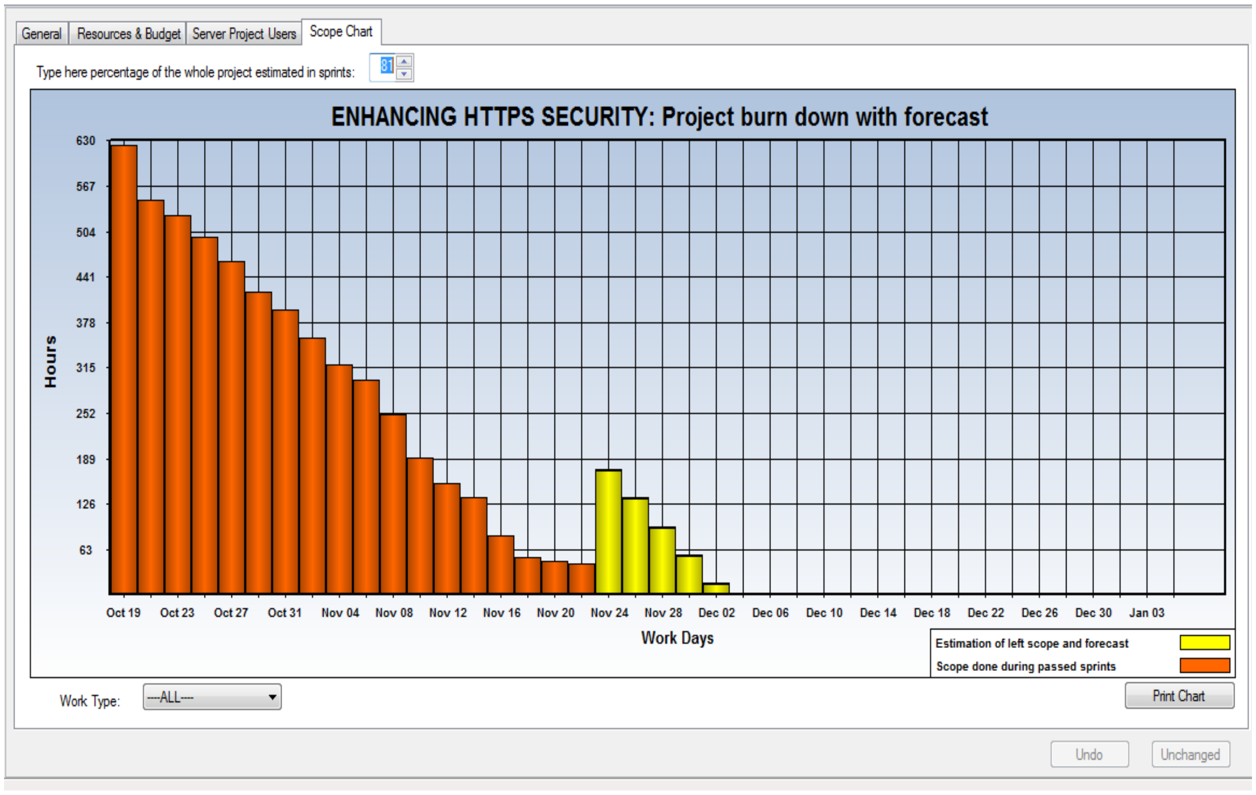
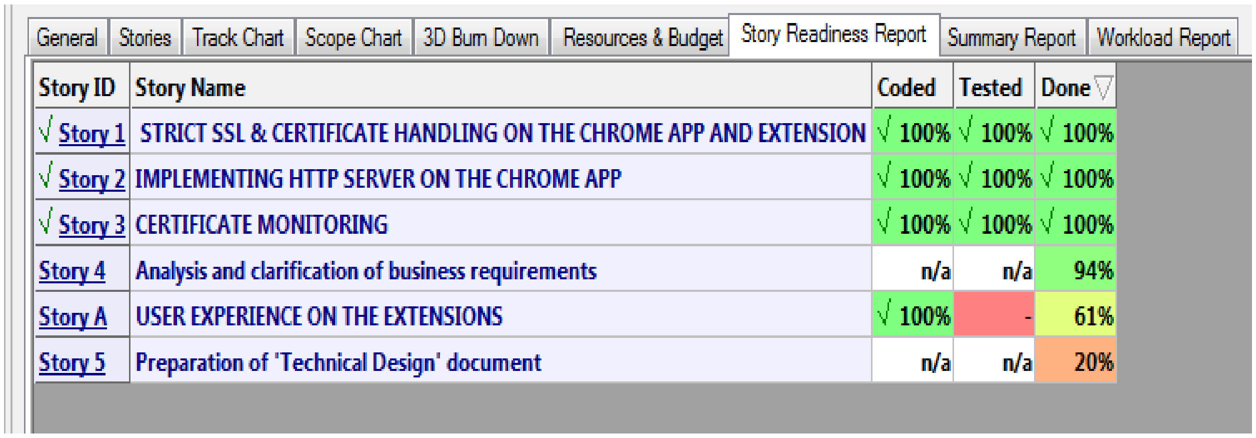


Figure 3. 4 project burndown chart with forecast

3.2.3 The Sprint Backlog.

This is a list of tasks maintained and compiled by the team based on the items from the product backlog that were selected to be part of the sprint. The list is similar to the product backlog, but there is a big difference. Where the items on the product backlog are features requested by the user, the sprint backlog is a list of tasks the developers must do to implement the items that the customer chose from the product backlog. The customer doesn't need to know about the items on the sprint backlog.

A general rule for the tasks on the sprint backlog is that they should be relatively short, i.e. between one hour and two days. This makes it easier to estimate the tasks, something that makes the sprint burn down chart (presented below) more accurate.



Story ID	Story Name	Coded	Tested	Done
√ Story 1	STRICT SSL & CERTIFICATE HANDLING ON THE CHROME APP AND EXTENSION	√ 100%	√ 100%	√ 100%
√ Story 2	IMPLEMENTING HTTP SERVER ON THE CHROME APP	√ 100%	√ 100%	√ 100%
√ Story 3	CERTIFICATE MONITORING	√ 100%	√ 100%	√ 100%
Story 4	Analysis and clarification of business requirements	n/a	n/a	94%
Story A	USER EXPERIENCE ON THE EXTENSIONS	√ 100%	-	61%
Story 5	Preparation of 'Technical Design' document	n/a	n/a	20%

Figure 3. 5 Sprint backlog From the Story readiness report

3.2.4 The Sprint Burn Down Chart

This is quite similar to the Project Burn down Chart, only that it measures the progress of the sprint instead of the project. However, the sprint burn down chart differs from the product burn down chart because the team usually discovers tasks they did not consider but that must be added to the sprint backlog. Since the chart displays the amount of work remaining and not the amount of work completed, the graph can in fact increase from one day to the next.

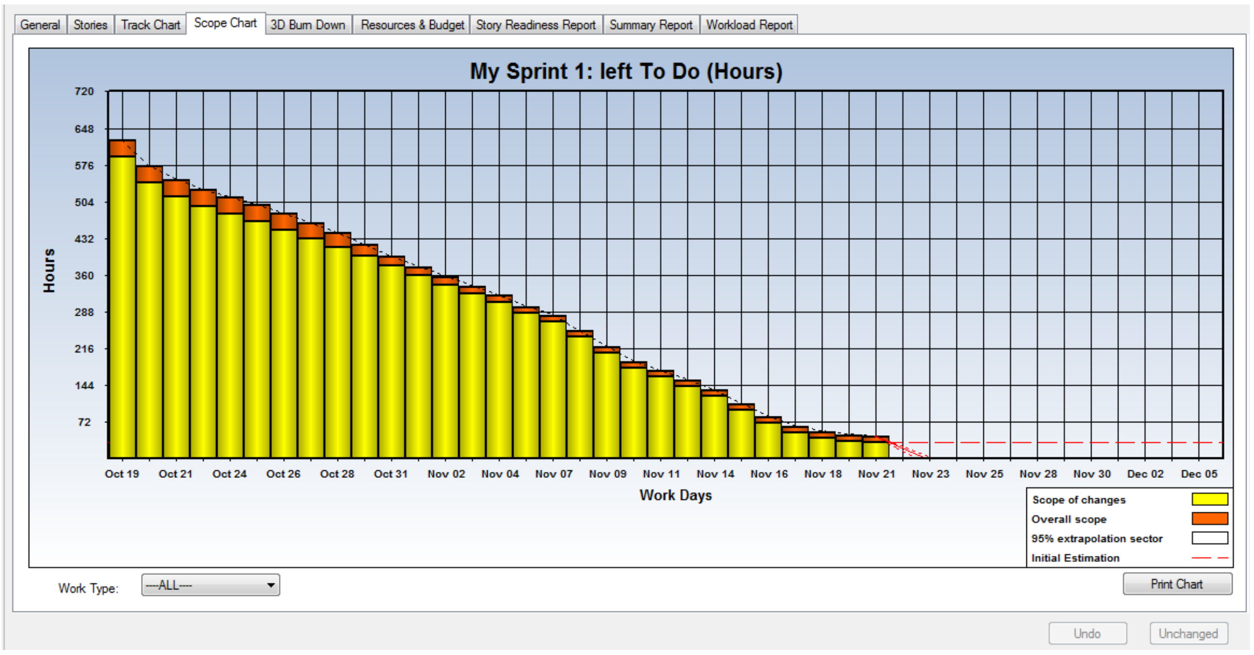


Figure 3. 6 Sprint backlog burn down chart

3.2.5 Impediment list.

An impediment is something that is holding back development in some way or another. The scrum master's responsibility to deal with any such impediments. This list is simply a set of tasks that the scrum master uses to track the impediments that needs to be solved.

Impendent list is listed in the discussion section of this project report

3.3 Sprint

All work is done in sprints lasting four weeks. Each sprint is started with a planning meeting divided in two sessions of at most 4 hours each. How the team works during the sprint is not specified, however, Schwaber has written that XP compliments Scrum nicely (Schwaber, 2002). XP covers engineering practices but doesn't go into detail on management practices and Scrum doesn't cover engineering practices but is quite clear on management practices. The way I see it, Scrum can be considered a replacement of the planning game in XP (Extreme Programming).

3.4 The Sprint Planning Meeting.

In the first session, the Customer chooses high priority items from the product backlog that should be completed in the upcoming Sprint. The customer explains the items to the team and they give an estimate on how long it will take to complete it. The sprint backlog is filled so that the sum of the item estimates is about the same as the available work time of the team during the upcoming sprint.

3.5 The Daily Activities.

During the sprint, the developers work on the items in the sprint backlog. Every day the developers synchronize their progress in a daily Scrum meeting that should last no longer than 15 minutes. During the meeting, all the developers will tell the others what they did since the last Scrum, if there are any impediments obstructing their work and what they are planning on doing until the next Scrum. Another important day to day activity is updating the sprint backlog and burndown chart.

3.6 Sprint Review Meeting

At the end of the sprint, the team meets with the customer and presents the result of the sprint. The users demonstrate the functionality they have completed and gets feedback from the customer. If the demonstrated functionality is what the customer wanted, then this gives the team a feeling of accomplishment as well as the customer a proof that the project is moving in the right direction. If the demonstrated functionality isn't quite what the customer was looking for it is now easy to explain how it is different and what should be done next. In some cases, it is enough to make a few changes while in other cases the implemented functionality must be discarded.

3.7 Sprint Retrospect Meeting

The intention of this meeting is to help the team improve their development process. The meeting is attended by the team, the scrum master and the customer (optional). During the meeting the team members take turns saying what went well during the last sprint, and what could be improved. After all team members, have had their say, they prioritize the possible improvements and discuss them in order. The meeting should not last more than 3 hours.

3.8 Project Startup

Ken Schwaber has had much success with his kick-starting of Scrum projects as described in the book Agile Project Management with Scrum (Schwaber, 2002). This process goes as follows.

The Scrum Master works with the customer and prepares a backlog. Then the Scrum Master, the Customer and the Team uses one day to go over this backlog. During this first day, the customer explains the items in the backlog to the team, and the team estimates how much work it would take to implement this. The customer then prioritizes the items in the backlog and divides the backlog items into sprints. The following day is the first day of the first sprint. This first sprint isn't very different from the following sprints, except that the first part of the sprint planning meeting has already been completed.

The team is now in complete control and have one task, namely to deliver the functionality the customer has requested. The sprint has begun.

3.9 Project Completion

As the project moves on and sprints are being completed, the customer will receive increments of the product. If the customer realizes that the product is good enough and that further development is unnecessary, then he should be able to stop the project. Depending on the contract that has been negotiated, there can be a penalty fee for premature termination of the project.

CHAPTER 4
SOFTWARE DEVELOPMENT.

Software Analysis and Requirement

This is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed.

4.1 Study of the current system.

We studied two defects in the current system that may allow the compromise of secure encrypted connections and the integrity of the SSL certificate used.

4.1.1 The Trouble with HTTP Strict Transport Security (HSTS)

Almost a year and a half after the HTTP Strict Transport Security (HSTS) mechanism was established as a standard, its adoption rate by websites remains low because developers are not aware of its benefits and Internet Explorer still doesn't support it, according to advocacy group the Electronic Frontier Foundation. This low rate of low adoption has been influenced by HSTS opting in security which can really devastate the availability of the website.

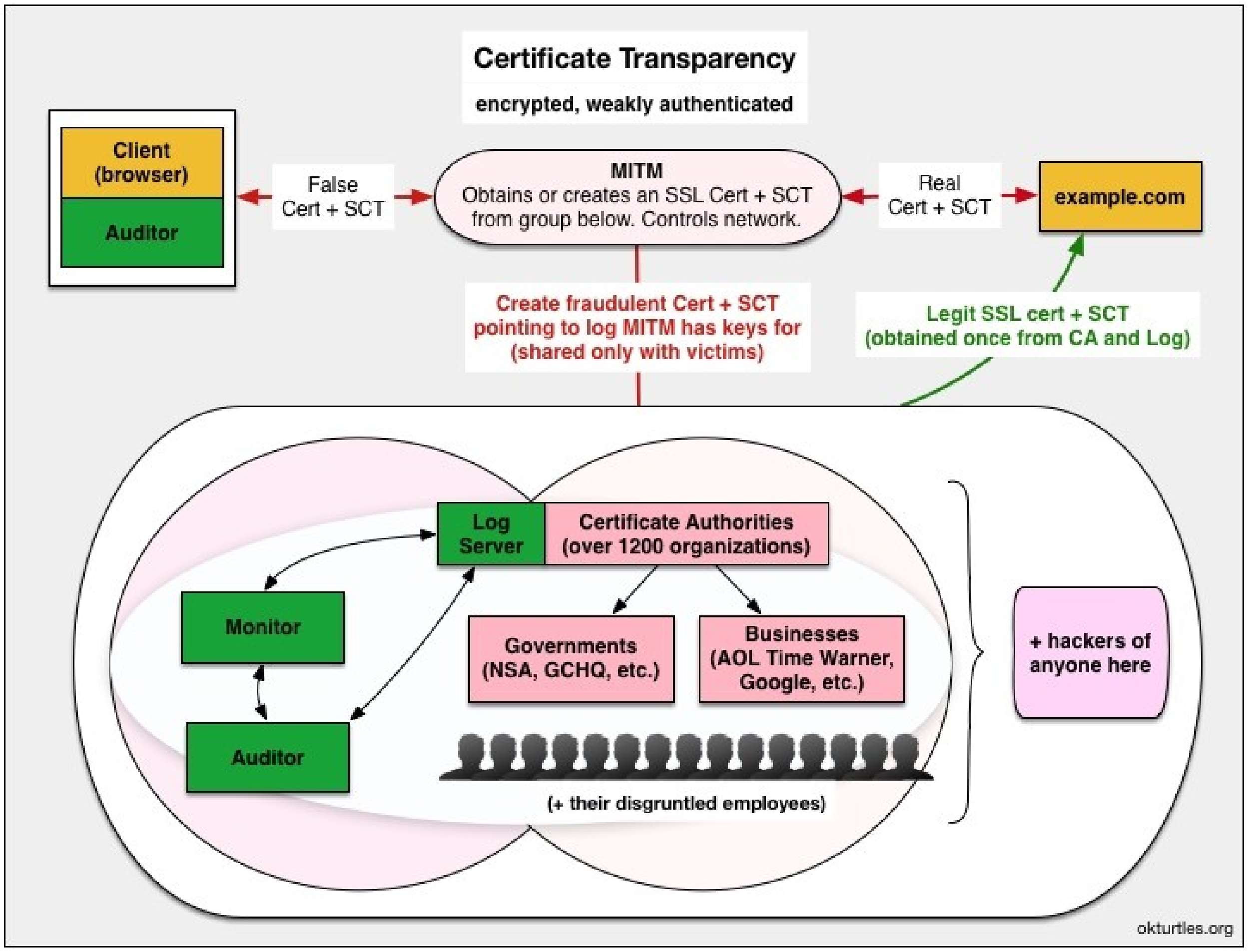
However, the support for HSTS in browsers has been incomplete, which likely discouraged websites from enabling the mechanism. "Only Chrome, Firefox, and Opera have had HSTS support for a significant period," the EFF technologist said (Constantin, 2014). "This is changing though: we noticed that Apple quietly added HSTS support to Safari in OS X 10.9. For now, Internet Explorer doesn't support HSTS -- which means that there's basically no such thing as a secure website in IE."

One problem with HSTS is that it assumes the first ever connection from a browser to a HTTPS website is achieved securely, without a man-in-the-middle attacker interfering and removing the HSTS policy header. In order to partially mitigate this problem Google Chrome and Mozilla Firefox contain pre-loaded lists of HSTS sites. This has been in practice difficult to implement for example, get actual sites supporting HSTS then add the site to a list on the client browser (Constantin, 2014). This was also a design flaw if the first connection is compromised and the preceding connections are compromised too how can HSTS prove to be useful?

HSTS does not persist browser restart it needs to be reconfigured again.

4.1.2 The Trouble with Certificate Transparency.

With the current certification authority mechanisms, it does not cater for man-in-the middle-attacks that leverage themselves in the network. Also, the log servers that contains the certificate bundle verification is kept away from the client browser access. This design flaw has enable the exploitation of encrypted connection by using rogue certificates (Slepak & Greg, 2014).


Certificate Transparency
encrypted, weakly authenticated
Client (browser)
Auditor
False Cert + SCT
MITM
Obtains or creates an SSL Cert + SCT from group below. Controls network.
Real Cert + SCT
example.com
Create fraudulent Cert + SCT pointing to log MITM has keys for (shared only with victims)
Legit SSL cert + SCT (obtained once from CA and Log)
Log Server
Certificate Authorities (over 1200 organizations)
Monitor
Auditor
Governments (NSA, GCHQ, etc.)
Businesses (AOL Time Warner, Google, etc.)
(+ their disgruntled employees)
+ hackers of anyone here
okturtles.org

(Greg, 2014)

4.1.3 Requirements of the new system.

The new system must be in a position to Enforce HTTPS connections and validate the integrity if the SSL/TLS certificate used. In Enhancing HTTPS connections, it should ensure the ability to detect a first compromised connection and should persist browser restarts. In validating the SSL/TLS certificate the client browser should have a mechanism to authenticate it to an updated open certificate transparency platform.

Benefits of Certificate Transparency.

i. Certificate Transparency leads to a situation where "It becomes impossible to misuse a certificate without detection"

ii. Certificate Transparency is a "Generally applicable" system where "No one is special" and where everyone "[is] able to participate".

iii. Certificate Transparency doesn't introduce trusted third-parties.

iv. Certificate Transparency doesn't push decisions onto the end user.

v. That DNSChain wastes energy and "has no mechanism for verification".

4.1.4 Functional System Requirements

The following are the system requirements for enhancing HTTPS security against sophisticated man-in-the-middle attacks.

i. Mandatory enforce HTTPS connections on all trackers.

ii. Enforce HTTPS connections on depreciated HTTP connections

iii. Allow the user to create a whitelist of trusted websites that do not support HTTPS connections

iv. Reveal the status and Integrity of the SSL/TLS certificate on all trackers and the requested urls.

v. Inform the user on a change in SSL certificate and inform the user.

vi. Allow the user to gain support from a browsing Secure community

4.1.5 Non-Functional Requirements

Requirement	Description
Usability	The interface should use terms and concepts, which are drawn from the experience of the people who will make most of the system. For example, the language should be non-technical.
Efficiency	The system must provide easy and fast access without consuming more resources
Reliability	User should never be surprised by the behavior of the system and it should also provide meaningful feedback when errors occur so that user can recover from the errors.
privacy	User data shall not be leaked for whatsoever reason
App security	The apps should be resilience to attacks

4.1.6 Feasibility Study.

The aim of the feasibility study activity is to determine whether it would be financially and technically feasible to develop the system or not. A feasibility study is carried out from following different aspects:

Feasibility Study	Description
Operational Feasibility:	The apps and extensions have been developed for any user who wants to use them. We have given a demo of our project and the users found the system friendly and easy to use. The interoperability with the existing system is also checked after integrating them to google chrome browser.
Technical Feasibility:	It determines if the system can be implemented using the current technology. This has been developed on purely JavaScript, JavaScript libraries and JSON. Which play a functional communication role with chrome browser application programming interfaces (APIs)

Economic Feasibility	The development may be expensive due to labor and development tools but the product has high return that dwarfs the development expense
Implementation Feasibility	This project can easily be made available online without much consideration of the hardware and software. The only required thing at the applicant's side is the Internet connection and Chrome web browser, which are a no difficult issue these days. After setting up the project online, even the development team can access the apps from anywhere.

4.1.7 Functions of the system

Intuitively, use cases represent the different ways in which the users can use a system.

4.1.7.1 Strict SSL use case diagram.

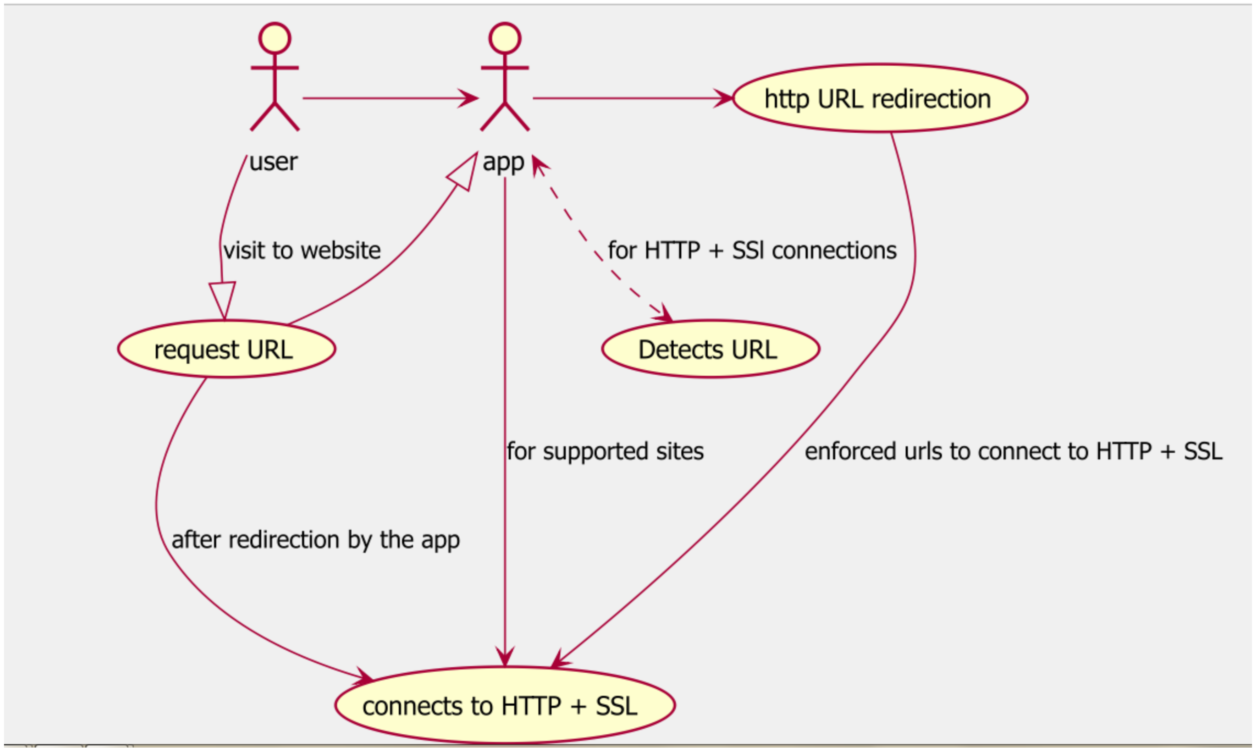


Figure 4. 1 Strict SSL use case Diagram

4.1.7.2 Cert Monitor use case diagram

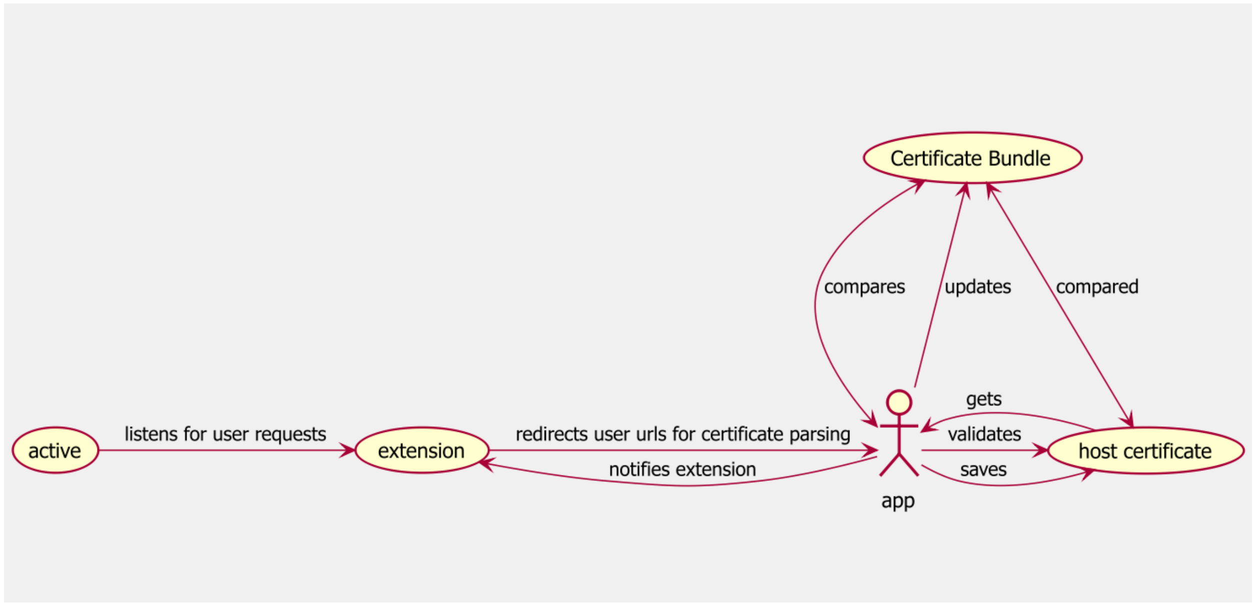


Figure 4. 2 Certificate Monitor Use Case Diagram

4.1.7.3 Cert Monitor Sequence Diagram

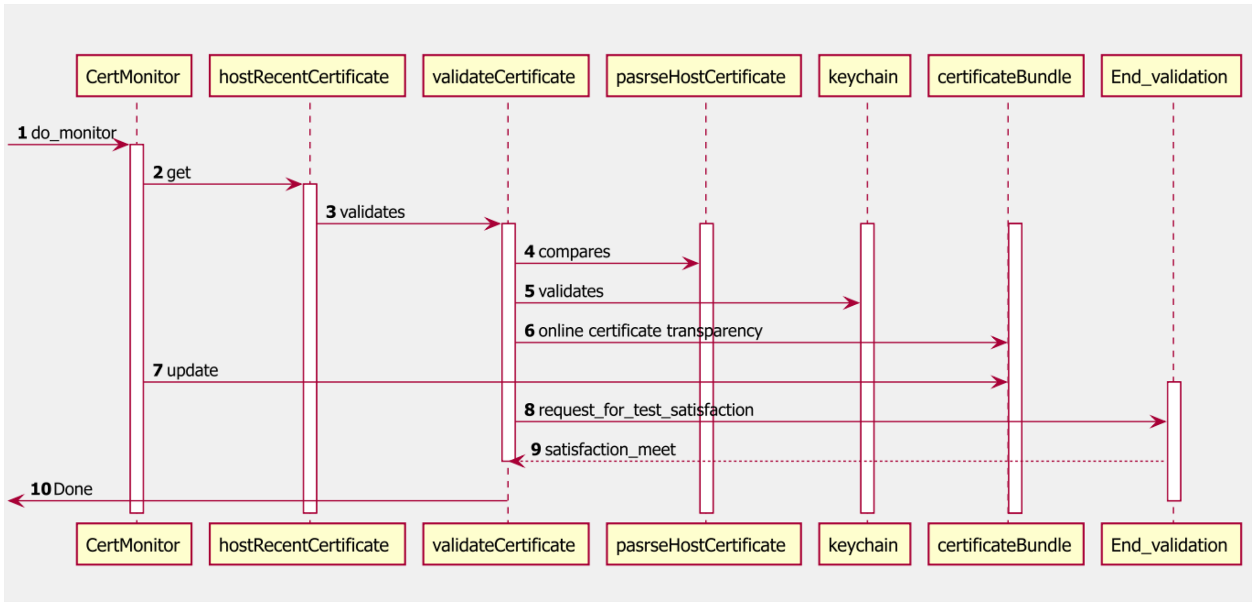


Figure 4. 3 Certificate Monitor Sequence Diagram

4.1.8 Hardware and Operating system considerations.

The application and browser extensions have been developed for google chrome browser. Therefore, the hard Your computer must meet the minimum system requirements before you can install and use Chrome

It's possible that Chrome may install on other platforms or versions not listed here, however Google enterprise level support is limited to systems that meet the minimum requirements. Google does not provide support if you install Chrome on any system that does not meet the specified criteria.

Windows

To use Chrome on Windows, you'll need:

i. Windows 7, Windows 8, Windows 8.1, Windows 10 or later
ii. An Intel Pentium 4 processor or later that's SSE2 capable

Mac

To use Chrome on Mac, you'll need:

i. OS X Mavericks 10.9 or later

Linux

To use Chrome on Linux, you'll need:

i. 64-bit Ubuntu 14.04+, Debian 8+, openSUSE 13.1+, or Fedora Linux 21+
ii. An Intel Pentium 4 processor or later that's SSE2 capableware requirement are of those that can run optimally of google chrome browser.

4.1.9 Programming languages, libraries and chrome APIs used

The development of the project "Enhancing HTTPS Security Against Sophisticated MITM Attacks" is composed of the following components:

Programming languages.

Programming language	**Why this programming language**
JavaScript	In computer science, a programming language is said to support first-class functions if it treats functions as first-class objects. Specifically, this means that the language supports constructing new functions during the execution of a program, storing them in data structures, passing them as arguments to other functions, and returning them as the values of other functions. JavaScript functions are First Class functions
JSON	JavaScript Object Notation is a lightweight data-interchange format. I used it to create the manifest.js of my chrome app and extensions

Table 4. 1 Programming Languages Used

Chrome APIs used

Chrome API used	**Description**
chrome.webRequest	I used the chrome.webRequest API to observe and analyze traffic and to intercept, block, or modify requests in-flight.
chrome.extension	The chrome.extension API has utilities that can be used by any extension page. It includes support for exchanging messages between an extension and its content scripts or between extensions, as described in detail in Message Passing.
chrome.storage	I used the chrome.storage API to store, retrieve, and track changes to user data.
chrome.sockets	I used the chrome.socket API to send and receive data over the network using TCP and UDP connections
chrome.runtime	I used the chrome.runtime API to retrieve the background page, return details about the manifest, and listen for and respond to events in the app

	or extension lifecycle. You can also use this API to convert the relative path of URLs to fully-qualified URLs.

Table 4. 2 Chrome APIs Used

Libraries used.

JavaScript library used	**Description**
Forge.min.js	The Forge software is a fully native implementation of the TLS protocol in JavaScript as well as a set of tools for developing Web Apps that utilize many network resources.
Moment.js	Parse, validate, manipulate, and display dates in JavaScript.
URI.js	URI.js is a JavaScript library for working with URLs.
Underscore-min.js	Underscore provides over 100 functions that support both your favorite workaday functional helpers: map, filter, invoke — as well as more specialized goodies: function binding, javascript templating, creating quick indexes, deep equality testing, and so on.
Restruct.js	restruct.js performs conversion to and from binary data types. It utilizes an intuitive declarative API to define formats for binary structure parsers and emitters. It works in both the browser and on Node.

Table 4. 3 Libraries Used

Software Design

4.2 Introduction

During analysis, the focus is on what needs to be done intendment of how it is done. During design, decisions are made about how the problem will be solved, first at a high level, then at increasingly detailed levels.

System design is the first stage in which the basic approach to solving the problem is selected. During system designing the overall structure and style are decided. The system architecture is the overall organization of the system into components called system. System design deals with transforming the customer requirements, as described in the product backlog, into a form that is implement able using the programming language.

As a system designer, we are tried to take following design decisions:

i. Organize the system into an application and two extensions.
ii. Organize sub-modules for each extension.
iii. Allocate tasks to the application.
iv. Choose an approach to manage data store.
v. Handle access to global resources.
vi. Choose an implementation logic.

4.2.1 Designs considerations during problem solving.

With the problem at hand which is to enforce encryption for websites that support HTTPS as much as currently possible in Chrome browser and enable client browser to connect to an encrypted connection and display certificate integrity information to the client, we meet these two requirements with the following designs models.

i. We had to get control of the client browser requests.
ii. We had to make our application resilience to any attacks. Moreover, this is safe guarded by the chrome app security model.

4.2.2 Rationale for choosing the design models.

Adversaries when performing man in the middle attacks leverage themselves on the network. This is due the improved security on the server and client browser with modern technology. However, adversaries may change how data is exchanged over the network bleeding out lots of exploits. SSL

stripping which connects the client browser to unencrypted connection and SSL/TLS sniffing attacks which utilizes forged certificates can be defeated in design.
Defeating SSL Stripping in design is to redirect any requested url by the client browser to an application that connect only in HTTPS and closing out all requests from the client browser. Redirection that happens immediately after request defeats any SSL Stripping in the network due to a change in scope of the attack tool logic and capability of the network. For example, if an SSL Stripping tool is placed on the network to listen to browser client request may listen infinity due to immediate redirection. In my Strict SSL extension and Cert Monitor app I have redirected request to localhost. Localhost redirection are far out of scope on any threat design models that leverages itself on the network since this has a loopback function to the client browser. This will not leverage the attacks on the attack tool on the network.

4.2.3 Chrome Browser App Architecture.

Chrome Apps integrate intimately with a client's operating system. They are intended to be running outside of a browser tab, to run robustly in offline and poor network situations and to have significantly more capable capacities that are available in a typical web browsing environment. The app container, programming, and security models support these Chrome App requirements.

Requirement 1. App Container Model.

The app container describes the visual appearance and loading behavior of Chrome Apps. Chrome Apps look different than traditional web apps because the app container does not show any traditional web page UI controls; it simply contains a blank rectangular area. This allows an app to blend with “native” apps on the system, and it prevents the user from “messing” with the app logic by manually changing the URL.
Chrome Apps are loaded differently than web apps. Both load the same type of content: HTML documents with CSS and JavaScript; however, a Chrome App is loaded in the app container, not in the browser tab. Also, the app container must load the main document of the Chrome App from a local source. This forces all Chrome Apps to be at least minimally functional when offline and it provides a place to enforce stricter security measures.

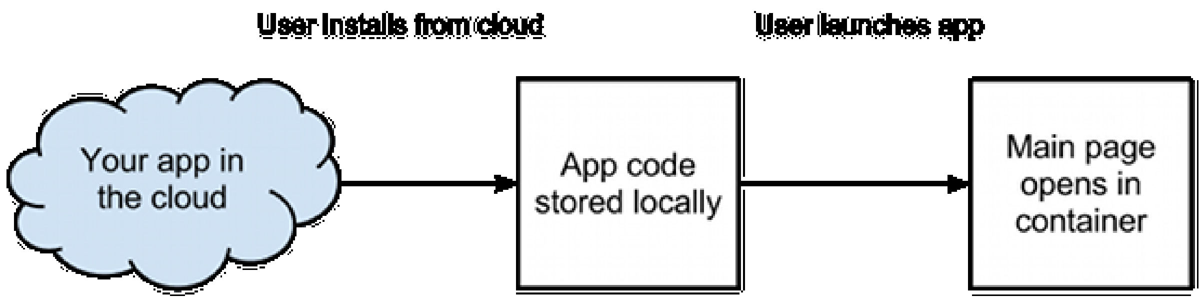


Figure 4. 4 App container Model

Requirement 2. Programming model

The programming model describes the lifecycle and window behavior of Chrome Apps. Similar to native apps, the goal of this programming model is to give users and their systems full control over the app lifecycle. The Chrome App lifecycle should be independent of browser window behavior or a network connection.

The "event page" manages the Chrome App lifecycle by responding to user gestures and system events. This page is invisible, only exists in the background, and can be closed automatically by the system runtime. It controls how windows open and close and when the app is started or terminated. There can only be one "event page" for a Chrome App.

Requirement 3. App lifecycle

App lifecycle defines the app runtime from initialization to termination. However, chrome app lifecycle is defined by the app runtime and event page. These are responsible for managing the app lifecycle. The app runtime manages app installation, controls the event page, and can shut down the app at any time. The event page listens out for events from the app runtime and manages what gets launched and how.

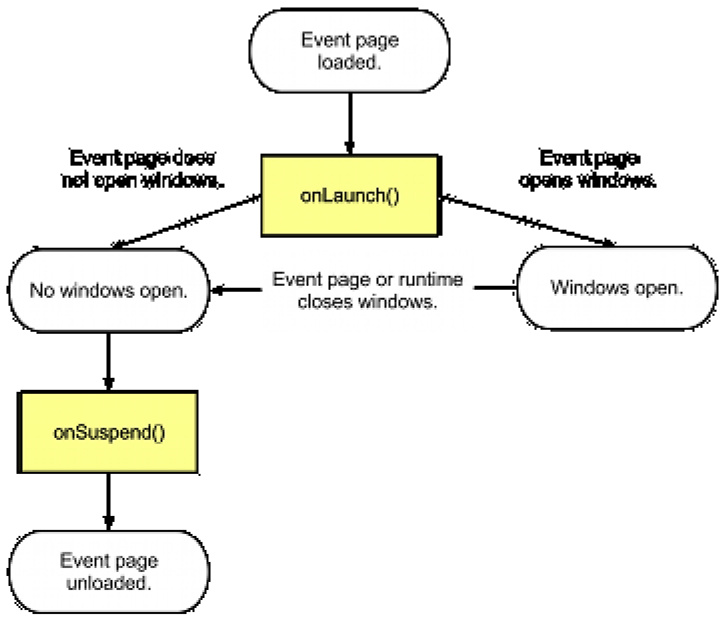

Event page loaded.
onLaunch()
Event page does not open windows.
Event page opens windows.
No windows open.
Event page or runtime closes windows.
Windows open.
onSuspend()
Event page unloaded.

Figure 4. 5 App life cycle

Stage	**Summary**
Installation	User chooses to install the app and explicitly accepts the permissions.
Startup	The event page is loaded, the 'launch' event fires, and app pages open in windows. You create the windows that your app requires, how they look, and how they communicate with the event page and with other windows.
Termination	User can terminate apps at any time and app can be quickly restored to previous state. Stashing data protects against data loss.
Update	Apps can be updated at any time; however, the code that a Chrome App is running cannot change during a startup/termination cycle.
Uninstallation	User can actively uninstall apps. When uninstalled, no executing code or private data is left behind.

Table 4. 4 App Lifecycle Table

Requirement 4 Security model.

The Chrome Apps security model protects users by ensuring their information is managed in a safe and secure manner. Comply with Content Security Policy (Google, 2016) includes detailed information on how to comply with content security policy. This policy blocks dangerous scripting reducing cross-site scripting bugs and protecting users against man-in-the-middle attacks.

Loading the Chrome App main page locally provides a place to enforce stricter security than the web. Like Chrome extensions, users must explicitly agree to trust the Chrome App on install; they grant the app permission to access and use their data. Each API that your app uses will have its own permission. The Chrome Apps security model also provides the ability to set up privilege separation on a per window basis. This allows you to minimize the code in your app that has access to dangerous APIs, while still getting to use them.

Chrome Apps reuse Chrome extension process isolation, and take this a step further by isolating storage and external content. Each app has its own private storage area and can't access the storage of another app or personal data (such as cookies) for websites that you use in your browser. All external processes are isolated from the app. Since iframes run in the same process as the surrounding page, they can only be used to load other app pages. You can use the object tag to embed external content; this content runs in a separate process from the app.

4.2.4 Strict SSL Design.

Strict SSL in design and in practice achieves to automatically connect a site to encrypted HTTPS connection and enforces all subsequent request to be over SSL. As soon as the domain is set to enforce, the browser will not send any unencrypted requests even if the application is paused the only way to connect to unencrypted connection is by whitelisting the site or the site no longer supports HTTPS connections. In design and practice the Strict SSL will cache all site which support SSL this reduces overhead and quick connection. Strict SSL in set to respect incognito mode of the chrome browser that states no caching of user data including caches for whatsoever reason.

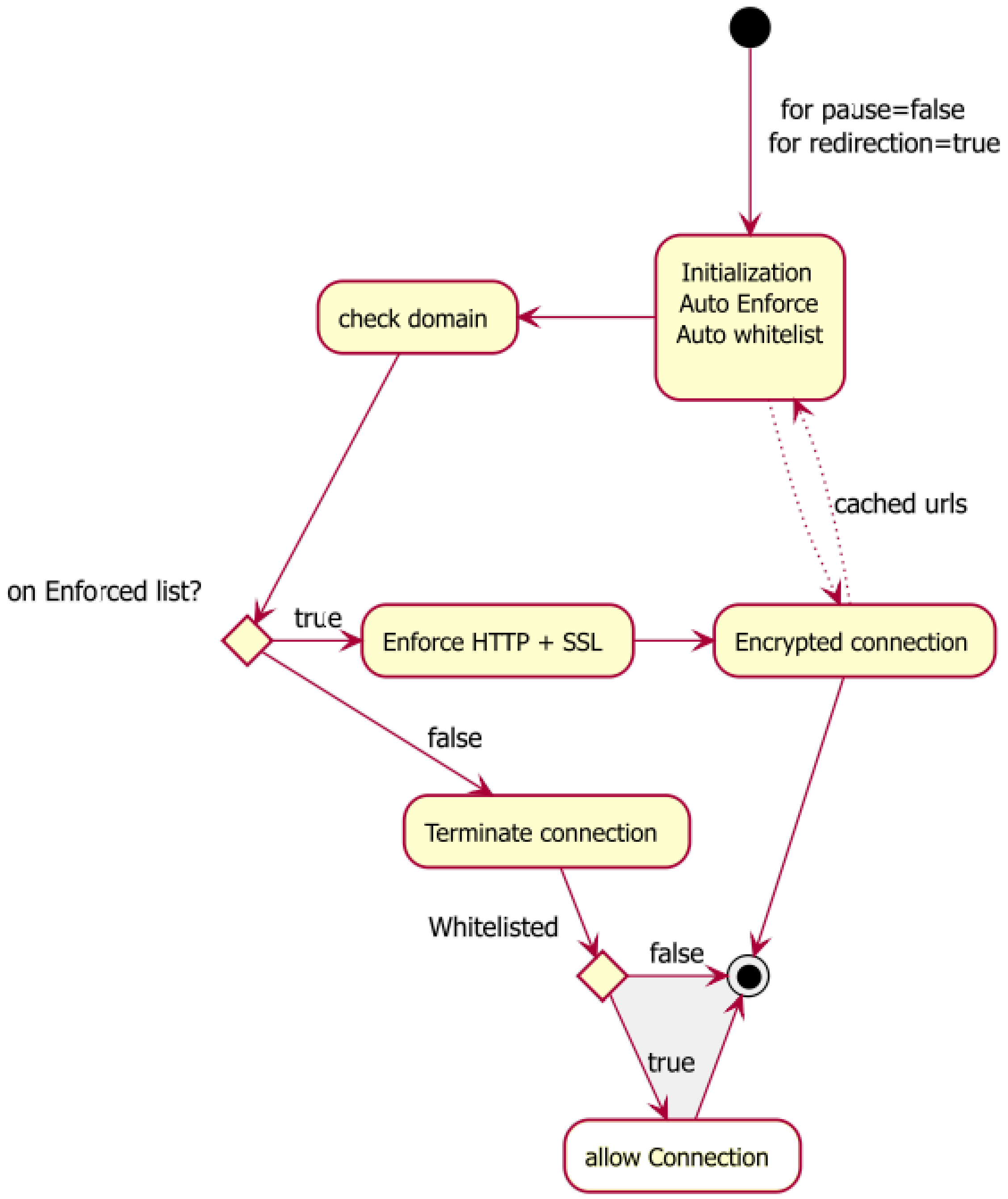


Figure 4. 6 Strict SSL Design

4.2.5 Certificate Monitor Design.

Cert Monitor in design and in practice sets to achieve validate the integrity of the SSL/TLS certificate.

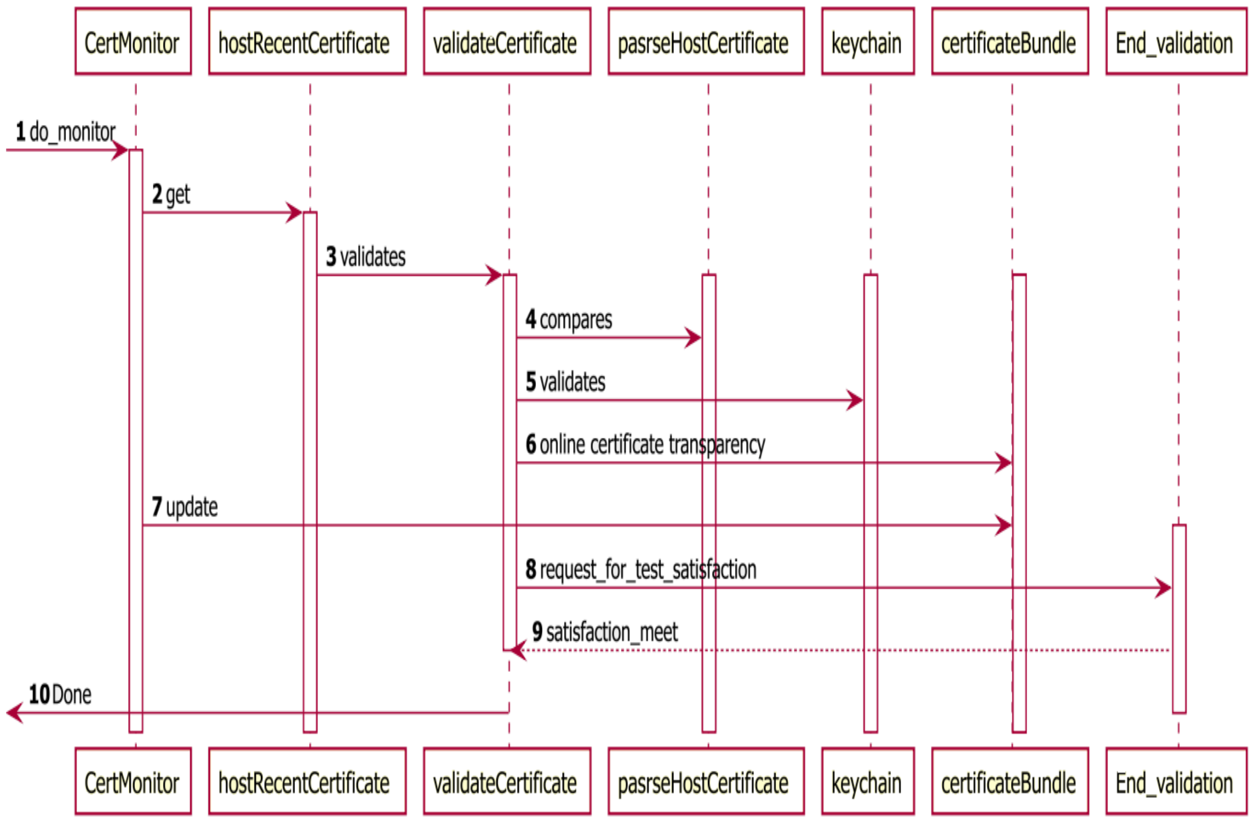


Figure 4. 7 Certificate Monitor Design

Software Implementation.

4.3 Software Implementation Environment.

The implementation view of software requirement presents the real-world manifestation of processing functions and information structures. This computerized system is specified in a manner that dictates accommodation of certain implementation details.

The implementation environment of the developed app and extensions facilitates multiple tabs to use this system simultaneously. The user interfaces are designed keeping in mind that the users of this system are familiar to using GUI-based systems. Thus, we restricted ourselves to developing a GUI-based system so that it becomes easier for the end user to get acquainted to the developed app and extensions.

4.3.1 4-Tier Implementation architecture

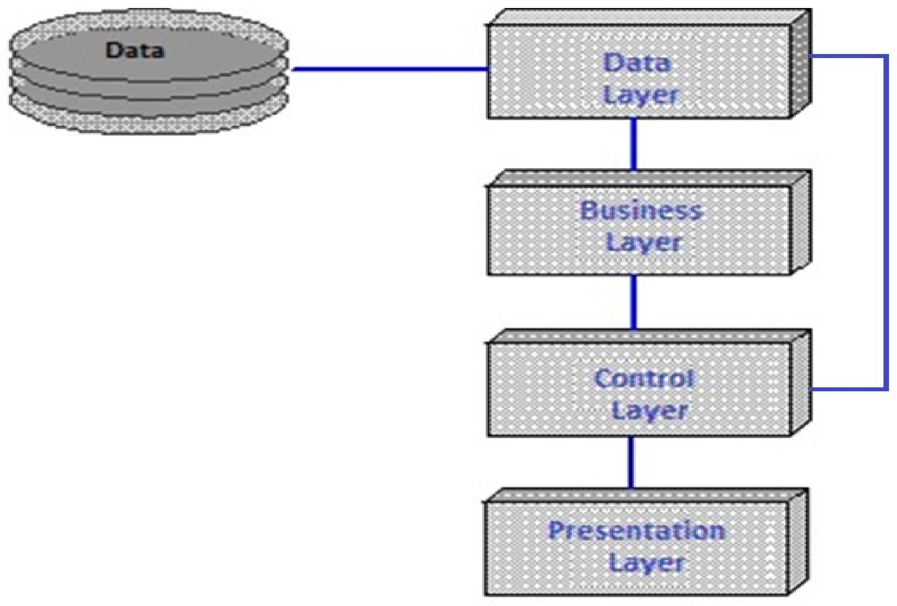


Figure 4. 8 4-Tier Implementation Architecture

We have also followed the web based 4-tier architecture as the implementation architecture which is as follows:

Layer 1. The presentation tier or user services layer

This layer gives a user access to the application. It contains all the web page so it is this interface through which user can access the application. This layer presents data to the user and optionally permits data manipulation and data entry.

Layer 2. The control tier or control layer.

This layer gives a good separation between code and its connectivity with local storage and chrome APIs. This layer includes the JavaScript code and JavaScript libraries.

Layer 3. The business services layer.

It consists of business and data rules. Also, referred to as the business logic tier, the middle tier is where we as developers can solve mission-critical business problems and achieve major productivity advantages. The components that make up this layer can exist on a host machine, to assist in resource sharing. These components can be used to enforce business rules, such as business algorithms and data rules, which are designed to keep the data structures consistent within either specific or multiple local storages. Because these middle-tier components are not tied to a specific client, they can be used by all applications and can be moved to different locations, as response time and other rules require. In my project the chrome app holds all the acceptance criteria of the product backlog.

Layer 4. The data tier or data services layer.

This layer interacts with persistent data usually stored in a database or in permanent storage. In my case, it interacts with data stored in the client host machine. In this layer, we have implemented the basic function through data can be accessed like insert, update, delete, selection.

Our Requirements are changing dynamically so we used four-tier architecture. The four-tier approach provides benefits such as reusability, flexibility, manageability, maintainability, and scalability.

Software Development Testing.

4.4 Testing Plan.

Software Testing has a dual function; it is used to identify the defects in program and it is used to help judge whether or not program is usable in practice. Thus, software testing is used for validation and verification, which ensure that software conforms to its specification and meets need of the software customer.

We resorted to Alpha testing, which usually comes in after the basic design of the program has been completed. The project supervisor will look over the program and give suggestions and ideas to improve or correct the design. They also report and give ideas to get rid of around any major problems. There is bound to be a number of bugs after a program have been created.

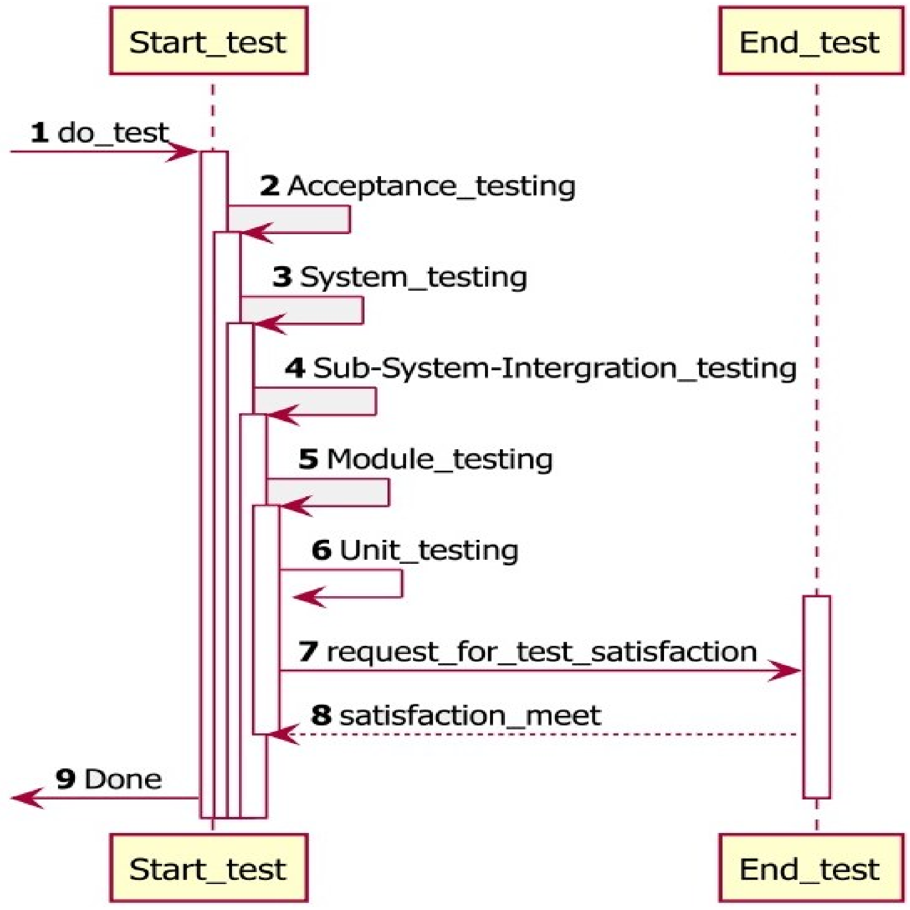


Figure 4. 9 Top-down Test Methodology

Test	**Description**
Unit Testing	Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.
Module Testing	Module testing is the testing of complete code objects as produced by the compiler when built from source
Sub System Integration Testing	Sub-system integration testing focuses on testing the external APIs (Application Programming Interfaces) between sub-systems.
System Testing	The system testing procedure tests for errors resulting from unexpected interactions among sub-systems and system components.
Acceptance Testing	The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.
Request test satisfaction	The purpose of this test is to ensure that the goals of the test were meet
Test satisfaction	This approves the Test procedure and ends Test

Table 4. 5 Top-down Testing Methodology Description

4.4.1 The Testing Process

We test the software process activities such as Design, Implementation and sprint backlog items. Because, design errors are very costly to repair once system has been started to operate, it is quite obvious to repair them at early stage of the system. So, analysis is the most important process of any project.

4.4.2 Requirement Traceability

As most interested portion is whether the system is meeting its requirements or not, for that testing should be planned so that all requirements are individually tested. We checked the output of certain combination of inputs so that we can know whether it gives desirable results or not. Strictly sticking to your requirements specifications, give you the path to get desirable results from the system.

4.4.3 Testing Schedule

We have tested each procedure back-to-back so that errors and omissions can be found as early as possible. Once the system has been developed fully we tested it on other machines, which differs in configuration. We had volunteers from the browsing secure community who literary tested the working of the application. Testing went concurrent with the development.

4.4.4 Test Strategy

There are types of testing that we implement. They are as follows:

1. Project resilience towards compromises. For example, for man-in-the-middle attacks the time span of the attacks are very short and can be well achieved in design. Spend more better designs that defend the user in logic and operation.
2. Decide on the effort required for testing based on the usage of the security solution. If the system is to be used by a large number of users, evaluate the impact on users due to a system failure before deciding on the effort.
3. A necessary part of the test case is a definition of the expected result.
4. Write test cases for invalid and unexpected as well as valid and expected input conditions.
5. Thoroughly inspect the results of each test.
6. We have performed both Unit Testing and System Testing on the applications and extensions to detect and fix errors. A brief description of both is given below.

4.4.5 Unit Testing

Objective

The objective of Unit Testing is to test a unit of code (program or set of programs) using the Unit Test Specifications, after coding is completed. Since the testing will depend on the completeness and correctness of test specifications, it is important to subject these to quality and verification reviews.

Input

1. Unit Test Specifications
2. Code to be tested

Testing Process

1. Checking for availability of Code Walk-through reports which have documented the existence of and conformance to coding standards.
2. Verify the Unit Test Specifications conform to the program specifications.
3. Verify that all boundary and null data conditions are included.
4. Features to be tested

Features to be tested.

Test Specification	**Description**
Functionality Test	All possible scenarios to test the functionality of the component are listed here. This list is made very exhaustive to cover all the expected functionality described in the Software Requirement Specifications and Design document completely.
‘Valid SSL/TLS’ as well as ‘Invalid SSL/TLS’ connections are clearly articulated.	Suitable Error/ Warning Messages
Connection Status on Access to a site	The user should be easily being able to tell the connection type to our app and extension
Privacy Concerns	The applications should not handle user sensitive data

Table 4. 6 Features to be Tested

Unit Test Specifications

A sample Unit Test Specification is as follows.

Form Template Functionality

Event	Action	Expected Result	Observed result	Verified
1.	On pressing enforce button	The enforced urls on the tab should reload to HTTPS	As Expected	YES
2.	On pressing options link	Open the whitelist and enforced lists	As Expected	YES
3.	On pressing ignore	Should allow the requested url to connect normally no enforcing HTTPS connection	As Expected	YES
4.	On pressing redetect	Should switch between HTTP and HTTPS connections	As Expected	YES
5.	On pressing clear button	Stored list of url should be cleared fom the storage list	As Expected	YES
6.	On pressing pause button	Strict SSL should pause its activities and wait for a reload	As Expected	YES

Table 4. 7 Unit Testing for Strict SSL

4.4.6 Integration Testing

After our individual modules were tested out we proceed to the integration testing to create a complete system. This integration process involves building the system and testing the resultant system for problems that arise from component interactions.

We have applied top-down strategy to validate high-level components of a system before design and implementations have been completed. Our development process started with high-level components and we worked down the component hierarchy.

4.4.7 System Testing

System testing is actually a series of tests whose purpose is to fully exercise the computer-based system. It verifies that system elements have been properly integrated and perform allocated functions. It checks whether the system as a whole works as per requirement. We have used Performance testing. Performance testing - designed to test the run-time performance of software, especially real-time software.

4.4.8 Performance Testing

This is designed to test the run-time performance of software within the context of an integrated system. Performance testing occurs throughout all steps in the testing process. Our system is checked for high load as well as low load.

4.4.9 Test Cases

A test case is a set of conditions or variables and inputs that are developed for a particular goal or objective to be achieved on a certain application to judge its capabilities or features.

It might take more than one test case to determine the true functionality of the application being tested. Every requirement or objective to be achieved needs at least one test case. Some software development methodologies like Rational Unified Process (RUP) recommend creating at least two test cases for each requirement or objective; one for performing testing through positive perspective and the other through negative perspective.

Test Case Structure

A formal written test case comprises of three parts -

Information

Information consists of general information about the test case. Information incorporates Identifier, test case creator, test case version, name of the test case, purpose or brief description and test case dependencies.

Activity

Activity consists of the actual test case activities. Activity contains information about the test case environment, activities to be done at test case initialization, activities to be done after test case is performed, and step by step actions to be done while testing and the input data that is to be supplied for testing.

Results

Results are outcomes of a performed test case. Results data consist of information about expected results and the actual results.

Designing Test Cases

Test cases should be designed and written by someone who understands the function or technology being tested. A test case should include the following information -

1. Purpose of the test
2. Software requirements and Hardware requirements (if any)
3. Specific setup or configuration requirements
4. Description on how to perform the test(s)
5. Expected results or success criteria for the test

Designing test cases can be time consuming in a testing schedule, but they are worth giving time because they can really avoid unnecessary retesting or debugging or at least lower it. Organizations can take the test cases approach in their own context and according to their own perspectives. Some follow a general step way approach while others may opt for a more detailed and complex approach. It is very important for you to decide between the two extremes and judge on what would work the best for you. Designing proper test cases is very vital for your software testing plans as a lot of bugs, ambiguities, inconsistencies and slip ups can be recovered in time as also it helps in saving your time on continuous debugging and re-testing test cases.

4.4.5.1 Test case for Certificate Monitor

Sr. No.	Test Condition	Expected Output	Actual Output	Pass/Fail
1	Get host recent certificate	Certificate Acquired	True	Pass
2	Save host recent certificate	Certificate saved	True	Pass
3	Validate certificate	Validation satisfied	True/False	pass
4	Parse host certificate	Certificate parsed	True	pass
5	Update cert bundle	Certificate bundle updated	True	pass

Table 4. 8 Test Case for Certificate Monitor

4.4.5.2 Test case for Certificate Monitor App Debugging

Sr. No.	Test Condition	Expected Output	Actual Output	Pass/Fail
1	Invoke sockets.tcpserver.listen	sockets.tcpServer.listen	sockets.tcpserver.listen	pass
2	Invoke sockets.tcpserver.create	sockets.tcpserver.create	sockets.tcpserver.create	pass
3	Save data to its local storage	storage.set	Storage.set	pass
4	Invoke blink request resource	blink.request.resource	blink.request.resource	pass
5	Retrieve data it has stored	storage.get	Storage.get	pass
6	Invoke sockets.tcp.send	sockets.tcp.send	Sockets.tcp.send	pass
7	Invoke sockets.tcp.onrecieve	sockets.tcp.onrecieve	sockets.tcp.onrecieve	pass
8	Notified of messages from other extension	runtime.onMessage externals	runtime.onMessage externals	pass

Table 4. 9 Test case for chrome app debugging

4.4.5.3 Test Case for Strict SSL Extension

Sr. No.	Test Condition	Expected Output	Actual Output	Pass/Fail
1	Redirect a request to a new webpage	Webrequest.onBeforeRequest	Webrequest.onBeforeRequest	pass

Table 4. 10 test case for Strict SSL

CHAPTER 5

Discussion and Conclusion

5.1 **Problems that Strict SSL and Cert Monitor set to solve.**

The two chrome based security solutions meant to enhance HTTPS security is to uncover the threat that results from unencrypted connections. The unencrypted connection may result from a client browser connection via http port 80. Also, there were trackers to website that handled user data or had the ability to gather client related data. These trackers if they set to connect over http unencrypted connection they may compromise the security of the of the client regardless the ability of the client to connect over secure https connections. Therefore, there was a need to connect the trackers mandatory over HTTP + SSL encrypted connection. Moreover, there is a possibility of questioning of the SSL/TLS certificate used to connect the encrypted HTTPS encrypted connection. This led to need to verify if the certificate integrity.

The motivation behind the problem was to set out to answer these two major problems.

1. Does the client browser connect over encrypted connection? And if that's the case
2. Is the integrity of the SSL/TLS certificate viable?

My security solution was set to answer the two questions with high degree of simplicity and articulation.

5.2 **How the Strict SLL set to ensure HTTP + SSL connections.**

The chrome extension Strict SSL ensured that the client browser requests were redirected to https connections and no user request was made to the server directly. This was achieved by design in the app system development. Strict SSL also allows the user to still connect to the websites that do not currently support HTTPS connection. Additionally, Strict SSL allow the user to cache a list of mandatory websites that he/she wishes to secure connect. By just a single button known as enforce.

5.3 **How the Certificate Monitor set to ensure the integrity of the of the certificate.**

The Certificate Monitor chrome browser application ensure the integrity of the SSL/TLS certificate used as is it compares the certificate against a trusted SSL/TLS certificate bundle supported and maintained the Mozilla foundation. Moreover, it also compared the certificate against the requested urls and the embedded urls. Furthermore, Certificate Monitor also checks a change in certificate by access the certificates vaults in the client machine and tracks changes to the certificate and displays any suspicious activity to the user. This is well illustrated in the design part of the system development phase.

5.4 Problem Encountered in the system conceptualization towards system actualization.

5.4.1 General problems encountered.

Challenged Faced	**Triumph Over Challenge**
Limited Development time	Scrum Methodology proved to an efficient and effective in the development of quality products over a limited time
SSL Certificate Design Modelling	The ability to acquire the SSL/TLS certificate was completely difficult as it led an actual implementation of TLS handshake in the chrome browser application
Development team exhaustion	The two products were developed by a team of two people.
Limited Finances	The testing of the connection required online connectivity as localhost test environment was limited to self-signed certificates. Self-signed certificated are really confusing to the end user as he/she may not realize the true integrity of the certificate.
Busy schedule	I used Wunderlist a to do application that categorized my daily activities in two parts 95% critical to do activities and the rest.
Team downtime	Intense development needs lots of coffee. I occasionally forgot to buy coffee.
Change Request during system development	The Scrum methodology played a real close part in achieving that change requests were successfully appreciated

Table 5. 1 General Problems Faced

5.4.2 Challenges faced during the development of Strict SSL chrome browser extension.

Challenged Faced	**Triumph Over Challenge**
Blacklisting and Whitelisting of urls in accordance to the set rules	The blacklisted and whitelisted urls were saved to a local storage in the client browser cache storage that was auto started during initializations of the chrome browser extension and browser
Respecting the Incognito mode of the chrome browse application	My applications had to be that the browser has opened an incognito tab and respect it according to the chrome browser incognito mode policy and restrictions.
Infinite loop detection of blacklisted urls	I made prioritization to the blacklist in code.
Chrome browser prioritization to load http connections	I have the extension to behave in such a way it caches the urls and when a url is enforced this shall also be enforced by the chrome browser even on reloading
Privacy concerns	I carried out stereotyping tests to reveal what my application had control over and permissions set.
Security of the extension	The chrome extension shall only be accessible for general use from the google chrome web store and shall be packaged by a private key which shall be used for any modification to the chrome browser extension
No passwords and Usernames	I stripped out any other detail from the urls during redirection and just got the domain.

Table 5. 2 Challenges faced during the development of Strict SSL chrome browser extension

5.2.3 **Challenges faced during the development of the chrome browser application.**

Challenged Faced	**Triumph Over Challenge**
Ability to get the host recent certificate	I was able to enable my application to access the Certificate store on the client host machine
Ability to get the host recent certificate	There is a workaround by utilization of chrome storage application programming interface (API). “chrome.storage.local.get”
Ability save Host certificate record.	There is a workaround by utilization of chrome storage application programming interface (API). “chrome.storage.local.set”
Behavior attacks on the chrome browser application	I disabled any logging information to the console even in code I commented out and shall only be made possible in a debugging process.
Validation of certificate	Certificates shall be validated by an ever-updated certificate bundle which shall be triple checked in the browser, requested urls and embedded urls
Updating the certificate bundle	Since the certificate bundle is strictly hashed by the Mozilla foundation I set an update to 24 hours to my application
Security of the application	The chrome extension shall only be accessible for general use from the google chrome web store and shall be packaged by a private key which shall be used for any modification to the chrome browser extension
Misconfigurations from the user	The app does have any user manipulation and the user is only notified through an extension.

Table 5. 3 Challenges faced during the development of the chrome browser application

5.2.4 **Challenges faced during Certificate Monitor app extension**

Challenged Faced	**Triumph Over Challenge**
Refresh rate for Notifications from the app	Initially there was no refresh rate but I set the refresh rate to 30 seconds rather than 1000 seconds
Ability for the extension to identify new tabs	I used a reset counter to change onto new tabs.
Security of the extension	The chrome extension shall only be accessible for general use from the google chrome web store and shall be packaged by a private key which shall be used for any modification to the chrome browser extension

Table 5. 4 Challenges faced during Certificate Monitor

5.2.5 How my application and extension have meet the objectives.

Objective	**Attainment of the Objective**
To mandatory enforce HTTPS connections on all trackers	All trackers to a website shall strictly connect to encrypted connection. This has been achieved by the chrome browser application Certificate Monitor
To enforce HTTPS connections on depreciated HTTP connections	Strict SSL has ensured that the requested web urls have been redirected immediately to connect to HTTPS connections. All encrypted connections shall be added to the enforced list automatically to reduce overhead. Strict SSL also checks if a site supports SSL/TLS and enforces all subsequent requests to be over SSL.
To inform the user on a change in SSL certificate and inform the user.	The chrome app has the ability to alert the user on the possibility of a connection to unencrypted connection and show the validity of the integrity of the certificate.
To reveal the status of the SSL certificate on all trackers	Certificate monitor displays through the chrome extension the status of the SSL certificate.

CONCLUSION

5.3.1 Recommendations

I recommend the application to every internet user who wishes to remain secure over encrypted connections and seeks to validate his/her secure connection in the most transparent manner with no additional hardware or software (or more precisely; at no further expense). I personally tell him/her this is the app to integrate and start a secure browsing experience.

5.3.2 Future Implementation.

Since Google has a Certificate Transparency Project that seeks to fix several structural flaws in the SSL certificate system, which is the main cryptographic system that underlies all HTTPS connections, by providing an open framework for monitoring and auditing SSL certificates this will allow me to develop a more resilience access to the basic components that drive certificate transparency such verification of certificates.

I would also to support the users as much as possible in the browsing secure community and customize security solution to their best need and interests of security.

5.3.3 Conclusion

Strict SSL and Cert Monitor chrome browser extensions and application respectively, provides the user with added security features in your browsing experience and greatly improves your privacy online. This is also made as transparent and automatic as possible. On Untrusted networks this is particularly important namely the internet via public connections such as public service vehicles, coffee shops and hotels.

Since Cert Monitor checks for third party connections known as trackers when you visit a page, it checks whether if these trackers connect over HTTPS. With this knowledge, you can tweet to the browsing secure community to create a change movement to the websites hosts and ask them to implement better security practices on their sites also you can also inquire the need of such trackers to the website site. However, remember to thank the sites that have implemented good security practices.

The security solutions have the advantage of being lightweight, informative, and simple to install.

REFERENCES

Barth, C. J. (2008). ForceHTTPS: protecting high-security web sites from network attacks,. *In Proceedings of the 17th International Conference on World Wide Web,.*

Brooks, J. F. (1978). *The Mythical Man-Month: Essays on Software Development.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Chen, M. W. (2009). Pretty-Bad-Proxy: An overlooked adversary in browsers' HTTPS deployments. *in Proceedingsof the IEEE Symposium on Security and Privacy.*

Comodo. (2011, 03 26). *Comodo Report of Incident - Comodo detected and thwarted an intrusion on 26-MAR-2011.* Retrieved from http://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html

Constantin, L. (2014, 4 8). *Low adoption rate of HSTS website security mechanism is worrying, EFF says*. Retrieved 12 1, 2016, from InfoWorld: http://www.infoworld.com/article/2610723/security/low-adoption-rate-of-hsts-website-security-mechanism-is-worrying--eff-says.html

Eckersley, P. (2015). *Sovereign key cryptography for internet domains*. Retrieved from pde@eff.org: https://git.eff.org/?p=sovereign-keys.git;a=blob;f=sovereign-key-design.txt;hb=master

Engert, K. (2013, 09 16). *DetecTor*. Retrieved from http://detector.io/DetecTor.html

Fahl, Harbach, Muders, Baumgärtner, Freisleben, & smiths. (2013). Rethinking SSL development in an appified world. *in Proceedings of the ACM Conference on Computer and Communications Security.*

Freedman, M. C. (2007). Peering through the shroud: the effect of edge opacity on IP-based client identification. *In Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation.*

Georgiev, (2012). The most dangerous code in the world: validating SSL certificates in non-browser software. *in Proceeding of Conference on Computer and Communications Security.*

Google. (2016). *Content Security Policy - Google Chrome*. Retrieved 12 1, 2016, from Developer.chrome.com.

Greg, S. (2014). *The trouble with SSL certificate.* Retrieved from okturles.org.

Hodges, C. J. (2012, 11). *HTTP Strict Transport Security (HSTS).* Retrieved from RFC 6797 (Proposed Standard).

Hoffman, C. (2012, 06 1). *Browser Plugins – One Of The Biggest Security Problems On The Web Today*. Retrieved 2016, from MakeUseOf: http://www.makeuseof.com/tag/browser-plugins-one-of-the-biggest-security-problems-on-the-web-today-opinion/

Holz, T. R. (2012). X. 509 forensics: Detecting and localising the SSL/TLS men-in-the-middle. *in Proceedings of the European Symposium on Research in Computer Security.*

Huang, E. Y. (2011). Talking to yourself for fun and profit. *in Proceedings of the Web 2.0 Security and Privacy.*

Jackson, A. B. (2007). Protecting browsers from DNS rebinding attacks. *in Proceedings of the ACM Conference on Computer and Communications Security.*

Keromytis, M. A. (2009). Doublecheck: Multi-path verification against man-in-the-middle attacks. *in IEEE Symposium on Computers and Communications.*

Kim, L.-S. H. (2013). Accountable Key Infrastructure (AKI): A proposal for a public-key validation infrastructure. *in Proceedings of the International Conference on World Wide Web.*

Kreibich, N. W. (2010). Netalyzr: Illuminating the edge network. *in Proceedings of the ACM SIGCOMM Conference on Internet Measurement,.*

Laurie. (2012). *Certificate Transparency Internet-Draft draft-laurie-pki-sunlight.* IETF.

Laurie, & Langley, E. K. (2012, 20 02). Certificate Transparency. *Internet-Draft draft-laurie-pki-sunlight*. IETF.

Marlinspike, M. (2011). SSL and the future of authenticity. *Black Hat Security.* USA.

Moxie, M. (2009). *New Tricks For Defeating SSL In Practice; BlackHat Security.* LAS VEGAS: blackhat.com.

Moxie, M. (2011, July 25). Retrieved from https://moxie.org/software/sslsniff/

Moxie, M. (2011, 5 15). Retrieved from https://moxie.org/software/sslstrip/

Moxie, M. (2016, November). Retrieved from http://www.thoughtcrime.org/software/sslstrip

Muders, F. B. (2012). Why eve and mallory love Android: an analysis of Android SSL (in)security. *in Proceedings of the ACM Conference on Computer and Communications Security.*

Mutton, P. (2016, 03 17). *95% of HTTPS servers vulnerable to trivial MITM attacks.* Retrieved from netcraft: https://news.netcraft.com/archives/2016/03/17/95-of-https-servers-vulnerable-to-trivial-mitm-attacks.html

Oppliger, (2009). *SSL and Tls: Theory and Practice.* Artech House Information Security and Privacy.

Reis, D. (2008). Detecting in-flight page changes with web tripwires. *in Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation.*

Rescoria, (2008). *RFC 5246 - the transport layer security (TLS) protocol version 1.2,.* Retrieved from RFC 5246: http://tools.ietf.org/html/rfc5246

Schwaber. (2002). *Scrum With XP.* Web.

Schwaber. (2003). *Agile Project Management with Scrum.* Microsoft Press.

Sleevi, R. (2016, 10 29). Google to Make Certificate Transparency Mandatory By 2017 . (threatpost.com, Interviewer)

Slepak, & Greg. (2014, 9 24). *The Trouble with Certificate Transparency.* Retrieved 121 1, 2016, from Blog.okturtles.com: https://blog.okturtles.com/2014/09/the-trouble-with-certificate-transparency/

Smith, M. (2012, 04 25). *New Variant Of Flashback For Mac Attacks Again [Updates]*. Retrieved 2016, from MakeUseOf: http://www.makeuseof.com/tag/variant-flashback-mac-attacks-updates/

Stamm, B. S. (2010). Reining in the web with content. *in Proceedings of the 19th International Conference on World Wide Web.*

Sunshine, E. A. (2009). Crying wolf: an empirical study of SSL warning effectiveness. *in Proceedings of the 18th USENIX Security Symposium.*

Vasco. (2011, 08). *DigiNotar reports security incident.* Retrieved from http://www.vasco.com/company/about_vasco/press_room/news_archive/2011/news_diginotar_reports_security_incident.aspx

Wendlandt, D. A. (2008). Perspectives: Improving SSH-style host authentication with multi-path probing. *in Proceedings of the USENIX Annual Technical Conference.*

APPENDIX.

Codes were generated by use of the PlantUML UML design software.

6.1 UML Code Syntax for StrictSSL Use Case diagram.

```
@startuml


user --|> (request URL): visit to website

(request URL) --|> app

user -> app

:app: -> (http URL redirection)

(http URL redirection) ---> (connects to HTTP + SSL):enforced urls to connect
to HTTP + SSL

:app: <..> (Detects URL): for HTTP + SSl connections

:app: --> (connects to HTTP + SSL): for supported sites

(request URL) ---> (connects to HTTP + SSL): after redirection by the app


@enduml
```

6.2 UML code Syntax for Certificate Monitor use case diagram.

```
@startuml


(active) -> (extension) :listens for user requests

(extension) -> app : redirects user urls for certificate parsing

app <-( host certificate): gets

app -> ( host certificate): saves

app->( host certificate): validates

( host certificate) <-down-> (Certificate Bundle): compared

app <-up-->(Certificate Bundle): compares

app -left-->(extension): notifies extension

app -up-> (Certificate Bundle): updates


@enduml
```

6.3 UML code Syntax for Top-Down Testing Methodology Diagram.

```
@startuml

autonumber
[-> Start_test: do_test
activate Start_test
Start_test -> Start_test:   Acceptance_testing
activate Start_test
Start_test -> Start_test:  System_testing
activate Start_test
Start_test -> Start_test: Sub-System-Intergration_testing
activate Start_test
Start_test -> Start_test: Module_testing
activate Start_test
Start_test -> Start_test: Unit_testing
activate End_test
Start_test -> End_test : request_for_test_satisfaction
Start_test<-- End_test : satisfaction_meet
deactivate Start_test
[<- Start_test: Done
deactivate End_test
@enduml
```

6.4 UML code syntax for Certificate Monitor Sequence Diagram.

```
@startuml
autonumber
[-> CertMonitor : do_monitor
activate CertMonitor
CertMonitor-> hostRecentCertificate:   get
activate hostRecentCertificate
hostRecentCertificate -> validateCertificate: validates
activate validateCertificate
activate  pasrseHostCertificate
activate keychain
activate certificateBundle
activate certificateBundle
```

```
validateCertificate -> pasrseHostCertificate :compares
validateCertificate -> keychain: validates
validateCertificate -> certificateBundle: online certificate transparency
CertMonitor -> certificateBundle:update
activate End_validation
validateCertificate -> End_validation : request_for_test_satisfaction
validateCertificate<-- End_validation : satisfaction_meet
deactivate validateCertificate
[<- validateCertificate: Done
deactivate End_validation

@enduml
```

6.5 UML code syntax for StrictSSL Activity Diagram.

```
@startuml


(*) -->[for pause=false\n  for redirection=true\n] "Initialization \nAuto
Enforce\nAuto whitelist\n"
..> "Encrypted connection"
-down..>[cached urls] "Initialization \nAuto Enforce\nAuto whitelist\n"
-left--> "check domain"
if "on Enforced list?" then
  -right-->[true] "Enforce HTTP + SSL"
  -right--> "Encrypted connection"
  --> (*)
else
  -->[false] "Terminate connection"
  if "Whitelisted" then
  -->[true] "allow Connection"
-->(*)
else
->[false] (*)
Endif

@enduml
```