KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Empirical Study on the Use of Client-side Web Security Mechanisms

Ping Chen

Supervisors:
Prof. dr. ir. W. Joosen
dr. ir. L. Desmet

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

August 2018

Empirical Study on the Use of Client-side Web Security Mechanisms

Ping CHEN

Examination committee:
Prof. dr. ir. Y. Willems, chair
Prof. dr. ir. W. Joosen, supervisor
dr. ir. L. Desmet, supervisor
Prof. dr. ir. C. Huygens
Prof. dr. ir. V. Rijmen
Prof. dr. J. Davis
Prof. dr. ir. E. Steegmans
Prof. dr. A. Rashid
(University of Bristol, UK)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

August 2018

© 2018 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Ping Chen, Celestijnenlaan 200A box 2402, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Preface

As the Web has become an essential part of our society, it is important to ensure the security of web applications. This dissertation focuses on measuring web security by investigating the use of client-side defensive mechanisms. It is the final result of my PhD research in the DistriNet research group. This work would not have been possible without the help of many others. Thus, in this preface, I would like to take the time to thank everyone who supported me during the course of my PhD.

First and foremost, I am grateful to my supervisor Wouter Joosen, for giving me the opportunity to join DistriNet family, and guiding me throughout my PhD. Despite his busy schedule, he has always provided me helpful feedbacks at critical points. My sincere gratitude also goes to my co-supervisor Lieven Desmet, for the daily support and encouragement over the past five years. He has patiently leaded me through difficult times during my PhD research. I also want to thank Christophe Huygens for guiding my research.

Second, I would like to thank other memebrs of my doctoral jury, Yves Willems, Vincent Rijmen, Jesse Davis, Eric Steegmans and Awais Rashid, for reading my thesis and providing comments that greatly improved this text.

During my PhD, I enjoyed the friendly environment in the DistriNet research group and the Computer Science Department. I am thankful to several past DistriNet members, Nick Nikiforakis, Steven Van Acker, and Philippe De Ryck, for providing help to my research on web security; and present colleagues Tom Van Goethem, Zubair Rafique, and Thomas Vissers, for the collaboration and inspiring discussions. I also want to thank my office mates in 04.152, Neline van Ginkel, Jo Van Bulck and Andreas Nuyts, for the fun and chatting. And a big thank you to DistriNet business office members: Ghita Saevels, Katrien Janssens and Annick Vandijck. Thank everyone in the department for making such an amicable environment.

Apart from all the colleagues in the department, I would like to thank several

friends in Belgium. Thank my housemates Chen Ruiqi and Hu Kaide, for helping me settle down in Leuven, and encouraging me to keep going. Thank Qing Xiaoyu and Huang Xu, for the strolls and other interesting activities.

I am also thankful to other friends in Europe, in the Netherlands: Roberto Lee, Hu Kaipeng, and Li Nan; in Germany: Zou Hanhai and Liu Jia; and in United Kindom: Lü You. And with special thanks to Hu Kaipeng, for being like a brother to me, always giving me support.

My deepest gratitude goes to my family in China, in particular my parents Qiusheng and Guiying, my sister Lijuan, for all their love and understanding over the years. Since they are non-English speakers, I want to express my thanks in Chinese:

天涯游子寸草心，无以为报三春晖。谢谢父母多年的理解与支持！

Lastly, I would like to thank the following institutions and projects for their financial support: the Research Fund KU Leuven, the EU FP7 projects WebSand, NESSoS and STREWS, the iMinds project TRU-BLISS, the Belgian Cost of Cybercrime Research project funded by Belgian Science Policy Office (BELSPO), and the Prevention of and Fight against Crime Programme of the European Union (B-CCENTRE).

Ping Chen
Leuven, August 2018

Abstract

Nowadays, no one disputes the fact that the web has become an essential part of our society. More and more organizations and individuals are relying on the web for almost all kinds of activities. Naturally, the rising importance of the web attracts an increasing number of web attacks. As the web (and the attacks) keep on expanding, it is important for website operators to ensure the security of their web applications. To defend against a rising number of web attacks, one basic yet important step is to adopt various known defense mechanisms that have been developed by the security community.

In this dissertation, we assess the security of websites from the adoption perspective of these client-side defense mechanisms. Client-side security mechanisms are configured and controled by web servers, and they help websites to reduce their client-side attack surface. As such, the presence of these mechanisms on a website might be used as an external indicator of the security awareness and practices of the website owner.

Firstly, we discuss the eight most-important client-side defense mechanisms that are used as metrics to design a web scoring system to measure the security of web applications.

Secondly, we propose an efficient crawling approach for large-scale web assessments to measure the adoption of these mechanisms. We then use this crawling approach to investigate mixed-content inclusion weaknesses, to conduct a security assessment for the Chinese Web, and to perform a longitudinal assessment on the adoption of client-side security mechanisms on the European Web. By quantifying a website's security level as a web security score, we can compare the security maturity of websites per country, sector and popularity.

Lastly, we explore the relationship between a company's cybercrime cost and the adoption of defense mechanisms on its website. Our correlational analysis shows that companies with better security defense tend to have less business loss caused by web attacks.

Beknopte samenvatting

Niemand zal betwisten dat het web tegenwoordig een essentieel onderdeel van onze samenleving geworden is. Steeds meer organisaties en individuen vertrouwen voor allerlei activiteiten op het web. Tergelijkertijd brengt het toenemende belang van het web ook steeds meer aanvallen met zich mee. Aangezien het web (en de aanvallen) blijven groeien, is het belangrijk voor website-exploitanten om de veiligheid van hun webapplicaties te blijven waarborgen. Om zich te verdedigen tegen de toenemende aanvallen op het web, is het gebruik van de reeds bestaande verdediging mechanismen een fundamentele maar belangrijke stap.

In dit proefschrift analyseren we de beveiliging van websites op basis van het gebruik van beveiligingsmechanismen aan de gebruikerszijde. Het gebruik van beveiligingsmechanismen aan de gebruikerszijde worden door webservers geconfigureerd en gecontroleerd, en helpen om het aanvalsoppervlak van websites aan de gebruikerszijde te verkleinen. De aanwezigheid van deze mechanismen op een website kan dus mogelijk gebruikt worden als een externe indicator van het veiligheidsbewustzijn en de veiligheidspraktijken van de website-eigenaar. Als eerste bespreken we de acht belangsrijkste verdedigingsmechanismen aan de gebruikerszijde die verder als maatstaven gebruikt worden voor hete meten van de veiligheid van webapplicaties.

Ten tweede stellen we een efficiënte webcrawl methode voor om grootschalige veiligheidsanalyses uit te voeren op basis van de aanwezigheid van dergelijke verdedigingsmechanismen. Vervolgens gebruiken we de webcrawl methode om de aanwezigheid van mixed-content zwakheden te onderzoeken, om een veiligheidsanalyse van het Chinese web uit te voeren, en om de evolutie op lange termijn te onderzoeken wat betreft het gebruik van client-side beveiligingsmechanismen op het Europese web. Door het beveiligingsniveau van een website te kwantificeren als een web security score, kunnen we veiliheidsmaturiteit van websites per land, sector en populariteit met elkaar vergelijken.

Tot slot onderzoeken we de relatie tussen de cybercrime kosten van een bedrijf en het gebruik van verdedigingsmechanismen op de website. Onze correlatieanalyse toon aan dat bedrijven met een betere verdediging aan de gebruikerszijde typisch minder zakelijke verliezen door aanvallen vertonen.

Contents

List of Figures

List of Tables

Abbreviations

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
APT	Advance Persistent Threat
CA	Certificate Authority
ccTLD	country code Top-Level Domain
CERN	European Organization for Nuclear Research
CI	Confidence Interval
CNNIC	China Internet Network Information Center
CSP	Content Security Policy
CSS	Cascading Style Sheets
CSRF	Cross-Site Request Forgery
CT	Certification Transparency
DNS	Domain Name System
DANE	DNS-based Authentication of Named Entities
DOM	Document Object Model
ECDF	Empirical Cumulative Distribution Function
EU	European Union
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HTTP Secure, HTTP over TLS/SSL
HPKP	HTTP Public Key Pinning
HSTS	HTTP Strict Transport Security
ICP	Internet Content Provider
ID	Identity
IE	Internet Explorer
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISP	Internet Service Provider

JS	JavaScript
MIME	Multipurpose Internet Mail Extensions
MITM	Man-In-The-Middle
OS	Operating System
OWASP	Open Web Application Security Project
SCT	Signed Certificate Timestamps
SE	Standard Error
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UI	User Interface
URI	Unifrom Resource Identifier
URL	Uniform Resource Locator
VM	Virtual Machine
VPN	Virtual Private Network
WWW	World Wide Web
W3C	World Wide Web Consortium
XCTO	X-Content-Type-Options
XFO	X-Frame-Options
XML	Extensible Markup Language
XSS	Cross-Site Scripting

Chapter 1

Introduction

We might not have noticed but the Web has become an inalienable part of our life. As of December 2017, more than half of the world's population is connected to the Internet [42]. Billions of users spend several hours online on a daily basis, mostly interacting on social networking web applications, such as sharing photos ("snap") in Snapchat, or following stories ("moment") in WeChat. Meanwhile, businesses are also increasingly depending on the Web for their day-to-day operations. As a noticeable example, the global e-commerce built upon the Web has been expanding at an average rate of 20% per year over the past decade, with trillions of dollars spent online per year, according to statistics from *Euromonitor* [13].

Despite its importance, most web applications are not well protected. According to a study by *WhiteHat Security*, the average age of an open critical vulnerability is over 300 days [133]. In other words, most web applications are vulnerable most of the time. The lack of web protection and the presence of vulnerabilities have given attackers a good chance. The past decade has seen increasing cybercrime activities on the web, which inflicted severe damage and harm to our society. A report about cybercrime [141] estimates that the global losses caused by cybercrime was $3 trillion in 2015 and the figure is projected to double by 2021.

Web attacks happen every day, and they affect both companies and individuals. Companies that have become victims of cyber security incidents suffer from business disruptions, which causes monetary losses in the short term, and reputation damage, which causes long-term harmful effects, with customers losing trust in them. Indeed, security breaches on websites can induce significant collateral damage to individual users as well. For instance, a stolen database that contains private data such as home address, mobile number, and identity

information, can be leaked online, or be sold in underground black markets for various illegal purposes.

Internet users in developed countries have long been concerned with their personal information kept by companies. The European Union already adopted the Data Protection Directive to protect private data back in 1995 [43], which has been recently upgraded to the General Data Protection Regulation [49] in 2016 and takes effect from May 2018. And recent years have also witnessed a rise of consumer privacy awareness in developing countries such as China [12]. The increasing security and privacy awareness results in stricter regulations and more complaints about privacy issues from customers.

Amid growing cyber threats, tightening regulations, and more demanding customers, ensuring the security of web applications has become critical for website operators. But this is not an easy task, as web attacks are constantly evolving. Instead of looking for a single silver-bullet solution, website owners should adopt a risk-based approach, treating cyber attacks as always-present risks. And to manage the risks properly, they can regularly assess their security posture and implement known defensive mechanisms to improve security protection.

1.1 Goal of This Thesis

The focus of this thesis is assessing websites security from the aspect of adopting defensive mechanisms.

To assess a website's security, a conventional approach is actively searching for vulnerabilities, which typically involves internal penetration testing and source code reviewing. While this approach is considerably effective in finding specific security issues, it is time and labour consuming and hence lacks scalability in both scope and consistency. Firstly, the traditional approach is limited in the scope of analysis. For big companies that have many web applications and a complex corporate structure, it is impractical to apply penetration testing to all the applications. Also, penetration testing is usually conducted in an isolated environment to void causing possible disruptions to service, so the analysis does not completely reflect the daily situation that an application encounters.

Moreover, the traditional approach provides only a snapshot analysis at a particular point in time. As web technology is changing fast and attackers move fast as well, the snapshot analysis might become inaccurate after a period of time. Ideally, companies can hire people to carry out periodical analysis. However, as penetration testing is costly, few organizations would do that. This

is particularly true for small and medium-sized enterprises (SMEs), who often do not have the resources for regular assessments. Thus, the traditional approach lacks long-term consistency.

In order to have a broad and consistent view of websites security, this dissertation proposes a complementary approach that stands outside of an application to check for the presence of defense externally, instead of going inside an application to look for vulnerabilities internally (as the conventional approach does). In other words, the proposed approach tries to measure how well a web application is protected from an outside perspective, which can serve as a proxy for its internal security. As web applications employ a client-server architecture, an outside view is the client-side view. Thus, the proposed approach focuses on the presence of defensive mechanisms that can be detected from the client side, henceforth referred to as client-side security mechanisms.

The client-side security mechanisms are developed by the security community to defend against various kinds of web attacks. While these mechanisms cannot completely prevent web attacks, they can be seen as basic hygiene practices, which help websites to reduce the attack surface and alleviate the consequences when attacks occur. Just as a man with good personal hygiene is less likely to get sick, a website can improve web security hygiene to minimise risks. Moreover, since some of these mechanisms are visible to the users (e.g., a green secure lock icon in the browser), adopting them also sends a positive signal to users, which helps an organisation to build trust with its customers.

Although the client-side security mechanisms requires a compliant client (web browser) to enforce the specified policies, the policies are specified and sent by the web server. Thus, the presence of these mechanisms on a website can be used as an indicator of the security awareness and practices of that website. By checking for the presence of client-side security mechanisms, we can have an approximate view of a web application's security posture. And this proposed approach for external assessment has several advantages, which complements the traditional internal assessment.

The main advantage is its simplicity and efficiency. Since client-side security mechanisms can be detected from the client side, the assessment can be carried out simply by visiting webpages in a browser, which does not cause any harm to the assessed website. With a customized web crawler that mimics a normal browser's behaviour, the assessment can be automated, which brings efficiency. As shown in later chapters, such an external assessment on a thousand websites can be done in just a few days.

The automated efficiency in turn brings scalability. By standing outside to conduct external assessment, one can straightforwardly expand the assessment

to include a large number of websites, which enables a much broader view. Thus, the assessor can have a bird's eye view of the security posture of a group of web applications. Furthermore, it allows the assessor to compare different (groups of) websites based on the adoption of client-side security mechanisms. In this way, a company can have a horizontal comparison, i.e., comparing itself to its peers in the same industry and learn from the leading peers.

The proposed approach also scales well with respect to long-term consistency. An assessor can employ it regularly to assess the security posture. By doing this, a company can achieve a vertical comparison over periods of time, i.e., comparing itself to its historical performance and find room for improvement. Such an external comprehensive assessment is also desirable for government and supervisory organizations to continuously monitor the security of the web environment.

This dissertation first presents the approach for assessing websites security from the aspect of adopting defensive mechanisms, and then shows its effectiveness through several large-scale assessments.

1.2 Other Research Conducted

The works presented in this dissertation are a subset of my PhD research. For the consistency of this dissertation, I have selected works of which I am the principal author and which are related to web security. Below I give a brief description of the other research I conducted during my PhD.

A study on Advanced Persistent Threats This paper [68] presents a comprehensive study on Advanced Persistent Threats (APTs). We characterize the distinguishing characteristics and attack model of APTs, and then analyze the adversaries' tactics and techniques through case studies of four APTs. We also enumerate some non-conventional countermeasures that can help to mitigate APTs.

Publication data: P. Chen, L. Desmet, C. Huygens. "A study on Advanced Persistent Threats". In *Proceedings of the 15th IFIP TC6/TC11 Conference on Communications and Multimedia Security*, pages 63-70, 2014.

Advanced or not? A comparative study of the use of anti-debugging and anti-VM techniques in generic and targeted malware This work [70] investigates the use of anti-debugging and anti-VM techniques in modern malware, and compare their presence in 16,246 generic and 1,037 targeted malware samples (APTs). Our study found that targeted malware does not use more anti-debugging and anti-VM techniques than generic malware, although

targeted malware tend to have a lower antivirus detection rate. Moreover, this paper also identifies a decrease over time of the number of anti-VM techniques used in APTs.

Publication data: P. Chen, C. Huygens, L. Desmet, W. Joosen. "Advanced or not? A comparative study of the use of anti-debugging and anti-VM techniques in generic and targeted malware". In *IFIP Advances in Information and Communication Technology, ICT Systems Security and Privacy Protection*, volume 471, pages 323-336, 2016.

Evolutionary algorithms for classification of malware families through different network behaviors This paper [124] presents a framework to efficiently classify malware families by modeling their different network behaviors. We propose protocol-aware and state-space modeling schemes to extract features from malware network behaviors. We analyze the applicability of various evolutionary and non-evolutionary algorithms for our malware family classification framework.

Publication data: M. Z. Rafique, P. Chen, C. Huygens, W. Joosen. "Evolutionary algorithms for classification of malware families through different network behaviors", In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, pages 1167-1174, 2014.

1.3 Contributions

Despite the increasing reliance on the web for business, and the growing financial losses caused by web attacks, many organizations fail to manage the security risks properly. This is mainly due to a lack of understanding of the impact of web security on business, and insufficient evidence to support policy decision making. In this thesis, we explore what technology evidences can be collected and analyzed to underpin an organization's policy to address security risks.

The main contributions of this thesis are twofold. Firstly, we designed a quantitative approach, based on the adoption of client-side security mechanisms, to measure websites security. Secondly, with the proposed approach, we investigated the relationship between a website's security and the cost of cybercrime induced to that site. To the best of our knowledge, this was the first interdisciplinary work that examines this association.

Our study showed that companies with better web security defenses tend to have less business loss and reputation damage. This finding, combined with the proposed approach to assess web security, can help organizations to make proper policy, based on data, to reduce the cybercrime cost. Our approach allows an

assessor to continuously monitor the security posture of websites externally, and it has been validated in collaboration with several financial institutions in Belgium.

Such kind of external assessment may also be helpful for other parties - for example, it can serve as a pricing factor for insurance company to establish cyber insurance premiums; it might also desirable for government or supervisory organizations to make proper policy based on evidence.

As for the other research related to targeted malware, our main contribution is to systematically define the attack model of Advanced Persistent Threats (APTs), and analyze techniques commonly seen in APT attacks. Our paper is among the first academic works that studies APTs comprehensively, and inspired others [107, 130, 108, 123] to this area.

1.4 Outline of the Text

In this dissertation, we discuss web security assessment from the client side. To help prepare the readers to understand the topic, we first introduce the basic knowledge of the web platform, and explains some common web attacks in Chapter 2. It sets the context in which this work should be viewed.

In a first contribution (Chapter 3), we enumerate eight client-side defense mechanisms that websites can adopt to maintain good web security hygiene. The presence of these mechanisms can be used as metrics to assess websites security. We then propose an efficient crawling approach for large-scale web security assessments based on discovering the use of client-side security mechanisms.

The effectiveness of this approach is shown in several published conference and workshop papers, which are adpated into four chapters for this thesis:

- Chapter 4 presents a thorough survey and analysis of mixed-content inclusion (HTTP contents included on HTTPS webpages), using our web crawling approach. This work was the first attempt to systematically investigate the prevalence and impact of mixed HTTP content on HTTPS webpages. It also documents the best practices to mitigate the issue for browsers, website owners and content providers. This work was published and presented at the 16th Information Security Conference [72].
- Chapter 5 gives a large-scale security assessment of the Chinese Web. The measurement is based on the usage of client-side security mechanisms and the strength of HTTPS implementations. We assessed the top 10,000

Chinese websites, and compare them to a set of 10,000 non-Chinese websites (comprised of neighboring non-Chinese sites in Alexa's top list). It also reports a severe inadvertent private data leakage issue that is unique in China. This work was published and presented at the 2014 Workshop on Cyber Security Analytics, Intelligence and Automation [71].

- Chapter 6 examines a long-term web evolution with respect to the adoption of client-side security mechanisms, through a four-year longitudinal security assessment of the European Web. It also proposes a web security scoring system based on the usage of client-side security mechanisms to compare the security postures among different (group of) websites. Using the scoring system, we compared websites from different business categories and different countries. We also identified several longitudinal trends. This work was published and presented at the Workshop on Empirical Research Methods in Information Security [69].

- As adopting client-side security mechanisms indicates a website's security awareness, we expect websites with better web hygiene will be more secure and less vulnerable to attacks, hence fewer losses incurred due to cybercrime. To verify this hypothesis, we study the relationship between a company's cybercrime cost and the adoption of client-side security mechanisms on its website in Chapter 7. We surveyed 263 Belgian companies about the impact of cybercrime on their business, and gathered the statistics on the usage of security features through website crawling. This work was published and presented at 4th International Workshop on Measurability of Security in Software Architectures [73].

We conclude the dissertation (Chapter 8) by revisiting the contributions, reviewing related works, and considering opportunities for future research.

Chapter 2

Background

Since its founding in 1989, the World Wide Web has significantly influenced human society. Browsing the Web has become an important part of people's daily lives, and the widespread use of smartphones in recent years further intensified this trend, as the always-connected mobile devices allow people stay online wherever they go. The web has also rapidly expanded into all kinds of business, becoming a driving force of a nation's economy.

In the meantime, the rising importance of the Web also attracts increasing attention from attackers. Cybercriminals have been leveraging the web platform to launch attacks against business and individuals, which costs global economy trillions of dollars per year. Web attacks occur every day, and even renowned giant companies such as Google are not immune to attacks, let alone ordinary web users.

The problem of widespread cyber attacks is essentially an issue of lacking risk management. Many companies are unaware of security risks when adopting web technologies, which results in inadequate effort on protection. Most web users are also not paying enough attention to web security. For both business and individuals, the lack of security awareness is partly due to an insufficient understanding of the Web.

Although people use the Web on a daily basis, many do not know the technology behind it. This chapter introduces the basic knowledge of the Web platform, and explains some common web attacks, which can help to raise security awareness among people. The information presented in this chapter also helps prepare the reader for the discussion in the following chapters.

2.1 The World Wide Web

W3C, an international standards organisation for the Web, defines the World Wide Web (WWW or the Web) as "an information space in which the items of interest, referred to as resources, are identified by global identifiers called Uniform Resource Identifiers (URI)." [103] This definition is too technical for a layman who might prefer the explanation given by Oxford Dictionary: "An information system on the Internet which allows documents to be connected to other documents by hypertext links, enabling the user to search for information by moving from one document to another."

Oxford Dictionary's definition is more user-friendly, and it points out the relationship between the Web and the Internet. Many people often confuse them, and use the words "Web" and "Internet" interchangeably. However, they are not synonymous. In reality, the Web is just an application running atop the Internet. A simplistic difference between them is that the Internet is built to connect computers, while the Web is built to connect people, i.e., to help people share information on the Internet.

More technically speaking, the Internet is a global system of computer networks that are interconnected using the TCP/IP protocols. Built on top of the Internet, the Web is an information space where resources are hosted on different networked computers, identified by Uniform Resource Locators (URLs), interlinked by hypertext links, and can be accessed through the HTTP protocol. The three fundamental elements of the web are URLs, HTTP, and HTML.

A Uniform Resource Locator (URL) [64], colloquially called a web address, is a reference to a resource on the web. It is a specific type of Uniform Resource Identifier (URI) [63] that, in addition to identifying a resource, specifies the location and access mechanism for the resource. The structure of a typical URL is illustrated in Figure 2.1, which includes a scheme, a hostname, and a path.

Figure 2.1: The structure of a typical URL

In addition to these basic URL parts, a generic URL can also specify authentication credentials (a username and password), a port number, a query, and a fragment. An example of a generic URL is: `http://user:pass@example.com:8080/path?key=value#fragment`.

An URL contains all necessary information for a web browser to retrieve the specified resource from a web server that hosts the resource. The communication protocol used between web browser and web servers is HyperText Transfer Protocol (HTTP) [90]. HTTP is a request/response-based client-server protocol, which allows a client (a web browser or other user agents) to retrieve data from a web server, and to submit data to a web server. Figure 2.2 shows an example of an HTTP request and response process, in which a client gets a document from a server.

Figure 2.2: An example of an HTTP request and response

HTTP can be used to transfer various types of contents (e.g., plain text, image, audio), of which the most basic and popular type is an HTML document. HTML (HyperText Markup Language) [62] is a markup language that describes the structure of a webpage semantically. As shown in Listing 2.1, an HTML document consist of nested HTML tags (also known as HTML elements) and text content placed between the tags. HTML tags are enclosed in angle brackets, and usually used in pairs. The entire document is marked with the <html> tag. Metadata about the webpage, such as the title, is placed inside the <head> tag. The <body> tag specifies the main content area of a webpage, where more than 100 different HTML tags can be used to organise the content. For example, the <a> tag creates a reference (or hyperlink) to other webpages, files, locations within the same page, or any other URLs.

```
1 <html>
2     <head>
3         <title>A simple webpage</title>
4     </head>
5     <body>
6         <h1>This is a heading</h1>
7         <p>This is a paragraph that contains a reference (URL) to
              <a href="http://www.example.com">example.com</a></p>
8     </body>
9 </html>
```

Listing 2.1: A simple HTML document

2.2 History of the Web

The World Wide Web was born at CERN (European Organisation for Nuclear Research) in 1989, originally developed for automatic information sharing between scientists [38]. Tim Berners-Lee, a British scientist at CERN, is considered as the father of the Web. He issued a proposal to build a web of hypertext documents, and led a team that created HTTP and HTML. By December 1990, Berners-Lee had built the first web server and web browser, as well as the first web site, which described the project. In 1993, CERN made the World Wide Web software publicly available with an open licence, which allowed the Web to flourish.

In the early days of the Web, most web contents are just text and hyperlinks. Browsers rely on external helper applications to view images. This plain landscape was changed in 1993 with the release of the Mosaic browser [27], which was the first browser able to display images along with text. With this feature and an attractive interface, the Mosaic browser enriched users' web browsing experience, and greatly popularised the Web.

In 1994, many of the original Mosaic browser's developers were employed by Netscape to create a new browser, called Netscape Navigator. The release of Netscape Navigator 2.0 in 1995, further influenced the Web with the introduction of JavaScript. JavaScript (JS) is a high-level interpreted programming language that can be used in webpages to make them dynamic and interactive. JavaScript quickly proved to be a success, and other browsers, such as Microsoft's Internet Explorer, started to adopt this technology.

Another technological upgrade for the Web is the use of Cascading Style Sheets (CSS) [111]. Along with HTML and JavaScript, CSS is one of the three core technologies for web content production. Often used to set the visual style of webpages, CSS is a computer language that expresses the presentation of structured documents. It enables the separation of presentation and content, which provides more flexibility and control on the formatting and rendering of the content. For example, CSS can specify a webpage to display the same content differently depending on the screen size or viewing devices.

Through the history of the Web, browser vendors have played a very prominent role. For most ordinary users, a web browser is the only way to access the Web. To attract more users, different browsers had been introducing various new features that were incompatible with each other. The "first browser war" between Netscape's Netscape Navigator and Microsoft's Internet Explorer in the late 1990s, saw the proliferation of non-standard and proprietary extensions to the HTML standard. In an effort to standardise web technology, Tim Berners-Lee left CERN and founded the World Wide Web Consortium (W3C) in 1994.

To solve the incompatibility problem of different browsers, W3C sets a series of standards, guidelines, and recommendations for web developers.

After two decades of development, the Web has become an increasingly dominant player on the Internet. Instead of using the Internet (TCP/IP) directly, more and more services are leveraging the power of the Web (HTTP), making the Web a rich application platform. The modern web is no longer limited to displaying information on static webpages. It also supports multimedia content, and allows user interaction. This new stage that features massively increased user participation is known as Web 2.0 [118].

In the era of Web 2.0, end-users become a pivotal part of the websites. For example, users can build social networks with other people in a virtual community (e.g., Facebook), create user-generated content for others to see (e.g., Wikipedia), work collaboratively with others in real-time (e.g., Google Docs), broadcast to millions of audiences in live-streaming (e.g., YouTube).

With the emergence of Web 2.0, websites become powerful interactive applications. Unlike desktop applications which need to be installed on a computer, web applications run in a browser and do not need to be installed. With a web browser, users can access web applications from anywhere, at any time.

2.3 Web Browser

The main function of a web browser (a browser) is to retrieve information resources on the Web, and to display them in a visual window for the user. The information resource, identified by a URL, is usually an HTML document, but may also be image, video or other type of content. Modern browsers typically have a modular architecture, which consists of eight subsystems, as shown in Figure 2.3:

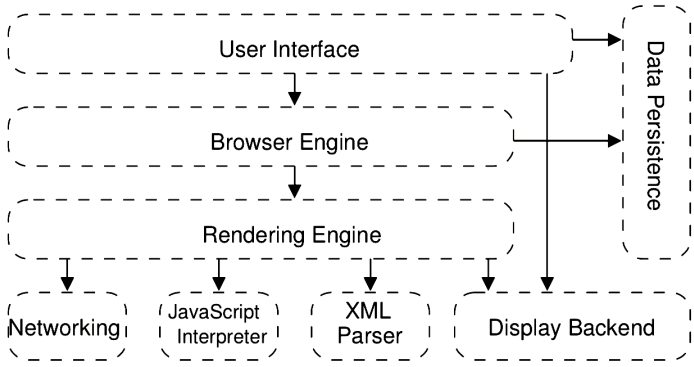

Figure 2.3: Reference architecture for web browsers [94]

- **User Interface.** The visual interface that interacts directly with users, which includes address bar, toolbars, and other buttons/menus.
- **Browser Engine.** A high-level interface to the underlying Rendering Engine, which handles browsing actions such as URL loading, forward, backward, and reload. It also provides hooks that allows querying and manipulating the Rendering Engine.
- **Rendering Engine.** The subsystem that creates a visual representation for the requested resource. It includes an HTML parser that turns an HTML document into a DOM (Document Object Model [143]) tree, wherein each node is an object representing a part of the document. The DOM allows scripts to dynamically access and update the content, structure and style of documents. The Rendering Engine is also responsible for parsing CSS and displaying embedded content such as images.
- **Networking Subsystem.** This component provides network communication between a browser and a server, through protocols such as HTTP. Besides transferring data, it converts data between different character sets, and resolves MIME types for files.
- **JavaScript Interpreter.** Also known as JavaScript Engine, this subsystem is used to parse and execute JavaScript code. In the browser, JavaScript can manipulate the webpage through the DOM.
- **XML Parser.** This subsystem parses XML documents into a DOM tree. The XML parser is a generic, reusable component. It is different from the HTML parser that is often tightly integrated with the rendering engine for performance reasons.

- **Display Backend.** This backend exposes a generic interface for drawing basic widgets like combo boxes and windows.
- **Data Persistence.** A storage subsystem for various data associated with the browsing session, such as bookmarks, browsing history, cookies, security certificates, and browser settings.

  The modular architecture allows different browsers vendors to share the same components. For example, the Blink [9] browser engine, forked from WebKit [41] and mainly developed by Google, is used in Google Chrome, Opera, and other Chromium-based browsers. Browser subsystems are not only reused by web browsers, but may also appeared in other desktop applications. Internet Explorer's layout engine, Trident [21], is also used by many applications on the Windows platform, such as Outlook Express. An interesting fact about the reuse of browser subsystems is, Chinese browsers often adopt a "dual-core engine" architecture for compatibility reasons, i.e., use two different browser engines. For instance, Maxthon uses Webkit and Trident; 360 Browser uses Blink and Trident. The Trident engine is used typically for accessing legacy Chinese online banking applications that are only compatible with Internet Explorer's specifications.

2.4 Web Application

The prominent feature of Web 2.0 is massive user interaction. To enable this feature, modern web applications relies on two important techniques: asynchronous JavaScript execution and session management.

Consider a common scenario of a user following a Twitter account on a webpage. Once the page URL is obtained, the user's browser retrieves HTML code and other resources from the server, and then renders the page in a window. Without the use of asynchronous JavaScript, the user would need to manually refresh the page every now and then, in order to see if new tweets are popping up. This is because the HTML code of the webpage is generated and sent by the web server, and it could not update itself without JavaScript.

But with asynchronous JavaScript techniques such as Ajax [92], web applications can retrieve data from a server asynchronously without interfering with the display and behaviour of the existing page. In this way, Twitter's server can insert JavaScript code that regularly contacts the server to retrieve the latest information in the background, and updates the webpage in seemingly real time. The result is an active webpage that always displays the latest tweets.

Besides the interactivity enabled by JavaScript, modern web applications also support authenticated sessions to keep track of users' state. The authentication step is often completed by requesting the user to enter a username and a password in an HTML login form. By submitting the form, the credentials are sent to the server for validation. However, because HTTP is a stateless protocol that does not require the server to retain information about each user for the duration of multiple requests. To avoid re-authentication in subsequent requests, web applications need a session management mechanism.

Once a session is established between a web application and a user, the web server can associate multiple requests from the same user, allowing the application to store helpful information such as each user's authentication state. The most commonly used session management mechanism is an HTTP cookie [58]. An HTTP cookie is a small piece of data sent by a web server through the "set-cookie" HTTP header, as shown in Figure 2.4.

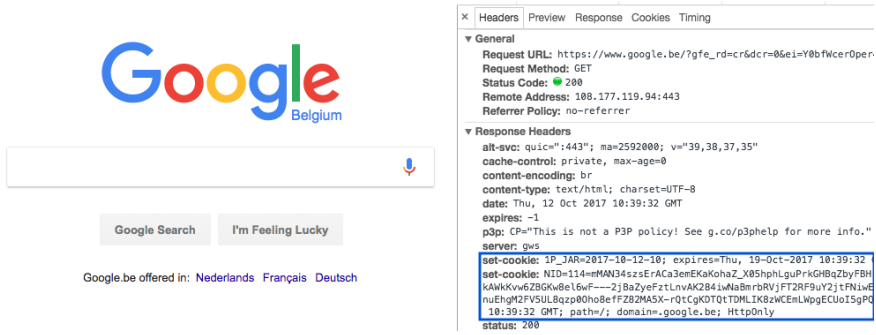

Figure 2.4: Example of cookies sent from Google Search

Browsers store cookies received from a server, and automatically attach it to every subsequent request to the same server. While cookies can be used for saving simple settings such as language preference, they are more often used to maintain an authenticated session. For this purpose, a web application creates and sends a session cookie with a unique identifier, upon a user's successful login. After that, the browser sends this cookie on subsequent requests to the server, allowing the web application to identify the user.

Since a session cookie is a user's unique identifier, any requests bearing the cookie is granted the privileges associated with the user's session. This means that the session cookie is similar to the user's credentials, as they both grant access to an authenticated session. The difference is cookies are limited to the lifespan of the current session, while credentials can be used repeatedly to

establish new sessions.

2.5 Web Security Threats

As Web applications become increasingly powerful, they have been widely adopted by various kinds of business. Meanwhile, the rise of the Web platform is also accompanied by increasing web attacks. Web attacks has caused great adverse effects on companies and individuals. Understanding these attacks can help both website owners and web users to protect themselves. But before diving into the details of web attacks, it is necessary to know the threat (attacker) behind them. Based on their capabilities, attackers can be categorized into different models, which are called threat models or attacker models. Four different threat models are often encountered in web attacks, as explained below.

- **Passive Network Attacker.** A *passive network attacker* [102] is an eavesdropper who passively monitors network traffic. The attacker can learn all unencrypted information (such as DNS requests, HTTP content) in the traffic, but cannot manipulate or spoof traffic. One common example of a passive network attack is wireless sniffing, in which a nearby attacker eavesdrops on unprotected wireless communications.
- **Active Network Attacker.** An *active network attacker* [53, 102] positions himself on a network between the web browser and the web server, and is able to intercept and tamper with the network traffic passing by. The attacker can read, modify, delete, and inject HTTP requests and responses, but he is not able to decipher encrypted information (e.g. content sent over HTTPS), nor impersonate an HTTPS endpoint without a valid certificate. An attacker can mount an active network attack in various ways, such as setting up a fake wireless hotspot ("evil twin" access point), and impersonating the user's network gateway through ARP poisoning.
- **Web Attacker.** A *Web attacker* [53, 60] is just a malicious principal who is able to register domains, obtain valid certificates for these domains, and control at least one web server to host web applications. As the web attacker model requires no special network abilities such as monitoring or intercepting traffic, every user on the Web can become a Web attacker. Thus the Web attacker is the most common threat on the Web. A typical web attacker can build a website that contains malicious code, and lure users to visit the malicious website in order to exploit the user's browser.

- **Gadget Attacker.** The *gadget attacker* [53, 60] has all the abilities of a web attacker as well as the ability to inject content ("gadget") into benign web applications. The injected content can be hyperlinks, images, and JavaScript code. In practice, this usually happens due to third-party content inclusion, where an honest website willfully integrates contents served by third-party content providers. Popular examples are JavaScript libraries, such as JQuery and Google analytics. Additionally, an attacker can also actively insert content into applications such as blog comments and forum discussions.

2.6 Common Web Attacks

Depending on the capabilities of different threat models defined in the previous section, attackers can carry out various attacks within the Web platform. This section covers five common web attacks that are relevant for the remainder of this dissertation.

2.6.1 Eavesdropping

Eavesdropping generally means secretly listening to private conversation of others without their consent. In this dissertation, eavesdropping is limited to a particular type of conversation, the network communication. By eavesdropping on network traffic, a *passive or active network attacker* can obtain sensitive data such as credit card info, username and password, and session cookies, if these supposedly secret information are sent over unencrypted channels such as HTTP.

The way to launch an eavesdropping attack depends on the type of targeted network. For wireless networks, listening traffic is straightforward, which only requires a powerful enough antenna. And there are many freely available tools for wireless sniffing. For example, Firesheep [65] is a browser extension that allows a user to sniffer session cookies in a WiFi network. Nowadays, the increasing adoption of wireless networks, especially the presence of publicly accessible WiFi hotspots, has made eavesdropping attacks extremely relevant in the modern web.

Eavesdropping on wired network is more difficult, which typically requires access to intermediaries within the network infrastructure, such as an ISP (Internet Service Provider) and a proxy server. In practice, this type of monitoring is often

carried out by government bodies. An extreme example is NSA's surveillance on Internet communication revealed by Edward Snowden [95].

The main approach to mitigate eavesdropping attacks is using secure communication channels such as HTTPS and VPN. HTTPS (HTTP over TLS (Transport Layer Security) [125]) is a commonly used technique to guarantee the confidentiality and integrity of web communication. It provides bidirectional encryption of communications between a client and server, which protects against eavesdropping and tampering with the contents. As of January 2017, more than half of the Web traffic is encrypted using HTTPS [91].

To counter wireless sniffing in a public WiFi network, end users may also use VPN (Virtual Private Network) to encrypt all the data during transmission. However, this only guarantees the confidentiality between the user agent and VPN server, an attacker with access to the VPN server or any intermediaries between the VPN server and the web server, can still monitor HTTP traffic.

2.6.2 Man-in-the-Middle attack

A man-in-the-middle attack (MITM) can be considered as an active eavesdropping attack, in which an *active network attacker* positions himself between the victim's browser and the targeted web application. The goal of a MITM attack is to be able to both monitor and manipulate the traffic. This often allows the attacker to secretly relay the communication between two parties who believe they are directly communicating with each other. In other words, the attacker can impersonate the user or the server in a man-in-the-middle attack.

MITM attacks require the attacker to be in the middle of the network, which can be achieved in various ways. For example, an attacker can control an intermediary machine in the network path, or set up a fake wireless hotspot to lure victim users to join in. Once an attacker positions himself in the middle, monitoring and manipulating non-encrypted traffic becomes straightforward. To mitigate MITM attacks, TLS protocol can be deployed. Besides offering confidentiality and integrity to prevent the attacker from reading and modifying any network traffic, TLS also provides entity authentication to authenticate the identities of involved parties.

Although TLS is mainly designed to counter MITM attacks, in reality, its actual details are often poorly understood, which frequently results in insecure deployments. Two types of insecure deployments of TLS can lead to MITM attacks. The first category is insecure design of web applications such as partial TLS coverage and mixed-content inclusion. A partial TLS coverage means a website does not force the use of TLS on all pages, which allows SSL-stripping

attacks. In an SSL-stripping attack [112], the attacker can exploit an HTTP request to a TLS-enabled website, causing the downgrade of the connection from HTTPS to HTTP. In mixed-content deployments [72], HTTP content is included by an HTTPS page, thus an attacker can manipulate the mixed HTTP content in order to compromise the HTTPS page.

The second type of insecure TLS deployments lies in the management of certificates. The entity authentication in TLS is based on X.509 certificates [79], which relies on a set of trusted third-party Certificate Authorities (CAs) to establish the authenticity of certificates. In practice, a web browser stores a list of trusted CAs, and any certificate issued by a trusted CA is inherently trusted by the browser. As a consequence, an attacker can use fraudulent but verified certificates to impersonate websites in MITM attacks. An example of such incident is an Iran attacker compromised DigiNotar, a Dutch CA, and issued a certificate for Google to conduct a MITM attack against Google services [99]. The trusted roles of CAs can even be further abused by government bodies to issue fraudulent certificates for surveillance purpose [151].

To avoid MITM attacks in the presence of TLS, web developers are recommended to follow best practices for deploying TLS [119]. For example, websites can remove mixed content and use HTTP Strict Transport Security [97] to ensure all the contents are always protected by TLS. As for the issue of fraudulent certificates, HTTP Public Key Pinning [88] can be used to specify a set of trusted public keys for a domain, and Certificate Transparency [20] can be used to audit issued certificates. Additionally, DNS-based Authentication of Named Entities (DANE) [98] can leverage the security of DNSSEC to bind certificates to domain names.

2.6.3 Cross-site scripting

Cross-Site Scripting (XSS) is a type of injection attacks, in which a *Web attacker* injects malicious scripts into otherwise benign and trusted websites. The injected code has the same privileges as the target application code, which allows the attacker to access all client-side application data such as session cookies. XSS was ranked as the most critical web application vulnerability by OWASP in 2007 [120], and had been a serious problem on the Web in the last decade. Nowadays, XSS attacks have become less relevant, but it remains one of the top 10 web security risks in 2017 [121].

Since a *Web attacker* does not have the ability to manipulate network traffic, the only way to inject content is through a web application's input fields such as HTML forms and URL parameters. To provide interactivity, modern web applications usually allow users to submit data to the applications. XSS

attacks can occur, if the submitted data are not properly checked. As an illustration, Figure 2.5 shows a simple search application that accepts inputs from users, and echos back users' queries on the page. A XSS attack occurred when the input contained scripts that were not detected by the application, as the end user's browser assumed the data came from a trusted source and simply executed the script.

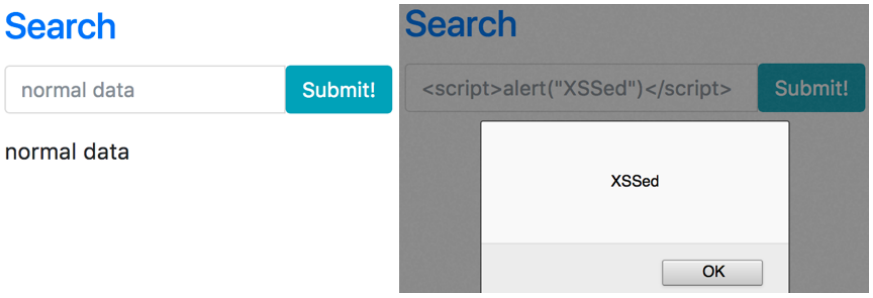

Figure 2.5: A simple search application that is vulnerable to XSS attacks

XSS attacks can be categorised into three well-known types: Stored XSS, Reflected XSS, and DOM-based XSS.

In a stored (or persistent) XSS attack, an attacker can store malicious code inside a web application, for example by hiding it in a forum post or blog comment. And when a victim's browser renders a page that includes the data provided by an attacker, the malicious code will be embedded and executed. Stored XSS typically occurs when the input data provided by the attacker is saved on a server-side database, but with the advent of client-side storage in HTML5, the attack payload can be permanently stored in the victim's browser as well.

Reflected (or non-persistent) XSS attacks are those where the injected code is not stored, but immediately reflected in a server's response. To launch a reflected XSS attack, an attacker typically hides malicious code in a link, and trick a victim to click the malicious link. The victim's browser then executes the attacker's code because it thinks the script came from a trusted server. The example illustrated in Figure 2.5 is a reflected XSS, where the malicious link has a form like this: `http://example.com/search?query=%3Cscript%3Ealert(%22XSSed%22)%3C%2Fscript%3E`

A third class of XSS attacks is DOM-based XSS [105], wherein an attacker stores malicious code into the DOM of a webpage that is visited by a victim user. In a DOM-based XSS attack, the injected data is not processed by server-side

logic of the web application, but by its client-side logic, thus the entire data flow never leaves the browser. The injected code is executed as a result of modifying the DOM environment used by original client side script.

Since the problem of an XSS attack is essentially the failure of a web application to recognize the insertion of code, the general approach to prevent XSS attacks is sanitization: applying filters to untrusted data provided by the users. Sanitization techniques typically replace or remove dangerous characters such as `< > & "` or check against a whitelist of allowed characters. Depending on the application context, several publicly available libraries can be used to sanitize data. For example, OWASP Java Encoder Project [30] is a sanitizer for Java applications, and HTML Purifier [16] offers automatic sanitization for PHP applications.

Besides sanitization, another relatively new approach to mitigate XSS is Content Security Policy (CSP) [148]. A Web application can use CSP to instruct a browser to disallow the execution of inline scripts and only accept scripts from specified trusted sources. In addition to sanitization and CSP, major modern web browsers such as Google Chrome, Internet Explorer, and Mozilla Firefox, also provide client-side XSS protection to detect and stop certain reflected XSS attacks.

2.6.4 Session hijacking

Session hijacking, also known as cookie hijacking, refers to the exploitation of an authenticated session to gain unauthorized access to a web application. Web applications usually generate and send HTTP cookies to a user's browser, in order to maintain a session with the user. In a session hijacking attack, an attacker steals the user's session cookies and then transfers an authenticated session from the victim's browser to the attacker's browser. In this way, the attacker can impersonate the victim user, performing actions in the name of the user in a web application.

A cookie hijacking attack is possible because a session cookie is a user's unique identifier, any requests bearing the cookie is granted the privileges associated with the user's session. To steal a user's cookies, an attacker can eavesdrop on the network traffic, as illustrated in Figure 2.6.

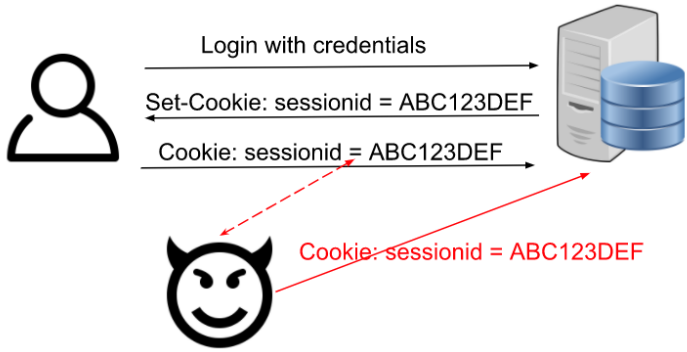

Figure 2.6: A session hijacking attack via eavesdropping

Besides eavesdropping or MITM attacks, an attacker can also obtain session cookies via XSS attacks. This is because HTTP cookies are stored in the user browser, and can be accessed by JavaScript through the `document.cookie` property of current page. Additionally, an attacker can simply guess the session cookie if it is generated in predictable patterns.

There are two different types of mitigation techniques for session hijacking. One is to limit the capability of a session cookie, making it no longer a bearer token for an authenticated session, thus even if an attacker has successfully obtained cookies, he cannot use them to transfer sessions. This can be achieved by binding a user's session to some specific properties such as IP address, or browser fingerprint [117]. When a web application detects changes of these properties, it can abolish the current session and request a re-authentication. Another way to limit a cookie's capability is using one-time disposable tokens [81].

The second type of mitigation techniques focuses on preventing cookie stealing. The HttpOnly and Secure cookie flags can be used respectively to ensure a cookie is not accessible through JavaScript, and is always transmitted over TLS. Correctly applying both attributes can effectively prevent the theft of session cookies via XSS and eavesdropping attacks.

2.6.5 Clickjacking

Clickjacking, also known as UI (User Interface) redressing attack, is a malicious technique of tricking a user into clicking on something different from what the user perceives they are clicking on. In a clickjacking attack, an attacker creates a malicious page that includes a frame page from a target application.

The targeted page is made transparent and hidden beneath the attacker's page. When a victim user clicks on a button or link on the malicious page, he is unintentionally interacting with the target application, as illustrated in Figure 2.7.

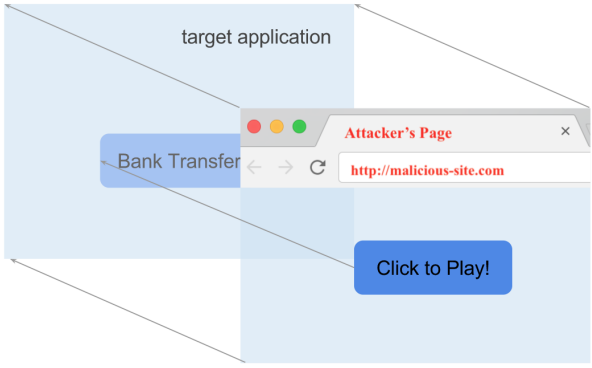

Figure 2.7: An illustration of a clickjacking attack

In essence, a clickjacking attack allows an attacker to forge a request from the victim's browser to the target application. The forged requests can be used to achieve various goals, such as getting Facebook likes (making users like a Facebook page) or Twitter followers (making users follow someone on Twitter); tricking users into downloading malware or clicking ads.

Website owners can protect their users against clickjacking attacks by using an HTTP header `X-Frame-Options` (XFO) [129] to disallow a webpage being framed in external web applications. This feature has been recently integrated in the Content Security Policy (CSP) with the frame-ancestors directive [148], thus CSP can also be used to achieve the same goal.

Besides server-side defences, a user can install browser add-ons such as NoScript [28], which includes a ClearClick feature to prevent users from clicking on invisible page elements of embedded documents.

2.7 Conclusion

The Web is a double-edged sword. On one hand, it has greatly enriched people's lives and helped businesses to thrive. On the other hand, it can be abused

by cybercriminals, putting individuals and companies at risk. Understanding web technologies and web attacks helps both website owners and web users to protect themselves.

Many web attacks can be avoided (or at least be alleviated), if web users are aware of security risks and websites owners manage the risks properly. As part of risk management, websites owners can implement known security mechanisms to prevent common attacks, and regularly assess their security posture. This will be discussed in the forthcoming chapter.

Chapter 3

Assessing Web Security

As the Web rapidly expands and gets deeply integrated into our society, it is important to ensure the security of Web applications. While individual users can learn to avoid some web attacks through security awareness education, the main responsibility of offering web protection should rest on the shoulder of website owners.

In general, business owners have three main motivations to ensure their websites' security: (1) to defend critical data assets from attacks. An unprotected website might lead to severe data breaches; (2) to build trust with customers. A well-protected website makes its user feel being cared and concerned; (3) to comply with legal requirements. Some countries have strict data protection law (e.g., the General Data Protection Regulation in European Union [49]), which requires websites to protect individuals' private data.

Web attacks are constantly evolving, and there are no silver bullets. To protect web applications, website owners can adopt a risk-based approach, i.e., to treat cyber attacks as always-present risks and manage the risks properly. As part of their risk management, websites owners can implement known security mechanisms, and regularly assess their security posture. While this cannot completely avoid attacks, it minimises attack surface and alleviates the consequences when attacks occur. Moreover, since some security mechanisms are visible from the client side, they provide reassurance to customers.

This chapter introduces several security mechanisms that websites owners can adopt to protect their web applications. These mechanisms are detectable from the client side, and can be used as metrics to assess Web security. Such an assessment can be carried out in an automated way and on a large scale, which

might be desirable to government and supervisory organizations to continuously monitor the security of the Web environment. To efficiently conduct such a large-scale assessment, a web crawling approach was proposed in this chapter.

3.1 Client-side Security Mechanisms

This section introduces some security mechanisms that are proposed to mitigate the common attacks in Section 2.6. While these security mechanisms are implemented by a web server, they require a client (conformant browser) to enforce it, thus they are referred as client-side security mechanisms/features in this dissertation.

3.1.1 HTTPS Support

The HTTPS [125] protocol is the standard solution for securing web traffic, which can thwart the eavesdropping and MITM attacks. It guarantees the confidentiality and integrity of web communications by adding the security capabilities of SSL/TLS to HTTP. The original SSL (Secure Sockets Layer) protocol was developed by Netscape for its Netscape Navigator browser in 1994. In 1996, the Internet Engineering Task Force (IETF), an Internet standards organization, took over the responsibility for the protocol. IETF later renamed SSL to TLS (Transport Layer Security) which was formally specified in [82].

When a web application uses HTTPS, the connection between a web browser and a web server is secured by TLS. A message transmitted over HTTPS is encrypted and its integrity is validated with a message authentication code. In addition to the confidentiality and integrity, TLS can also authenticate the identity of the communicating parties by using public-key cryptography. In practice, HTTPS only provides website authenticity with the CA/B (Certificate Authority/Browser) trust model, while the end user is authenticated with other mechanisms.

Since HTTP provides no security guarantees, website operators are strongly recommended to adopt HTTPS. While HTTPS has long existed as a standard solution, the adoption of it has been slow. Besides the performance overhead of SSL/TLS, the complexity of HTTPS deployment has also deterred its adoption. However, these concerns have been largely addressed by recent improvements such as the support of HTTP/2 [61] and the free SSL/TLS certificates provided by Let's Encrypt [23]. And web browsers are holding non-secure sites more accountable, for example, Google Chrome (from version 56) displays a "Not

Secure" warning for pages served over HTTP. These efforts have speeded up the HTTPS adoption in recent years.

To correctly deploy HTTPS, web developers can follow the best practices recommended by OWASP [119].

3.1.2 HTTP Strict-Transport-Security

HTTP Strict Transport Security (HSTS) [97] is a web security policy mechanism which helps websites to prevent SSL-stripping attacks [112]. The HSTS Policy is sent by the server to the browser via an HTTPS response header field named `Strict-Transport-Security`. It specifies a period of time during which the user's browser is instructed that all requests to that website need to be sent over HTTPS, regardless of what a user requests.

To implement an HSTS policy, a web server supplies a `Strict-Transport-Security` header over an HTTPS connection (HSTS headers sent over HTTP are ignored). The header sets a max-age (specified in seconds) during which a browser can only interact with the website by using secure HTTPS connections. For example, `Strict-Transport-Security: max-age=3600` instructs a browser to use only HTTPS for future requests for an hour (3600 seconds). The HSTS header can also specifies an optional parameter `includeSubDomains` to include all of the site's subdomains as well.

When a conformant browser receives an HSTS header from a website, it will automatically turn any insecure HTTP links to that website into secure HTTPS links during the specified time frame. The browser will update the expiration time whenever it receives an HSTS header, so web applications can prevent the timeout from expiring by always sending HSTS headers. Additionally, modern browsers typically maintain an "HSTS preloaded list", which is a list of known HSTS-enabled sites that are hardcoded into the browser as being HTTPS only. Google Chrome also offers an "HSTS Preload List Submission" service [15], which allows a website to use `preload` parameter in an HSTS header to submit itself to Chrome's "HSTS preloaded list".

3.1.3 Public Key Pinning and Certificate Transparency

Public Key Pinning and Certificate Transparency are two approaches that aim to address the issue of fraudulent SSL/TLS certificates used in MITM attacks.

Public Key Pinning allows websites to specify trusted public keys in an HTTP response header named `Public-Key-Pins` [88]. It tells a web browser to

associate a set of specific cryptographic public keys with a certain website for a given time. During that validity time, the browser only accepts a server with one or more of those specified public keys. Thus, even if an attacker compromises a CA to forge a certificate, he cannot use the forged certificate to impersonate the website's server.

The idea of public key pinning was originally started at Google in 2011, as an effort to protect its web services from MITM attacks [45]. The approach was called *static pinning*, in which Google hardcoded some whitelisted public keys in Chrome. Google later expanded the idea into *dynamic pinning*, which was standardized as Public Key Pinning Extension for HTTP (HPKP) in 2015 [88]. However, it turned out that HPKP was difficult to implement and maintain [48], and it can be abused by attackers [50].

There are two main security issues with HPKP: HPKP Suicide and RansomPKP. HPKP Suicide refers to the situation where an HPKP-enabled website loses the pinned public keys. The keys might be accidentally deleted, or stolen in a hack incident. Consequently, browsers that have stored the site's HPKP policy will not be able to connect to the site until the HPKP policy expires. The second issue, RansomPKP, happens in a web server breach scenario. When an attacked gains control of the server, he can set malicious HPKP headers such as pinning the attacker's keys. It will take a lot time and effort for a website to recover if its HPKP is abused.

With these issues exist, and no support from other browsers (IE/Edge and Safari), Google Chrome plans to abandon HPKP in 2018 [52], and turns to another approach called Certificate Transparency. Certificate Transparency (CT) [20] is an open framework for monitoring and auditing SSL/TLS certificates in nearly real time. It requires certificate authorities to publish all issued certificates in public CT Logs. This allows quick detection of any mis-issued certificates.

A website can use an HTTP response header called `Expect-CT` to enforce Certificate Transparency. The `Expect-CT` header instructs a browser to expect a valid Signed Certificate Timestamps (SCTs) to be served when connecting to the website. By combining `Expect-CT` with active monitoring of CT logs, website operators can pro-actively detect fraudulent SSL/TLS certificates. Certificate Transparency and `Expect-CT` are still under the drafting process by IETF, and there is no support from other browsers as of December 2017. But starting from April 2018, Google Chrome requires Certificate Transparency for all newly issued, publicly trusted certificates.

Since Public Key Pinning (HPKP) is phasing out, and Certificate Transparency (CT) is proposed to achieve similar goals, these two approaches (HPKP/CT) are being treated as a single security feature in this dissertation.

3.1.4 HttpOnly and Secure Cookies

HttpOnly and Secure Cookies are cookies set with the `HttpOnly` and `Secure` attributes. These two flags can be used to protect session cookies and prevent session hijacking.

First introduced in Internet Explorer 6 SP1 in 2002, the `HttpOnly` attribute is designed to mitigate the risk of malicious client-side scripts accessing sensitive cookie values. Cookies are accessible to JavaScript code by default, which allows attackers to steal the cookies via an XSS attack. Using the `HttpOnly` attribute in a `Set-Cookie` header restrict the access of that cookie to the HTTP(S) protocol, making it inaccessible to client-side JavaScript [58].

The purpose of the `Secure` flag is to prevent cookies from being observed by unauthorized parties due to the transmission of a cookie in clear text. Although the traffic between a web server and a browser is encrypted when using HTTPS, the cookies stored in the browser are not, by default, limited to an HTTPS context. Thus an active network attacker can intercept any outbound HTTP request from the browser and redirect that request to the same website over HTTP in order to reveal the cookies [58]. By setting the `Secure` attribute, the scope of a cookie is limited to secure channels, thus stopping browsers from sending cookies over unencrypted HTTP requests.

The `HttpOnly` and `Secure` attributes are set via a `Set-Cookie` HTTP response header. For example, this header `Set-Cookie: OSID=5wRhE...; path=/; domain=mail.google.com; Secure; HttpOnly` sets a cookie named `OSID` with both flags enabled. A browser records this information as part of cookie data in its storage system, as shown in Figure 3.1.

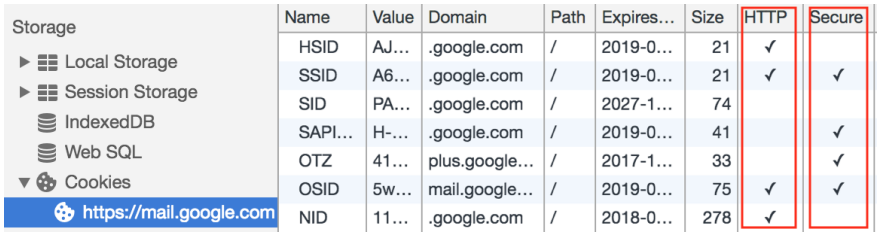

Figure 3.1: Example of HttpOnly and Secure Cookies stored in a browser

3.1.5 Content Type Options

When a web server sends a resource to a browser, it can use the `Content-Type` header to indicate the media type of the resource. The content type is specified as a MIME (Multipurpose Internet Mail Extensions) type, which is a way to identify different Internet resources. If a server does not provide the `Content-Type` header or the specified MIME type is ambiguous, some browsers such as Internet Explorer will use a detection algorithm to determine the content type, which is called MIME sniffing. However, this MIME-sniffing feature can be abused by attackers to disguise a particular file type as something else, which might give them the opportunity to perform cross-site scripting attacks.

In order to disable MIME sniffing, thus reducing exposure to attacks, Microsoft introduced the `X-Content-Type-Options` (XCTO) header [18]. A website can send this header with "nosniff" value (`X-Content-Type-Options: nosniff`) to tell a browser not to sniff MIME type and instead follow the value specified in the `Content-Type` header. The `X-Content-Type-Options` header is supported by most browsers including Chrome, Firefox, IE/Edge.

3.1.6 Content Security Policy

Content Security Policy (CSP) [148] is a security policy that helps to mitigate several types of attacks, including cross-site scripting (XSS), clickjacking and data injection attacks. CSP provides a standard method for website owners to declare approved origins of content that browsers should be allowed to load on that website. It can be used to cover many different type of web resources, such as JavaScript, CSS, images, HTML frames, audio and video files, and other embeddable objects.

CSP was originally proposed and implemented by Mozilla Firefox in 2010 [137], in order to help websites to prevent XSS attacks. It was quickly adopted by other web browsers and has been published as a W3C recommendation in 2014 [148]. As of 2017, a new version of CSP is being developed under W3C to include more features [147].

To enable CSP, a web server can send the `Content-Security-Policy` header. Alternatively, a CSP policy can also be specified in a `<meta>` element. The primary use of CSP is to mitigate XSS attacks. To achieve this goal, a website can send a CSP header that looks like Listing 3.1.

Listing 3.1: Preventing XSS with a CSP policy

```
1 Content-Security-Policy: default-src 'self'; img-src *; \
2 script-src trusted.example.com;
```

In the above example, the policy specifies that 1) the default trusted origin is the website itself (via `default-src` directive); 2) images can be loaded from anywhere (via `img-src` directive); 3) executable script is only allowed from trusted.example.com (via `script-src` directive). Hence, whenever a requested resource originates from a source that is not defined in the CSP, it will simply not be loaded. For example, even if an attacker is able to inject malicious JavaScript in the webpage, the injected code will not be executed, as it is not from the specified trusted origin.

With more than 20 different directives, CSP is very versatile and flexible. Besides specifying trusted origin for various content types, a website can also use CSP to guarantee secure communication. For example, the `upgrade-insecure-requests` directive instructs a browser to upgrade HTTP links to HTTPS links; the `block-all-mixed-content` directive prevents loading HTTP-served content in an HTTPS page.

3.1.7 X-Frame-Options

X-Frame-Options (XFO) [129] is an HTTP response header designed to mitigate Clickjacking attacks. In a Clickjacking attack, the attacker redresses the user interface of website A with transparent layers, and then trick the user into clicking on a button on an embed page from website B when they were intending to click on the same place of the overlaying page from website A.

To stop Clickjacking attacks, a website can use the `X-Frame-Options` header to tell a browser whether a certain page is allowed to be embedded in a frame. There are three possible directives for this header: `DENY`, `SAMEORIGIN`, and `ALLOW-FROM`. If the header is set to `DENY`, then the browser will prevent the page from rendering when embedded within a frame. On the other hand, if `SAMEORIGIN` directive is specified, then the page is only allowed to be embedded in other pages from the same domain. The `ALLOW-FROM` directive is used to specify a trusted origin that can embed the page.

The function of `X-Frame-Options` has been integrated in the CSP version 2 [148] with the `frame-ancestors` directive. Thus a website can also use CSP to prevent clickjacking attacks. Setting `frame-ancestors: none` in CSP has the same effect as `X-Frame-Options: DENY`. And `frame-ancestors: self` is similar to `X-Frame-Options: SAMEORIGIN`.

3.2 Large-scale Web Crawling Approach

This section describes a web crawling approach that is used for several large-scale web assessments presented in the forthcoming chapters of this dissertation. The approach can be adopted by an outside assessor to analyse websites security in an efficient fashion.

We begin with a set of websites to be assessed. Since the analysis is based on the security features that can be found in webpages, we first need to find enough webpages for each given website. To achieve this, the Bing Web Search API [8] is used to obtain popular webpages from a domain. The popular webpages found by Bing Search is a good representative of a website, as people typically enter a website through search engines. The presence or absence of defensive mechanisms on these popular webpages can largely reflect the website's security.

More specifically, we use the `site:domain` operator with Bing Web Search API to obtain a set of popular page URLs for a domain. For instance, a single search for `site:facebook.com` in Bing will return a set of 50 webpages belonging to `facebook.com`. Ideally, we should get as many pages as possible for a website, since a larger sample size is more representative. However, modern websites often serve dynamic pages, generating webpages based on the parameters in an URL, which results in some very correlated page URLs. For example, although `http://example.com?id=1` and `http://example.com?id=2` are different, they are produced by the same server-side logic, thus having the same security features. In other words, the representativeness is not fully determined by the size of URLs set.

In our experiment, we typically obtain up to 200 page URLs for a website, in order to have a reasonable representative sample size. Optionally, one can measure the similarity between URLs to filter out correlated URLs, which can reduce the sample size yet retain the same level of representativeness, making the lateral crawling phase faster.

After page URLs of all websites are obtained, the second phase is to visit these pages with a crawler. To avoid hurting the performance of websites, the URLs of all websites are randomly shuffled before being feeded to the crawlers. The crawlers are built on top of a headless scriptable browser, such as HtmlUnit [17] and PhantomJS [31]. By loading webpages in a headless browser with an appropriately set user-agent, we mimicked the behavior of a regular user visiting a website with a normal browser.

To efficiently crawl millions of webpages within a reasonable period of time, we distribute the work across multiple threads or machines. Celery [11], a Python library, is used to manage such a distributed task queue, where it uses a broker

such as RabbitMQ [33] to accept crawling tasks. A crawling task is simply using a crawler to visit a webpage, and receive HTTP responses. The results obtained by crawlers are all stored into a database (MongoDB [26]), which can be accessed by an analyser program for data processing or a visualizer program for data visualization.

The steps of the web crawling approach is illustrated in Figure 3.2. Using this approach, we can crawl 1 million webpages within 2 days, with 50 networked machines.

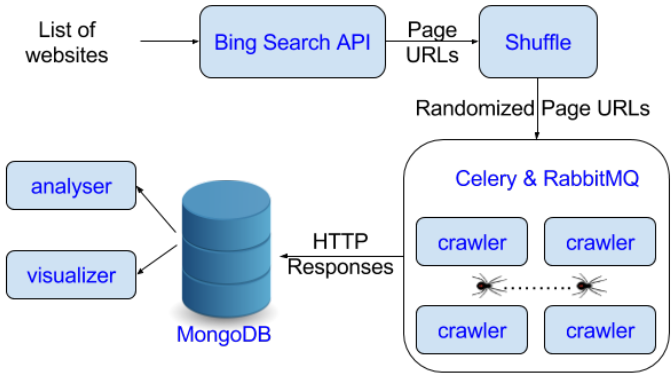

Figure 3.2: Large-scale web crawling approach

Chapter 4

Large-scale Analysis of Mixed-content Inclusion

Preamble

The contents of this chapter are adapted from the paper titled "A Dangerous Mix: Large-scale analysis of mixed-content websites" [72], which was published in the Proceedings of the 16th Information Security Conference in 2013. This work was done with the collaboration of other authors from DistriNet, KU Leuven. Ping Chen was the lead author of this paper.

This chapter investigates the issue of mixed-content inclusion, using the web crawling approach introduced in the previous chapter. Our work sheds light on the prevalence and impact of mixed HTTP content on HTTPS webpages.

4.1 Introduction

The World Wide Web is built on the HTTP protocol, which does not provide security by default. To ensure secure communication of sensitive data, websites can deploy SSL/TLS (Secure Socket Layer or Transport Layer Security) protocol. In this way, the plain HTTP becomes HTTPS (HTTP over SSL/TLS). HTTPS can protect web applications against passive network attackers, i.e. eavesdroppers, that steal sensitive data and credentials from HTTP requests and responses passing by over the network, as well as against active network

attackers tampering with the content of these requests and responses. It has become the standard solution for securing web communication.

While websites are migrating to HTTPS, attackers are also shifting efforts to break the HTTPS communication. A variety of attacks against SSL/TLS have been found. Some of them are targeting cryptographic weaknesses and design flaw in the protocol, such as BEAST [83], CRIME [128] and POODLE [114]; some of them are exploiting implementation errors in popular cryptographic software libraries, such as the Heartbleed [84] vulnerability in openSSL. Besides exploiting these protocol vulnerabilities, attackers can also forge fake SSL/TLS certificates to perform man-in-the-middle (MITM) attacks, as illustrated in the DigiNotar [99] incident.

Complementary to protocol vulnerabilities and infrastructure flaw in HTTPS, mixed-content inclusion can also be used to compromise TLS-enabled websites. In mixed-content websites, the webpage is delivered to the browser over HTTPS, but some of the additional content, such as images and scripts, are delivered over a non-secured HTTP connection. These non-secured communications can be exploited by an active network attackers to gain access to wide set of capabilities ranging from access to cookies and the forging of arbitrary requests, to the execution of arbitrary JavaScript code in the security context of the TLS-protected website.

Desktop browsers are recently catching up to mitigate this vulnerability, but the large majority of browsers on mobile devices, such as smartphones and tablets, leave the end-user unprotected against this type of attack. This is worsened by the fact that it is typically pretty straightforward to launch an active network attack against a mobile user (e.g. via open Wi-Fi networks, or setting up a fake wireless hotspot).

In this chapter, we report on an in-depth assessment of the state-of-practice with respect to mixed-content vulnerabilities. In particular, the main contributions of this chapter are the following:

- We study the different types of mixed-content inclusions, and assess their security impact.
- We present a detailed analysis of mixed-content inclusions over the Alexa top 100,000 Internet domains, showing that 43% of the Internet's most popular websites suffer from mixed-content vulnerabilities.
- We document the behaviour of mobile browsers in the face of mixed-content inclusions.

- We enumerate the best practices as well as novel mitigation techniques against mixed-content inclusions for browsers, website owners and content providers.

4.2 Problem Statement

It is well-known that HTTP is vulnerable to eavesdropping and man-in-the-middle (MITM) attacks, and HTTPS is designed to precisely prevent these attacks by adding the security capabilities of SSL/TLS to HTTP. SSL/TSL enables authentication of the web server, and provides bidirectional encryption of the communication channel between the client and server. While web applications are increasingly adopt HTTPS for protection, it is important to avoid mixed-content inclusion that might render the HTTPS protocol useless.

In a mixed-content (also known as non-secure/insecure content) website, the webpage is delivered to the browser over TLS, but some of the additional content, such as images and scripts, are directly delivered over a non-secured HTTP connection from the content provider towards the web browser. Considering an *active network attacker*, who positions himself on a network between the web browser and the web server. Although the attacker can not intercept the webpage delivered over HTTPS, he can read and modify any HTTP resources on that page, which might lead to the compromise of a TLS-enabled website, as shown in Figure 4.1.

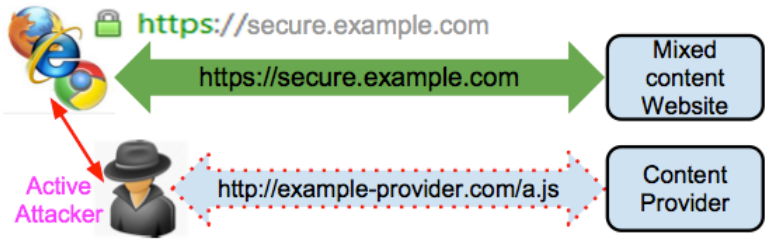

Figure 4.1: Mixed-content vulnerability

An attacker can mount such an active network attack in various ways, such as setting up a fake wireless hotspot (aka “evil twin” access point), sniffing and spoofing network frames on an existing Wi-Fi network, or by impersonating the user’s DNS server or gateway.

4.3 Impact of Mixed Content Attacks

When mixed content passes by an active network attacker, he can launch various attacks against the content provider, the TLS-protected website including the mixed content, and the user's browser. The impact of a mixed content attack can be categorised as follows:

- **Cookie stealing:** When a browser requests mixed content, it may include cookies associated with the content provider, which allows the attacker to obtain the cookies. Moreover, if the content provider and the TLS-protected website using mixed content happen to be on the same domain, sensitive cookies used over HTTPS can get exposed to the attacker via a HTTP request, unless the cookie is protected by the "secure" flag.
- **Request forgery:** As mixed content is requested over HTTP, the attacker can manipulate the HTTP requests and responses and use them to trigger or forge arbitrary HTTP requests, which may lead to certain variants of SSL-Stripping [112] and Cross-Site Request Forgery (CSRF) [59].
- **DOM data leakage:** Mixed content may leak confidential data that is displayed as part of the HTTPS webpage. For example, the attacker can manipulate mixed-CSS content to obtain sensitive data in DOM via scriptless attacks [96, 152]: CSS selectors can match against particular content in the DOM, and leak the result of the test by fetching a web resource (e.g. image) monitored by the attacker.
- **JavaScript execution:** For some types of mixed content, the attacker can inject arbitrary JavaScript code that will be executed in the context of the HTTPS website using the mixed content. This allows the attacker to run arbitrary JavaScript code as if it was originating from the TLS-protected site, and access a variety of security-sensitive JavaScript APIs. Moreover, the attacker can inject malicious payloads, such as the BeEF framework [7], to take over the user's browser and even exploit the vulnerabilities in the browser and underlying OS.

The impact of a mixed content attack depends on the content resource. Five specific types of mixed content are studied in this chapter: Image, iframe, CSS, JavaScript and Flash.

- **Image:** Though mixed-Image content, is considered passive content and thus less harmful, it can still be used to forge HTTP requests, and it may also reveal cookies. Additionally, an attacker may change the responses to

serve shocking images in order to harm the reputation of the vulnerable site.

- **iframe:** Similar to mixed-Image content, mixed-iframe content may lead to request forgery and cookie stealing attacks. Because of the same origin policy, an iframe cannot break out of its own frame and thus the attack does not give the attacker many more possibilities than mixed-Image content, though it may lead to some attacks in older versions of vulnerable browsers [19].
- **CSS:** Besides the aforementioned capabilities of mixed-Image content which are equally present in mixed-CSS content, an attacker can also use CSS content to obtain sensitive data from the DOM via scriptless attacks [96].
- **JavaScript:** With the very rich set of JavaScript APIs available on the web, the attacker can launch all the aforementioned attacks by intercepting the mixed-JavaScript content and replacing it with his own malicious code.
- **Flash:** Mixed-Flash content gives the attacker the same power as mixed-JavaScript content, since it can also be used to execute arbitrary JavaScript in the context of an HTTPS webpage, for example, via ActionScript's `getURL("javascript:myfunction();");`).

The various types of mixed content and their impact are summarised in Table 4.1.

Content type	Cookie stealing	Request forgery	DOM data leakage	JavaScript execution
Image	x	x		
iframe	x	x		
CSS	x	x	x	
JavaScript	x	x	x	x
Flash	x	x	x	x

Table 4.1: Impact of mixed content attacks

4.4 Data Collection

To assess the prevalence mixed-content inclusion on the Web, we did a large-scale data collection experiment. The main goal of this experiment is to assess the

state-of-practice of mixed-content usage on the most popular Internet websites. More specifically, we aim to (1) identify the distribution of mixed-content inclusions over different types of mixed content, as well as over local and remote inclusions; (2) investigate the availability of mixed content over HTTPS.

Starting with Alexa's list of 100,000 most popular domains, we first select the TLS-enabled websites by sending a single HTTPS request to each domain. With the obtained list of HTTPS websites, we then crawled these domains, by using the web crawling setup and methodology introduced in Section 3.2.

As a result, we extracted 18,526 HTTPS websites from Alexa top 100,000 Internet domains, and in total 481,656 HTTPS pages are crawled, with an average of 26 HTTPS pages per website.

From the crawled HTTPS websites, 7,980 (43%) were found to have at least one type of mixed content. This means that almost half of the HTTPS protected websites, are vulnerable to one or more of the attacks mentioned in the previous sections. In total, 620,151 mixed-content inclusions were found through our experiment, which maps to 191,456 mixed-content files and 74,946 HTTPS webpages.

Table 4.2 gives an overview of the distribution of mixed-content inclusions. Image and JavaScript are the most included mixed content types, with 30% and 26% of the HTTPS websites using them respectively, while mixed-Flash content is much less used. As for the distribution over remote and local inclusions for each mixed content type, mixed iframe, JavaScript, and Flash content is mostly served by remote providers, while the majority of mixed Image and CSS inclusions are locally included.

	# Inclusions	% remote inclusions	# Files	# Pages	% Websites
Image	406,932	38%	138,959	45,417	30%
iframe	25,362	90%	15,227	15,419	14%
CSS	35,957	44%	6,680	15,911	12%
JavaScript	150,179	72%	29,952	45,059	26%
Flash	1,721	62%	638	1,474	2%
Total	620,151	47%	191,456	74,946	43%

Table 4.2: Overview of distribution of mixed-content inclusions

To better understand the risks associated with websites using different types of mixed content, we calculate the percentage of websites that are exposed to different levels of attacks as shown in Figure 4.2. The calculation is based on the impact analysis for each mixed content type (Table 4.1), which groups

different types of mixed-content inclusions according to the associated attacks. Figure 4.2 shows that 27% websites are exposed to attacks up to "JavaScript execution", by including mixed JavaScript or Flash content.

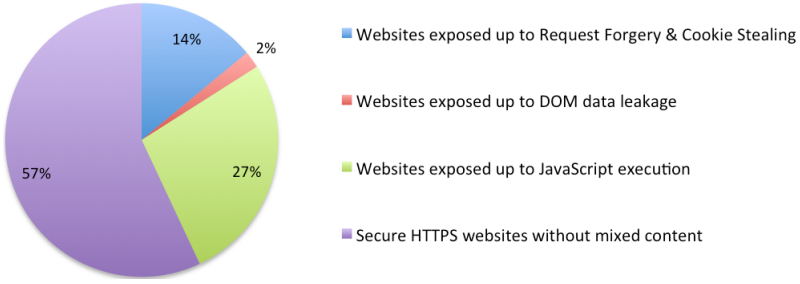

Figure 4.2: Percentage of HTTPS websites vulnerable to mixed-content attacks

4.5 Discussion

In this section, we discuss some characteristics of mixed content and the websites including them, as discovered in our experiment. First, we identify the distribution of websites having mixed content over Alexa rank ranges and different site categories, and then present some examples of important websites having mixed-JavaScript content. Second, we investigate the availability of mixed content files over HTTPS, and present the some of the most popular mixed-JavaScript content discovered. We mainly focus on mixed-JavaScript inclusions when giving examples, since (1) the number of mixed-JavaScript inclusions and websites including mixed-JavaScript content is large enough to fully represent the mixed content inclusion problem, and (2) mixed-JavaScript content is more dangerous than the other types of mixed content.

4.5.1 Websites having mixed content

The 18,526 HTTPS websites assessed in our experiment are roughly equal distributed across Alexa's rank ranges (per 10 thousand), as indicated by the blue bar in Figure 4.3. And the percentage of sites having mixed content in each rank range (the red line) fluctuates around 43%, which is the total average number. This implies that there is no correlation between a website's popularity and its probability of having mixed content. Websites with high Alexa ranks might also be vulnerable to mixed content attacks.

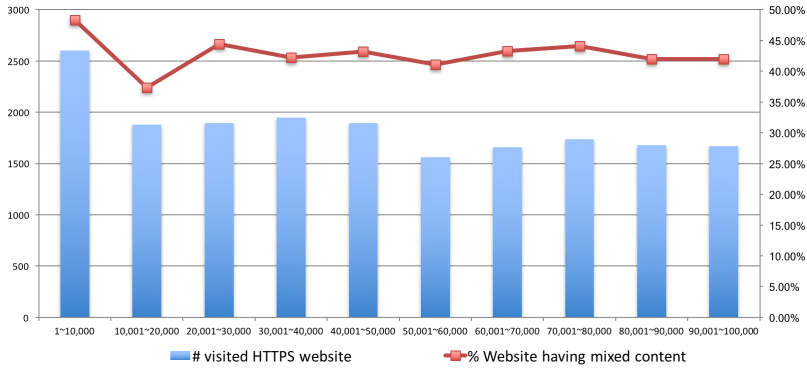

Figure 4.3: Distribution of websites having mixed content over Alexa ranks

To better understand the websites having mixed content, we also categorize the websites by checking McAfee's TrustedSource Web Database [24], comprised of 104 site categories. Most of the visited HTTPS websites are categorized into 88 categories, with 1,181 websites remaining uncategorized. The majority of visited websites (66%) can be categorized into 10 popular categories.

The number of websites and the percentage of websites having mixed content over the top 10 categories are presented in Figure 4.4. The 'Government/Military' websites are doing better than websites in all other categories, with "only" 31% of them websites having mixed content. 38% of 'Finance/Banking' websites having mixed content, which is worrisome, since these websites contain valuable information and are typically the targets of attackers.

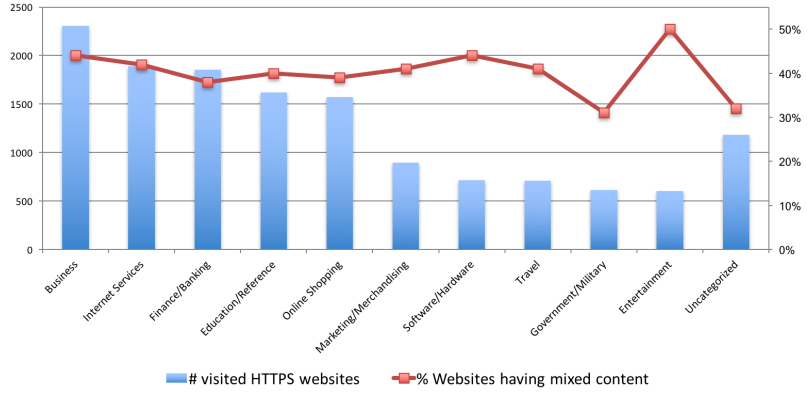

Figure 4.4: Distribution of websites having mixed content over top 10 categories

For the 74,946 HTTPS pages having mixed content, we check whether these pages have an equivalent HTTP version of the same content. While most of them do have an HTTP version, 9,792 (11%) pages are only served over HTTPS, and these "HTTPS-Only" pages map to 1,678 (9%) HTTPS websites. We consider it likely that these "HTTPS-Only" pages contain more sensitive data and should be more secure, compared to those pages having the same content served over HTTP. Thus, mixed-content inclusions on "HTTPS-Only" pages can have more severe consequences when successfully exploited.

Table 4.3 lists ten examples of "HTTPS-Only" pages (selected from Alexa's top 1,000 websites) having mixed-JavaScript content. These pages provide important functionalities like "Account Signup" , "Account Login" , and "Password Recovery", all of which process sensitive user information and thus can lead to user-data leakage if the mixed-JavaScript content is intercepted by an attacker.

HTTPS-Only pages	Functionality
`www.aweber.com/signup.htm`	Account Signup
`www36.verizon.com/callassistant/signin.aspx`	Account Login
`secure.pornhublive.com/forgot-password/`	Password Recovery
`euw.leagueoflegends.com/account/recovery/password`	Password Recovery
`ww15.itau.com.br/privatebank/contatoprivate/en/index.aspx`	Contact Form
`dv.secure.force.com/applyonline/Page1?brand=ccn`	Application Form
`www.tribalfusion.com/adapp/forms/contactForm.jsp`	Contact Form
`support.makemytrip.com/ForgotPassword.aspx`	Password Recovery
`jdagccc.custhelp.com/app/utils/create_account/red/1`	Account Signup
`ssl6.ovh.net/~pasfacil/boutiquemedievale/login.php`	Account Login

Table 4.3: Ten example "HTTPS-Only" pages having mixed-JavaScript content

Of the 1,678 HTTPS websites that have "HTTPS-Only" pages, we found 97 websites that are using HTTP Strict Transport Security (HSTS) policy [97], which indicates that these websites are making use of the latest protection technology for ensuring the use of SSL, but they still fail to achieve their goal by including mixed content from insecure channels.

4.5.2 Providers of mixed-content files

For the total of 191,456 mixed-content files, we check whether the providers serve these files over a secure HTTPS channel next to their insecure HTTP versions. While the majority of mixed JavaScript, iframe and CSS content files are available over HTTPS, the percentage of mixed Image content files available

over HTTPS is significantly less, as shown in Table 4.4. Though website owners should be responsible for the mixed content issue, the data in Table 4.4 indicates that blaming them is too simplistic, since it ignores the fact that approximately half of the mixed content files are only available over HTTP.

Type	# Files	% HTTPS-Available	Type	# Files	% HTTPS-Available
Image	138,959	40%	JavaScript	29,952	58%
iframe	15,227	77%	Flash	638	46%
CSS	6,680	60%	Total	191,456	47%

Table 4.4: Percentage of "HTTPS-Available" files, per mixed content type

Table 4.5 presents the top ten providers of mixed-JavaScript content and the number of websites using them. It shows that nine out of ten of the most frequently used content providers offer their JavaScript content also over HTTPS, and hence about 28% of the mixed-JavaScript inclusion can be avoided if the websites correctly include JavaScript coming from these nine providers.

Mixed-JavaScript content providers	Available over HTTPS ?	# Websites	% Inclusions
googleapis.com	Yes	753	6%
google.com	Yes	617	4%
addthis.com	Yes	451	4%
googleadservices.com	Yes	443	3%
googlesyndication.com	Yes	416	3%
twitter.com	Yes	358	3%
facebook.net	Yes	292	2%
google-analytics.com	Yes	283	1%
sharethis.com	No	234	2%
doubleclick.net	Yes	175	2%

Table 4.5: Top ten mixed-JavaScript content providers

As for the other mixed content types, it is the same case that most of the top ten providers of mixed content serve their content also over HTTPS. As shown in Figure 4.5, about 60% mixed-iframe inclusions and 50% of mixed-Flash content inclusion are provided by the top ten providers, which again indicates that many insecure inclusions can be avoided if the websites correctly include this content.

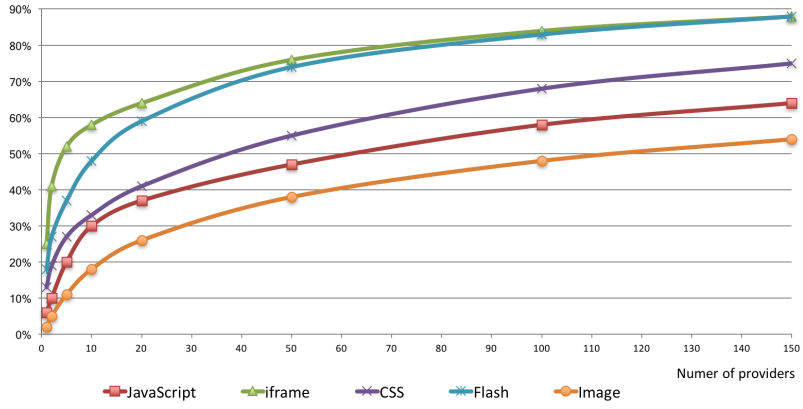

Figure 4.5: Percentage of mixed-content inclusion per cumulative number of top mixed-content providers

For cumulative numbers of top mixed-content providers, Figure 4.5 shows the percentage of mixed-content inclusions per provider. For example, the top 100 content providers serve about 80% of mixed iframe and Flash inclusions, more than half of mixed JavaScript and CSS inclusions, and 48% of mixed Image inclusions. This indicates that if the top 100 mixed-content providers offer their content over HTTPS and the websites correctly include this content, then the majority of mixed-content inclusions can be altogether avoided.

4.6 Mixed Content Mitigation Techniques

In this section, we investigate and enumerate protection techniques that can be used for browser vendors, TLS-protected websites, and content providers, to mitigate the issue of insecure inclusion of content.

4.6.1 Browser vendor

Blocking mixed content at the browser level, is the most straightforward way to mitigate the mixed content issue. While most desktop browsers have developed a mixed-content blocker to protect users against insecure content, mobile browsers lag behind on this, despite the fact that mobile browsing is becoming increasingly important to users.

As part of our study, we investigated how all the major mobile browsers for Android, iOS, Windows Phone and Windows RT platform handle mixed content – shown in Table 4.6[1]. We unfortunately discovered that most of them do not have a mixed-content blocker, with the exception of Chrome for Android and IE 10 Mobile which protect the user against mixed content. Firefox for Android plans to have a mixed-content blocker in a future release[2].

Platform	Mobile browser	blocked ?	secure padlock shown ?
Google Android 4.2	Chrome 28	Yes	with a yellow triangle
	Firefox 23	No	No
	Android browser	No	open padlock
	Opera Mobile 12	No	No
Apple iOS 6.1	Safari 6	No	No
	Chrome 28	No	with a yellow triangle
	Opera Mini 7	No	No
Windows Phone/RT 8	IE 10 Mobile	Yes	No

Table 4.6: Mobile browsers' behavior towards mixed content

With respect to desktop browsers, Internet Explorer (IE) is the first browser that detected and blocked mixed content with IE 7, released in 2006. When mixed content is detected, the browser warns the user and allows her to choose whether insecure content should be loaded [44]. Many users, however, would probably click "Yes", rendering the mixed-content blocker useless [139]. An elegant way to handle mixed content would be to silently block mixed content without prompting the users. This approach has been chosen in Chrome (version 21+) [46], Internet Explorer (version 9+) [29], and the recently released Firefox 23 in August 2013 [47]. Safari[3] and Opera[4] browsers do not currently have a mixed-content blocker.

Chrome, IE, and Firefox all have a mixed-content blocker, but they handle different types of mixed content in different ways. While mixed CSS, JavaScript, Flash content are blocked by all of them, currently only IE and Firefox block mixed-iframe content, though Chrome plans to also block mixed-iframe content in a near future release[5].

An interesting fact is that mixed-Image content is blocked in IE 7 and IE 8, but it is not blocked in IE 9 and IE 10. Since mixed Image and iframe content

[1]The study was conducted in July 2013, which might not reflect the status in 2017.
[2]Firefox for Android version 25 has a mixed content blocker (released in October 2013).
[3]Safari browser blocks mixed content from version 9 (released in September 2015).
[4]Opear adopted Chromium as its base since version 15 in August 2013, thus having the same mixed content blocker as Chrome now.
[5]Chrome blocks mixed-iframe content from version 29 (released in September 2013).

technically have the same impact which may lead to attacks "Request forgery" and "Cookie stealing", we recommend all browsers vendors to block all types of mixed content, thus completely eliminating the mixed content issue from the browser side. While this move would likely break some insecurely-coded websites, the security benefits of mixed-content-blocking definitely outweigh the temporary frustration of users when they encounter some websites that do not properly work.

4.6.2 Website owner

HTTPS websites can explicitly opt-in to only include content from secure channels. As shown in Table 4.4, 44% of the mixed-content files are not correctly included, since the secure HTTPS version of the same resources exist and could thus be readily used. For the remaining set of mixed-content files that do not have a secure version, the resources can be cached locally, or proxied using their own SSL server.

To provide better security, a website using HTTPS can use a combination of the HTTP Strict Transport Security (HSTS) and Content Security Policy (CSP) [137] protocols, as illustrated in Listing 4.1.

Listing 4.1: Protecting TLS-protected sites via HSTS and CSP

```
1 Strict-Transport-Security: max-age=86400; includeSubDomains
2 Content-Security-Policy: default-src https:; \
3 script-src https: 'unsafe-inline'; \
4 style-src https: 'unsafe-inline'
```

First, HSTS can be used to guarantee that webpages are only served over HTTPS by forcing a compliant browser to only issue HTTPS requests for that website (line 1). By enforcing the HSTS policy, it can prevent SSL-stripping attacks [112]. Second, CSP can be used to detect mixed content violations (in report-only mode), and to actively block mixed content by specifying that only TLS-protected resources are allowed to be included (line 2). Notice that in this example the *unsafe-inline* directives are added to preserve temporary compatibility (lines 3-4), but website owners are encouraged to fully embrace the CSP technology so that they achieve full protection and no longer need these unsafe directives.

4.6.3 Resource provider

Resource providers can also mitigate the mixed content issue by offering content over HTTPS (even only over HTTPS). Moreover, resource providers can also use

HSTS to migrate non-HTTPS resources automatically and secure to HTTPS version. Based on the figures in Table 4.5, pushing an HSTS header on only nine of the most frequently used script providers would already render the mixed-content inclusion problem harmless for about 28% of the JavaScript inclusions. And as shown in Figure 4.5, if all the top 100 mixed-content providers enforce HSTS headers, the majority of insecure mixed-content inclusions can be avoided.

Notice, however, that not all browsers have support for HSTS policies (e.g., latest IE 10 and Safari 6)[6], and that HSTS inherently has a bootstrapping problem during a browser's very first visit to an HSTS website. During this first request, an active network attacker can strip the HSTS header and circumvent this protection technique. To counter this, Firefox and Chrome are shipped with a pre-loaded list of important HSTS websites [32]. Additionally, Google Chrome offers an "HSTS Preload List Submission" service [15] to allow a website submit itself to Chrome's "HSTS preloaded list".

4.7 Limitations

To analyze the mixed-content issue on a large scale, our study relies on automated crawling. This automation might overlook some facts. Firstly, the crawler is built upon a headless browser, which does not interact with webpages. So, it could not detect mixed content that are triggered by user interaction (dynamically loaded via CSS/JavaScript code execution). Secondly, when checking the availability of mixed content over HTTPS, we simply changed the scheme in resource URL from "http" to "https", and then examined the response status of the new URL. This might fail to observe the same "HTTPS-available" resources located on other domains or other paths. Also it includes measurement errors if the "https-version" resource does not match the "http-version" resource. One way to address this latent error would be downloading the resource files and comparing the hash values of these files.

Readers should also note that the study presented in this chapter only provides a snapshot analysis in 2013. As web technology evolves, the landscape of mixed-content issues might have changed. To detect mixed content on a webpage, one can manually check the error messages prompted in a browser's console, or use some online tools to scan the page [34, 25, 36].

[6]IE 11 added HSTS support in June 2015, Safari 7 supported HSTS in October 2013.

4.8 Conclusion

The issue of mixed content was already noticed by security researchers in 2008, Collin Jackson and Adam Barth documented different browsers' behaviour towards mixed content [102]. At that time, only Internet Explorer 7 displays a "nonsecure content" warning prompt to let users decide whether to allow mixed content or not. The issue was largely ignored for several years, mainly due to the very low usage of HTTPS.

As the Web increasingly adopts SSL/TLS, the mixed content issue becomes more relevant. To better understand the issue, this chapter presents a large-scale survey on the mixed-content inclusion problem in 2013. It was the first attempt to systematically analyze the issue, and uncover the state of practice of mixed content inclusion. Our study shows that almost half of the HTTPS websites have at least one mixed-content page. And more than half of these vulnerable pages allow for the most powerful attacker model, i.e., the ability to execute arbitrary JavaScript in the context of the vulnerable website.

A sincere effort from the websites themselves involving converting the HTTP requests to HTTPS requests, would significantly diminish the total number of vulnerabilities but it would not eradicate them completely since a large amount of content is only provided over HTTP. This, in turn, points towards the responsibility of the content providers who must take action so that their resources may be made available over secure channels.

Another approach to mitigate the issue is blocking mixed content in browsers. While most desktop browsers already have a mixed content blocker in 2013, some mobile browsers lagged behind on offering protection against mixed content, despite the increasing popularity of mobile devices. Additionally, mobile browsers, when compared to desktop browsers, have less support for displaying HTTPS connection details, and for the warning of users about mixed content [56]. Since users are entering into the "Post-PC era", i.e., preferring mobile devices for regular Internet browsing and even for sensitive online transactions, it is important for mobile browsers to have equivalent protections as their desktop counterparts.

As of December 2017, all major web browsers (both desktop and mobile versions) block mixed content by default, which considerably alleviated the issue. However, this does not completely solve the problem for websites owners and web users. Firstly, browsers' mixed-content blocker might cause some webpages to become unfunctional if, for example, the page depends on a mixed JavaScript file. To ensure both functionality and security, websites owners still need to fix the issue by themselves. Furthermore, some web users are still using old web browsers that do not provide protection. These users are still under the threat

of MITM attacks. Thus our study on mixed-content issue is still relevant for websites owners and web users. It motivates websites owners to address the issue, and reminds web users the possible invisible threat behind a seemingly secure website.

Chapter 5

Security Assessment of the Chinese Web

Preamble

The contents of this chapter are adapted from the paper titled "Security Analysis of the Chinese Web: How well is it protected?" [71], which was published in the Proceedings of the 2014 Workshop on Cyber Security Analytics, Intelligence and Automation. This work was done with the collaboration of other authors from DistriNet, KU Leuven. Ping Chen was the lead author of this paper.

This chapter presents a large-scale security assessment of the Chinese Web. The assessment is based on analysing the usage of known security mechanisms introduced in Section 3.1, and evaluating the strength of HTTPS implementations. Our work uncovers some "Chinese characteristics" with respect to Web security.

5.1 Introduction

With the establishment of a 64K dedicated circuit to the United States, China became part of the global Internet in April 1994 [67]. Since then the Web develops fast in China. As of June 2017, China has 751 million Internet users, which accounts for 54% of its population, and more than 5 million websites exist on the Chinese Web [76]. The Web is getting increasingly integrated

into Chinese society, and this process is further facilitated by the widespread availability of cheap smartphones in recent years. 96% of Chinese Internet users regularly access the Web with their mobile phones.

While Chinese Internet users enjoy the convenience and flexibility that the web brings them, and Chinese companies heavily depend on the Web for their business operations, at the same time, the Chinese Web also draws increasing attention from attackers. Several severe data breaches have made a headline in the past few years. In 2011, more than 40 million user accounts were stolen from Tianya Club (天涯, tianya.cn, the largest Chinese online forum). Over 20 million hotel booking records containing customers' private information (e.g., identity number, home address) were leaked on the Chinese web in 2013. And recently JD.com (京东商城), a popular Chinese e-commerce site, was reported to have been attacked, which resulted the leakage of 12GB of user data in 2017.

These security incidents are often caused by unpatched known vulnerabilities. According to an analysis report from Qihoo 360, a Chinese Internet security company, about 46% Chinese websites have vulnerabilities in 2016 [132]. Active scanning/testing for vulnerabilities, as done by Qihoo 360, can help websites to mitigate attacks, but it is typically time and labor consuming. In this chapter, we assess the security of the Chinese web from another perspective. Instead of searching for vulnerabilities and weaknesses, we seek to discover the usage of defense mechanisms by Chinese websites, trying to answer the question "How well is the Chinese web protected?".

In particular, we focus on client-side security policies, and HTTPS implementations, both of which can be passively detected. These defense mechanisms are developed by the security community for securing the Web, thus the presence of these mechanisms on a website can be used as an indicator of the security awareness and practices of that website. By passively analysing the adoption of defense mechanisms, an outsider can assess the security of a large number of websites belonging to a country, or a specific industry sector. As people depend more and more on the web for their daily lives and businesses, such a large-scale assessment might be desirable for government and supervisory organizations to continuously monitor the security of the Web environment.

The recent work by Van Goethem et al. [140], in which they conducted a security assessment for more than 22,000 European websites, has shown such a large-scale security analysis of the web is achievable, albeit challenging. In this chapter, we apply the same basic methodology to investigate the security of the Chinese web through a large-scale experiment.

The main contributions of this chapter are the following:

- We report the usage of client-side security policies in the top 10,000 Chinese websites, and compare it to the statistics obtained from non-Chinese websites, showing that the Chinese web lags behind with respect to the adoption of client-side security policies.
- We provide a comprehensive evaluation of HTTPS implementations on the Chinese web, illustrating that the majority of HTTPS adopters do not have secure and protected HTTPS implementations.
- We present a case study on the inadvertent private data leakage, showing that 6% of websites leak Chinese identity numbers, that are collected in spreadsheet files and can be obtained by search engines.

5.2 Data Collection

According to the status report published by China Internet Network Information Center (CNNIC) in 2017, there are more than 5 million websites on the Chinese web [76]. Since checking the whole content of all websites on the Chinese web is close to infeasible due to its enormous size, we focus on the high-profile websites that are ranked in Alexa's list of most popular websites.

Starting with Alexa's list of top 1 million sites, we first select the set of `.cn` domains from the list. Next, we cross-check `top.chinaz.com`, a website providing a ranked list consisting of more than 6,000 Chinese websites, in order to identify the set of Chinese websites not using `.cn` domain (e.g., `baidu.com`, `qq.com`) in Alexa's list. As a result, we obtain a set of more than 12,000 Chinese websites.

We then filter out the websites without an ICP (Internet Content Provider) license. The ICP license is a permit issued by the Chinese government to permit China-based websites to operate. Websites operating in China without an ICP license will be fined or shut down as specified by Chinese Internet regulations. By removing websites without an ICP license from our dataset, we try to avoid the inclusion of some malicious websites involved in illegal online activities, such as phishing and porn websites. After this filtering step, we obtained about 10,000 websites as our targets, to represent the high-profile part of the Chinese Web.

In order to assess the security of the top 10,000 Chinese websites, we conducted a crawling experiment to visit the popular webpages from these websites. For each website, we obtain up to 200 webpage URLs by using the Bing Search API [8] with parameters `site:domain` and `Market:zh-cn`. For instance, the search for `site:baidu.com Market:zh-cn` in Bing will return a set of Chinese

webpages belonging to `baidu.com`. By setting the parameter `Market:zh-cn`, we instruct Bing to only search for Simplified Chinese webpages originating from Mainland China, while excluding any English or Traditional Chinese webpages that are not targeting users in Mainland China (hence they not considered as part of Chinese web). After the webpage URLs are obtained, we use the web crawling setup and methodology introduced in Section 3.2 to visit each URL.

For a comparison analysis, we collected a set of 10,000 non-Chinese websites, and launched the same crawling experiment for these non-Chinese websites. The set of non-Chinese websites was collected as follows: for each Chinese website included in our study, we randomly select a non-Chinese website with the closest rank from Alexa's top 1 million sites. For example, `yahoo.com` (rank 4) is selected as it is the closest neighbor of `baidu.com` (rank 5).

In total, we analyzed more than 1.4 million webpages for the top 10,000 Chinese websites, with an average of 147 webpages per website.

5.3 Usage of Client-side Security Policies

Client-side security policies are declarative security mechanisms, whereby a website communicates its intent and leaves it up to the browsers to enforce it. There are a number of benefits to the use of client-side security policies [101], with auditability being one of them. Client-side security policies are typically sent via HTTP response headers, thus it is straightforward to determine a website's security expectations by passive analysis.

By instructing browsers to enforce protection through server-provided policies, websites can address a number of security issues in a very straightforward manner. Although these policies are not security panacea, their usage on a website can indicate the "security consciousness" of that website and its security objectives.

In our assessment, we focus on four security features that are briefly described below (detailed explanation can be found in Section 3.1).

- **HttpOnly Cookies** The `HttpOnly` attribute in a `Set-Cookie` header restrict the access of that cookie to the HTTP(S) protocol, preventing cookie stealing via JavaScript.
- **Content Security Policy (CSP)** CSP can be used to mitigate several types of attacks including XSS. It allows a site to declare trusted sources of content that browsers should be allowed to load on that site.

- **X-Frame-Options** The `X-Frame-Options` header is designed to avoid Clickjacking attacks.
- **X-Content-Type-Options** The `X-Content-Type-Options` header prevents MIME-sniffing, thus reducing exposure to certain attacks.

Table 5.1 gives an overview of the usage of these security policies on the Chinese web. HttpOnly cookies are much more widely used than other policies, and their usage on Chinese websites is greater than that of non-Chinese websites. The `X-Frame-Options` and `X-Content-Type-Options` header are much less popular, and their usage on the Chinese web is lower than the estimated global statistics. For CSP, the adoption both on the Chinese web and globally is quite low, most likely due to the relative newness of the mechanism.

Security mechanism	% Non-Chinese Website	% Chinese Website	Example Chinese Website
HttpOnly Cookies	25.7%	43%	alipay.com
Content Security Policy	0.06%	0.01%	zhihu.com
X-Frame-Options	5.8%	1.2%	weibo.com
X-Content-Type-Options	4.6%	0.5%	alibaba.com

Table 5.1: Usage of client-side security policies on the Chinese web

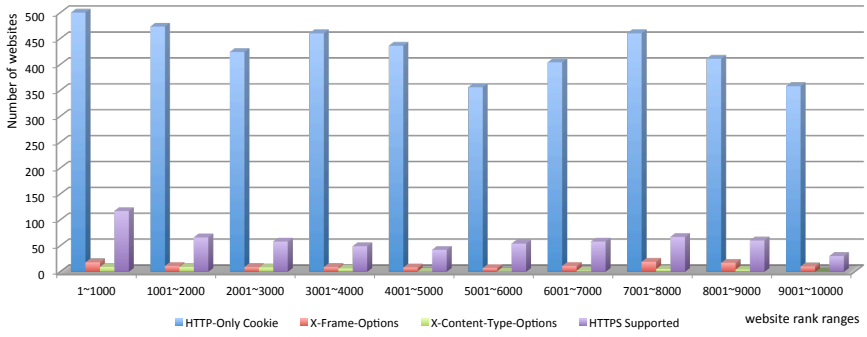

Figure 5.1: Distribution of websites using client-side security policies over Alexa rank ranges

By analyzing the rank of websites using client-side security policies, we found that there is no strong correlation between a website's popularity and its

adoption of client-side security policies, as shown in Figure 5.1. The figure also shows the distribution of Chinese websites with HTTPS implemented, which will be discussed in next section.

Since client-side security policies rely on a browser's enforcement, it is important to supply the browser with a correct policy. In our experiment, we found 5 Chinese websites using the `X-Frame-Options` header with an incorrect value, e.g., `SAMEORIGIN/DENY`, which is effectively ignored by the browser. Overall, the adoption of client-side security policies on the Chinese web is low, which indicates that a lot Chinese websites lag behind with respect to adopting countermeasures on the client side.

5.4 Security of HTTPS Implementations

The HTTPS protocol is the standard solution for securing web traffic, which guarantees the confidentiality and integrity of web communications by adding the security capabilities of SSL/TLS to HTTP. It also provides website authenticity with the CA/B (Certificate Authority/Browser) trust model. While HTTPS is designed to provide strong security, it may fail to achieve the desired security goals if it is implemented in the wrong way. A range of security issues associated with HTTPS have, over the time, been discovered, ranging from cryptographic weaknesses and design flaws in SSL/TLS protocol, to the insecure design of HTTPS websites, and bad coding practices [74].

When migrating to HTTPS, websites should try to avoid known security issues, and consider to add extra defenses to HTTPS. In this section, we assess a website's HTTPS implementation in two ways: (1) the presence of known security issues related to HTTPS; (2) the usage of extra client-side security policies for the better enforcement of HTTPS.

Note that when discussing HTTPS security, we use the active network attacker model. An active network attacker positions himself on a network between the host running the web browser and the web server, and is able to intercept and tamper with the network traffic passing by. The attacker can read, modify, delete, and inject HTTP requests and responses, but he is typically not able to decipher any encrypted information.

5.4.1 HTTPS security issues

To build a secure HTTPS website, there are a number of security pitfalls that websites should try to avoid. In this section, we check whether an HTTPS

website suffers from the following security issues:

- **Insecure SSL/TLS Implementation** In our assessment, we use an SSL scanner called sslyze [37] to search for the following security issues related to SSL/TLS: broken certificate validation chain (e.g. untrusted CA), support of insecure SSL 2.0, use of weak ciphers (e.g. export-grade ciphers with small encryption key length), and the vulnerability to insecure renegotiation attacks [39], and CRIME attacks [128]. This assessment is similar to Qualys' SSL survey for the global SSL landscape [35].
- **Post-to-HTTPS Forms** It is a relatively common practice in many HTTPS websites to provide a form (such as a login box) on an HTTP page while arranging for any sensitive information to be submitted over HTTPS. This, however, is a bad practice, since an active network attacker can launch an SSL-stripping attack to steal a user's sensitive data without raising the user's suspicion [112].
- **Mixed-content Inclusion** Mixed-content inclusion occurs when the main webpage is sent over a secure HTTPS channel, while some additional content included on that page, such as images and scripts, are delivered over non-secured HTTP connections. As a result, an active network attacker can still try to compromise an HTTPS website by intercepting and modifying the unencrypted content [72].

5.4.2 Client-side security policies for HTTPS websites

In addition to the client-side security policies discussed in Section 5.3, HTTPS websites can also make use of the HTTP `Strict-Transport-Security` policy and the `Secure` attribute of cookies. These mechanisms are specifically designed for HTTPS websites and can be used to mitigate some HTTPS-specific security issues.

These two security features are briefly described below (detailed explanation can be found in Section 3.1).

- **HTTP Strict-Transport-Security (HSTS)** Set by a website via a HTTP response header field (`Strict-Transport-Security`), HSTS specifies a period of time during which the user's browser is instructed that all requests to that website need to be sent over HTTPS, regardless of what a user requests. The HSTS Policy helps protecting website users against both passive eavesdropping, as well as active man-in-the-middle (MITM) attacks.

- **Secure Cookies** Although the traffic between a web server and a browser is encrypted when using HTTPS, the cookies stored in the browser are not, by default, limited to an HTTPS context. By setting the `Secure` attribute, the scope of a cookie is limited to secure channels, thus stopping browsers from sending cookies over unencrypted HTTP requests.

5.4.3 Findings and discussion

In order to identify HTTPS pages from the 10,000 websites, we try to enumerate HTTPS URLs for each website. First, we select HTTPS URLs obtained from Bing. Additionally, we search for any HTTPS links on HTTP webpages during our crawling experiment, and add these HTTPS links to our dataset for later crawling. By doing so, we identified 672 Chinese HTTPS websites, with an average number of 15 HTTPS pages per site. For the dataset of non-Chinese websites, we found 1,601 HTTPS websites, with an average number of 19 HTTPS pages per site. We summarise our findings concerning HTTPS security issues and defense mechanisms in Table 5.2.

	% Non-Chinese HTTPS websites	% Chinese HTTPS websites	Example findings
Insecure SSL/TLS Implementation	70.9%	84.1%	passport.baidu.com
Post-to-HTTPS Forms	31.8%	21.4%	tenpay.com
Mixed-content Inclusion	20.7%	10.6%	xiu.com
HTTP Strict Transport Security	3.6%	1.3%	alipay.com
Secure Cookies	19.9%	30.7%	hangseng.com.cn

Table 5.2: Assessment overview for Chinese HTTPS websites

The vast majority (84.1%) of Chinese HTTPS websites have SSL/TLS implementation issues. More specifically, 13% of websites are using self-signed certificates, 19% have insecure SSL 2.0 enabled, 30% support weak ciphers, and 18% and 33% are vulnerable to CRIME and SSL Renegotiation attacks, respectively.

One can see that Chinese HTTPS websites tend to have less problems with respect to post-to-HTTPS forms and mixed-content inclusion, when comparing

to the non-Chinese HTTPS websites. Note, however, that we can not claim that Chinese HTTPS websites are doing better than non-Chinese HTTPS websites. Since 28% of Chinese HTTPS websites have only one HTTPS webpage, the probability of these issues occurring in Chinese websites is smaller than on non-Chinese websites, which offer many more pages over HTTPS.

As for the usage of client-side security policies on HTTPS websites, we only found 8 websites using HSTS policies, and 206 websites having secure cookies. Interestingly, we noticed that websites are more likely to enable other client-side security policies when they have HTTPS implemented. For example, 69.8% of Chinese HTTPS website also make use of HTTP-Only Cookies, a fraction much higher than the one presented earlier in Table 5.1 (43%).

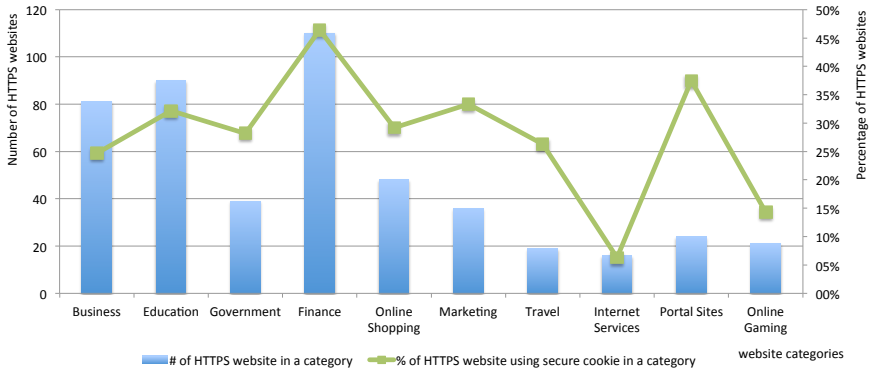

Figure 5.2: Distribution of Chinese HTTPS websites over top 10 categories

As for the distribution of HTTPS websites, we didn't find any strong correlation between a website's popularity and its adoption of HTTPS (as shown in Figure 5.1). To better understand the Chinese websites using HTTPS, we categorized the 672 HTTPS websites, using McAfee's TrustedSource Web Database [24]. For HTTPS websites not found in McAfee's database, we manually visit the websites and label them.

The number of HTTPS websites and the percentage of HTTPS websites having secure cookies over the top 10 categories are presented in Figure 5.2. One can see that, the financial websites are doing better than websites in all other categories, with 110 financial websites having HTTPS implemented, and about half of them using secure cookies.

With only 6.7% of websites having HTTPS implemented, the adoption rate of HTTPS on the Chinese web is much lower, compared to the global statistic of 27% of the Alexa top 1 million websites already used HTTPS in 2010 [127].

Although HTTPS is good for security, many websites are reluctant to migrate to HTTPS, due to several concerns, such as, SSL/TLS performance overhead, and operational costs. Beside these common concerns shared by websites globally, Chinese websites have to consider a China-specific issue when deciding whether to adopt HTTPS, which is the lack of HTTPS support from Baidu.

As explained in Baidu's official search engine optimization (SEO) guide [5], Baidu does not have a good support for HTTPS, thus Chinese websites are recommended to avoid the use of HTTPS. If HTTPS is essential for a website, the website should try to make the HTTPS webpages also available over HTTP, in order to get indexed by Baidu.

To further support our claim about Baidu's partial liability for the low HTTPS adoption on the Chinese web, we compare the search results from Baidu and Bing, for the 672 Chinese HTTPS websites. While we obtained about 2,000 HTTPS URLs for 296 (44%) HTTPS websites during data collection with Bing Search API, we didn't get any HTTPS URLs from Baidu when using the same queries like `site:example.com` to search for the popular webpages of each HTTPS website.

5.4.4 Usage of KNET trusted website certificate

One of the benefits of implementing HTTPS is website authenticity, which can help users to identify legitimate websites and prevent some phishing attacks. With only 605 Chinese websites having HTTPS implemented, and 13% of them using a self-signed certificate, it is challenging for Chinese Internet users to identify phishing websites by checking SSL certificates. In order to protect their users from phishing attacks, many Chinese websites opt for a Chinese-specific approach for website authentication by using trusted website certificate issued by KNET (owned by CNNIC) [22].

KNET certifies a website based on its ICP license, registration information in industrial and commercial bureau, and organization information. Certified websites are recorded in the "National Trusted Site Database Open Platform", which is used by various client-side applications to help users identify trusted websites. These applications include search engines (Sogou, SOSO and Bing), browsers (Sougou, Maxthon, Taobao, Ali, QQ and 114la), anti-virus software (Jinshan), and IE browser plugins. Considering the large amount of phishing websites on the Chinese web [77], and the various concerns about HTTPS implementations, we think that using a trusted website certificate is a practical and effective way for Chinese websites to defend themselves against phishing attacks.

By querying the trusted site database, we found that only 21.8% (2,182) of the investigated Chinese website are using KNET's trusted website certificate. Similar to the distribution of HTTPS websites, certified trusted websites mostly belong to categories such as government, business, finance, and online shopping.

5.5 Identity Leakage

Online data leakage incidents are usually caused by cyber attacks. For example, in December 2013, Chinese hackers leaked over 20 million hotel reservation records containing customer information [1]. Careless or improper management of sensitive data, however, may also lead to incidents. For instance, a Chinese university inadvertently made 2,000 students' identity numbers and bank accounts available, through files located on their website [2]. In this section, we investigate the issue of inadvertent data leakage. More specifically, we check whether a website is hosting spreadsheet files containing Chinese identity numbers (henceforth ID numbers), that can be obtained simply via search engines (as shown in Figure 5.3).

Figure 5.3: Searching inadvertent identity leakage via Google

For each Chinese website in our dataset, we query both Google and Baidu with "身份证(literally means ID card) `site:example.cn filetype:xls`" in order to obtain the first hundred search results. Then, for each result, we examine the description part using regular expressions and the Chinese ID checksum algorithm [150], in order to find valid ID numbers. If the description part of a search result contains a valid ID number, then we claim that the search result (spreadsheet file) leaks private ID information. At the end of this process, we found 2,496 spreadsheet files hosted on 548 websites through Google searches, and 1,422 spreadsheet files hosted on 315 websites through Baidu searches.

In total, 3,680 spreadsheet files containing ID numbers were found, located on 603 (6%) Chinese websites. After categorizing these 603 websites, we found that the majority of these files are hosted on governmental and educational websites, as shown in Table 5.3. By examining the file names and descriptions from the results of the two search engines, we found that these spreadsheet files containing ID numbers are typically used for announcing the results of various activities, such as recruitment in government, enrollment in universities, and awarding prizes and scholarships.

Category	#/% websites	#/% files	Example findings
Government	247 (41%)	1,592 (43.2%)	Candidates for a government bureau (广州市工商局体检名单) (gzaic.gov.cn)
Education	237 (39.3%)	1,543 (41.9%)	Registrants in a university (浙江大学报名点名单) (zju.edu.cn)
Others	119 (19.7%)	545 (14.9%)	Participants in match (竞走锦标赛人员名单) (sport.org.cn)

Table 5.3: Distribution of websites leaking spreadsheet files containing Chinese ID numbers

The issue of inadvertent private data leakage on Chinese Web, is mainly due to the lack of privacy awareness in Chinese organizations and the lack of legal protection for private information in China. Although such inadvertent mistakes are not malicious in nature, the leaked IDs can be used by criminals for various illegal activities, such as selling them to teenagers for online gaming registration (to evade the anti-addiction system), using them for train ticket scalping (a valid ID is required for buying train tickets online), and operating malicious online shops with impersonated identities.

Moreover, some leaked spreadsheet files also contain other types of personal information such as phone numbers, occupations and addresses. A motivated adversary can leverage all these private information to conduct social engineering attacks against related individuals, and subsequently to launch targeted attacks against associated organizations. For example, some leaked spreadsheet files contain recruitment information for the government, which could, in principal, be used by advanced persistent threat actors [68] to attack the Chinese government.

We consider this as a severe privacy issue on the Chinese web. If an ID number

or phone number must be made available for identification or authentication, organizations can straightforwardly mask some digits in the number (e.g., 123456******1234 for an ID number, 138****1234 for a phone number), in order to protect private information. Considering that the last digit of an ID number is a checksum, it is better to also mask that digit (e.g., 12345619****567*) instead of masking only 4 birthday digits (as used for printing an ID number on a train ticket), in order to obtain higher anonymity and prevent brute-forcing the ID number based on the ID checksum algorithm.

To quantify websites that are using masked digits for protection, we also searched for masked ID numbers when examining results from search engines. In total, we found 83 websites using masked digits when publishing spreadsheet files. Not surprisingly, 9 of them are financial websites, including the big four Chinese banks. This shows that financial organizations put more effort in protecting themselves and their users.

5.6 Revisiting the Situation in 2017

It has been almost three years since our first study on Chinese Web [71]. To get a view on the recent situation, this section presents a follow-up assessment in 2017. For each security feaure, Table 5.4 shows the percentage of Chinese websites with that feature enabled in 2014 and 2017. Clearly, the usage of defence mechanisms on the Chinese Web has considerably increased over years, although the overall adoption rate is still lower, compared to the global Web.

Security Features	% Websites in 2014	% Websites in 2017
HttpOnly Cookies	43.0%	56.7%
Content Security Policy	0.01%	1.7%
X-Frame-Options	1.2%	16.8%
X-Content-Type-Options	0.5%	7.8%
HTTPS Support	6.7%	25.8%
Secure Cookies	2.1%	6.8%
Strict Transport Security	0.07%	4.6%

Table 5.4: Adoption rate of defence mechanisms on the Chinese Web in 2014 and 2017

A particular progress is the increasing adoption of HTTPS on the Chinese Web. This improvement can be partly attributed to the change of Baidu's crawling policy. As explained in Section 5.4.3, Baidu was partially blamed for the low

HTTPS adoption in 2014, it had no support for HTTPS websites until May 2015 [6]. While more Chinese websites are migrating to HTTPS, many of them fail to address HTTPS security issues. 37% of Chinese HTTPS websites have mixed-content inclusion in 2017, a significant increase from the 10% in 2014.

As for the inadvertent identity leakage, it remains a serious privacy issues. Although we found more websites are using masked digits to hide identity number, there are still about 4% of Chinese websites should be blamed for their poor management of individuals' private data in 2017. Most of them are websites of government bodies or educational institutes.

5.7 Limitations

During our data collection process, we used Alexa's list to select .cn domains and Bing search engine to obtain page URLs. However, both Alexa and Bing are not dominant in the Chinese market. Thus our dataset might be biased towards Chinese websites that are optimized for Alexa and Bing (e.g., by including site validation code in HTML). Moreover, our crawler runs outside China, the content it retrieved might be different if it runs inside China. This is mainly due to the so-called "Great Firewall" employed by Chinese government to censor international traffic, and content restrictions (based on IP geolocation, copyright, etc.) imposed by website owners. To have a better representative dataset, one can use the Baidu search engine and run the crawler inside China.

A second limitation is that many small and medium Chineses organizations, while having their own websites, do not use the website to provide service to customers, instead they rely on web services provided by big companies like BAT (the three largest tech giants in China: Baidu, Alibaba, and Tencent) for business operations. For example, a small company might have an online shop on Taobao (owned by Alibaba), or a service site in WeChat (owned by Tencent), while its own website is only used for displaying some static information. Thus, the impact of an insecure Chinese website might not be as severe as an insecure European website.

5.8 Conclusion

As the web becomes more complex and popular, security and correctness become ever more crucial attributes of web applications. A variety of methods and techniques have been proposed to test web applications [110], and most of them are designed to detect specific vulnerabilities and errors such as SQL Injections

and XSS attacks. In this paper, we analyze the presence of defense mechanisms, and use it as a security indicator to measure the security of the Chinese web.

To the best of our knowledge, this paper is the first that attempts to analyze the overall security of the Chinese web from the aspect of the adoption of defense mechanisms. By investigating the usage of client-side security policies, and assessing the HTTPS implementations on the Chinese web, we observed that the majority of websites lack support for client-side security policies, and that the statistics of vulnerable HTTPS implementations of Chinese websites are also worrisome. Moreover, we found that 6% of websites are leaking Chinese ID numbers through spreadsheet files that can be obtained by simple searches in Google and Baidu. We hope that our study can help Chinese websites owners to discover and prioritize the adoption of security mechanisms, and raise the security and privacy awareness among Chinese web users.

Chapter 6

Longitudinal Study of Web Security

Preamble

The contents of this chapter are adapted from the paper titled "Longitudinal Study of the Use of Client-side Security Mechanisms on the European Web" [69], which was published in the Proceedings of the 25th International Conference Companion on World Wide Web in 2016. This work was done with the collaboration of other authors from DistriNet, KU Leuven. Ping Chen was the lead author of this paper.

This chapter presents a long-term longitudinal security assessment of the European Web. Our study identifies a trend of increasing usage of client-side security mechanisms on the web, and proposes a web security scoring system to compare the security postures among different websites.

6.1 Introduction

The web is constantly evolving, with new technologies such as HTML5 and CSS3 getting widely used and supported, which provide Internet users richer experience. In the mean time, the attacks on the web are also changing, shifting from server exploitation such as SQL injection to client-side attacks such as XSS and Man-in-the-Middle (MITM) attacks such as SSL-stripping.

In response to this trend, various client-side security mechanisms are developed, such as Content Security Policy (CSP) and HTTP Strict-Transport-Security (HSTS). These client-side security mechanisms are server-driven, but requires the browser to enforce them. The presence of these mechanisms on a website can be used as an indicator of the security awareness and practices of that website.

This chapter tries to give an overview of the adoption of client-side security mechanisms on the web, and provide a state-of-practice reference model of web security for website operators. To achieve this, we crawled more than 18,000 European websites in a four-year period. With the gathered data, we analyze the evolution of the usage of client-side security mechanisms, and use a security scoring system to compare a website to its peers (based on business vertical or popularity), in order to provide a web security baseline for website operators.

Our main contributions are the following: (1) We report the usage of seven client-side security mechanisms on European web in September 2013, September 2015 and September 2017, and analyze the evolution of adoption (i.e., identify which security features are being adopted over time); (2) We propose a web security scoring system to compare the security posture (i.e., the practice of the usage of client-side security features) among different websites, and among countries, business sectors; (3) We provide a web security baseline and maturity model for website operators, by applying the web security scoring system to a set of websites.

6.2 Data Collection

To study the security posture of the European web popular websites from the 28 member states in the EU are chosen to represent the European web. For each EU country, we selected the top 1,000 websites ending with the corresponding ccTLD (country code top-level domain) from Alexa's list of the top 1 million sites [3][1]. As a result, we have a set of 23,050 European websites.

We then use the crawling approach introduced in Section 3.2 to collect data from these websites in a four-year's timeframe. We remove the websites with less than 50 successfully crawled pages from our dataset. As a result, we have a dataset of 20,157 websites in 2013, 18,074 websites in 2015, and 14,984 websites in 2017, as shown in Table 6.1. The number of websites decreased over time because some websites disappeared and some websites changed domain names.

[1] Alexa top 1 million list in September 2013

Time	# of sites	# of pages	avg. # of pages/site
Sept. 2013	20,147	3,499,080	174
Sept. 2015	18,074	2,992,395	166
Sept. 2017	14,984	2,266,338	151

Table 6.1: Overview of European Web dataset for longitudinal study

6.3 Security Features and Scoring System

To design a scoring system that reflects a website's security level, we had a panel, consisted of four web security experts from our research group, to discuss about what metrics can be used for that. Instead of focusing on finding web vulnerabilities, we considered the question "how could we change a non-secure website to be better secure, by adopting defensive mechanisms step by step, based on priority consideration?"

6.3.1 Client-side security features

We began with literature on the common web attacks and best practices to improve web security. In particular, we went through the OWASP Top Ten Most Critical Web Application Security Vulnerabilities [120] and the CWE/SANS Top 25 Most Dangerous Software Errors [113], to identify the major web threats. We observed that while server-side attacks, such as SQL injection and command injection, are still prevalent, the client side is receiving increasing attention from web attackers. Meanwhile, client-side countermeasures and security policies are also developing [131].

We identified three categories of threats that can be mitigated from the client side: Insecure Communication, Cross-site Scripting (XSS), Insecure Framing. In order to counter these threats, website operators can adopt seven client-side security features.

- **Category 1: Secure Communication** This category includes four features that contribute to secure communication between a server and a browser: HTTPS support, HTTP Strict Transport Security (HSTS) and Secure Cookies.
- **Category 2: XSS Mitigation** This category includes three features that can be used to mitigate XSS attacks: HTTPOnly Cookies, X-Content-Type-Options (XCTO), and Content Security Policy (CSP).

- **Category 3: Secure Framing** This category has one feature X-Frame-Options (XFO) to enable secure framing.

By taking into consideration of the maturity of the feature, the complexity to implement, the coverage of the feature, and the impact of related attacks, we discussed how these client-side security features should be adopted. The maturity is assessed based on the age of a feature, and the support from browsers. The implementation complexity reflects the efforts needed to deploy the feature. The coverage indicates the effect of a feature to prevent attacks.

Feature	Maturity (Year)	Coverage	Complexity	Impact
HTTPS support	High (1996)	Low	High	High
HSTS	Medium (2012)	High	Low	High
SecureCookies	High (2000)	Medium	Low	Medium

Table 6.2: Client-side security features for Secure Communication

For the first category ("Secure Communication"), HTTPS is the most fundamental feature. While it is well matured, the implementation of HTTPS still involves complex configuration and management [74, 115]. Once a website has HTTPS implemented, it can then adopt Secure Cookies and HSTS to strength HTTPS. Both HSTS and SecureCookies are relatively easier to adopt, since they only require web developers to specify the policy in HTTP response headers. Though HSTS is less matured than SecureCookies, it provides broader coverage against attacks by forcing all communication over HTTPS (hence preventing cookies stealing as well).

Feature	Maturity (Year)	Coverage	Complexity	Impact
HttpOnlyCookies	High (2002)	Medium	Low	Medium
XCTO	Medium (2008)	Low	Low	Low
CSP	Low (2014)	High	Medium	High

Table 6.3: Client-side security features for XSS Mitigation

For the second category ("XSS Mitigation"), the HttpOnly Cookies is the most matured feature. It addresses the issue of cookie stealing in XSS attacks (medium coverage and impact). While CSP is relatively newer and difficult to implement [144, 89], it can be used to prevent not only XSS, but also clickjacking and data injection attacks (high coverage and impact). XCTO has low coverage and impact, since it only address XSS via MIME sniffing.

For the third category ("Secure Framing"), X-Frame-Options (XFO) can be used to prevent clickjacking attack. Compared to the other two categories, clickjacking has lower impact, and it is easy to mitigate.

Feature	Maturity (Year)	Coverage	Complexity	Impact
XFO	Medium (2009)	Medium	Low	Low

Table 6.4: Client-side security features for Secure Framing

6.3.2 Web security scoring system

Based on our expert opinions, we came up with a web security scoring system built upon the assessment of seven web security features. These features serve as the metrics of our web security scoring system, since they reflect a website's effort on security and they are easily detectable from normal browsing.

For each (group of) website(s), we define the overall security score ($OverallScore$) as a weighted average of three distinct subscores:

$$\begin{aligned}\textbf{OverallScore} = &\frac{40}{100} \times SecureCommunicationScore \\ &+ \frac{40}{100} \times XSSMitigationScore \\ &+ \frac{20}{100} \times SecureFramingScore\end{aligned}$$

As part of of scoring system, we assess for each security feature how well the (group of) website(s) is doing compared to websites in the full dataset. For instance, we want to grade a website with a score 0.61 for the feature HTTPS, if the website outperforms 61% of the websites in our dataset (i.e. by having a higher percentage of pages over HTTPS). The scores of the individual features are then combined to provide a metric for the three subscores.

More concretely, we apply the following approach:

1. For each security feature, we compute an empirical cumulative distribution function (ECDF) for all websites. The ECDF is computed based on the percentage of webpages having that feature on a particular website.

2. This computed ECDF is used to calculate an ECDF value per website and per feature.
3. The subscores are calculated by applying a weighted averages of the ECDF values.

The weight given for each feature reflects the relative importance and maturity of the feature in each category. The more fundamental and matured feature get relatively higher weights. In particular, the following weights are used to calculated the three subscores:

Secure Communication Score. This subscore is measured by applying a weighted average of the HTTPS, HSTS, and Secure Cookies usage.

$$\begin{aligned} SecureCommunicationScore = &\frac{45}{100} \times HTTPS \\ &+ \frac{25}{100} \times SecureCookies \\ &+ \frac{30}{100} \times HSTS \end{aligned}$$

XSS Mitigation Score. This subscore measured by applying a weighted average of the HttpOnly Cookies, XCTO, and CSP usage.

$$\begin{aligned} XSSMitigationScore = &\frac{50}{100} \times HttpOnlyCookies \\ &+ \frac{30}{100} \times XCTO \\ &+ \frac{20}{100} \times CSP \end{aligned}$$

Secure Framing Score. This subscore is measured by the XFO usage.

$$SecureFraming = \frac{100}{100} \times XFO$$

6.4 General Findings

6.4.1 The use of security features on European web

Table 6.5 gives an overview on the use of security features on European web over four years. It clearly illustrates that the web security on the European

Web did improve, as each of the security features have been adopted in 2017 by a larger fraction of websites than in 2013. One can also observe that the pace of improvement accelerates over time; the past two years have seen much faster adoption of security features.

Security feature	% of websites		
	Sept. 2013	Sept. 2015	Sept. 2017
HTTPS Support	22.96%	33.29%	71.97%
Secure Cookies	5.86%	7.56%	25.01%
HSTS	0.49%	4.30%	19.82%
HttpOnly Cookies	36.52%	43.86%	54.89%
XCTO	2.24%	6.82%	24.43%
CSP	0.05%	0.43%	5.71%
XFO	4.80%	14.93%	32.08%

Table 6.5: Overview of the use of security features on European web

We then assess to what extent the security of a particular website did improve over time, by measuring for each website if it adopts more security features over time or not. Since none of the websites have all seven security features enabled in 2013, there is space for improvement for all the websites.

By doing this, we found that more websites improved from 2015 to 2017 than they did between 2013 and 2015. There are 8,685 websites adopted more security features by 2017 than what they already have in 2015, while the number of improved websites from 2015 to 2017 is 5,756. And by 2017, there are 377 websites have all seven security features enabled.

6.4.2 Websites that adopted more security features

In this section, we investigate the relationship between the adoption of security features on a website and its popularity (measured by its Alexa global rank [3]), its sector (derived from McAfee's TrustedSource Web Database [24]). We expect that higher ranked popular website and websites belonging to critical sectors such as finance and online shopping, might have more incentive to adopt security features in order to protect their asset, compared to the less-known or less-valuable websites.

To confirm this hypothesis, we first use Point-biserial correlation to study the correlation between the adoption of security features in a website and its Alexa rank. Generally, the Pearson product-moment correlation coefficient (Pearson's r) is widely used in statistics as a measure of the degree of linear dependence

between two quantitative variables. In our case, the adoption of security feature is a binary choice, thus we use the Point-biserial correlation coefficient. The Point-biserial correlation coefficient is a special case of Pearson in which one variable is quantitative and the other variable is dichotomous. The result of Point-biserial correlation varies between -1 and $+1$, and a positive coefficient implies that as one variable increases, the other variable also increases and vice versa. When using Point-biserial correlation to test statistical dependence, we set the significance level to 5%. The p-value is calculated using Student's t-distribution. We accept the hypothesis only if the p-value is smaller than the significance level.

datasets	coefficient	p-value
2013 vs 2015	-0.076	3.1×10^{-29}
2015 vs 2017	-0.089	7.4×10^{-27}
2013 vs 2017	-0.098	3.2×10^{-32}

Table 6.6: The correlation between the adoption of security features in a website and its Alexa rank

We compared the three datasets in pairs to study the websites that adopted more security features over different time periods. As shown in Table 6.6, all the correlation coefficients are negative, with p-value less than 5% (hence a negative correlation). It confirms our hypothesis that higher ranked websites tend to adopt more security features. To better illustrate this correlation, for per 10,000 Alexa ranks, we calculate the percentage of websites that belongs to that rank range, which have adopted more security features in 2017 versus 2013, as shown in Figure 6.1. We can observe a downtrend for the percentage of websites that adopted more security features over the Alexa ranks.

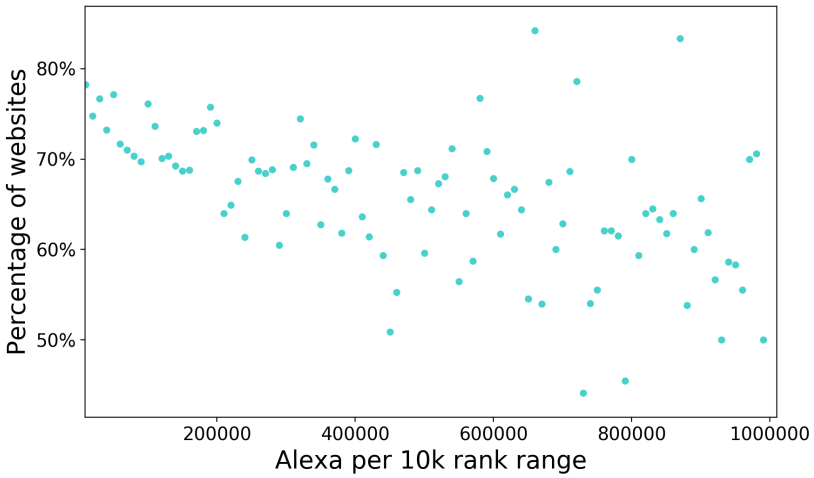

Figure 6.1: Percentage of websites that adopted more security features in 2017 versus 2013, plotted per 10k Alexa ranks

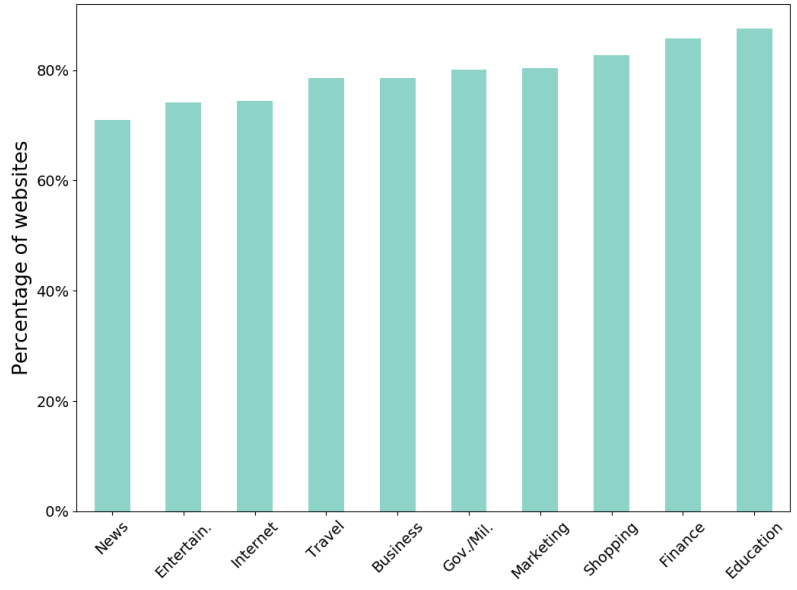

Figure 6.2: Percentage of websites that adopted more security features in 2017 versus 2013, grouped per business vertical

As for the relationship between the adoption of security features in a website and its sector, we calculate the percentage of websites that adopted more security features in each sector. Figure 6.2 shows the top 10 sectors that have larger percentage of websites adopted more security features over time. It comes as no surprise that *Education* and *Finance* are the best two performing categories. Since educational and financial organizations handle a lot sensitive personal data (students' profiles, financial transactions), they have more incentives to adopt security features for protection.

And among the 377 websites that have all seven security features enabled in 2017, 72 sites are from the *Education* sector, accounting for 10% educational websites; 63 sites are from the *Finance* sector, accounting for 11% financial websites.

6.5 Web Security Score Analysis

In the previous section, we assess to what extent the security of a particular website did improve over time, by measuring for each website if it adopts more security features over time or not. In this section, we investigate how consistently a security features is applied on a given website, by calculating ECDF-based security scores (as explained in Section 6.3), which essentially compare the usage of a security feature on a particular website with the usage on other websites in the dataset.

6.5.1 EU web security score, in terms of website popularity

To assess the web security score in terms of website popularity, the websites are grouped per 10,000 Alexa ranks, and the average score is calculated for websites that belongs to that rank range. Figure 6.3 shows the average *OverallScore* for per 10k Alexa ranks.

Figure 6.3 hints that higher ranked websites tend to have higher score. To confirm this assumption, the Spearman correlation is used to assert the correlation between the *OverallScore* in a website and its Alexa rank (as listed in Table 6.7).

Spearman's rank correlation coefficient is a nonparametric measure of the monotonicity of the relationship between two variables. It is defined as the Pearson correlation coefficient between the ranked variables. However, unlike the Pearson correlation, the Spearman correlation does not assume that both variables are normally distributed. It is a nonparametric statistic, which do

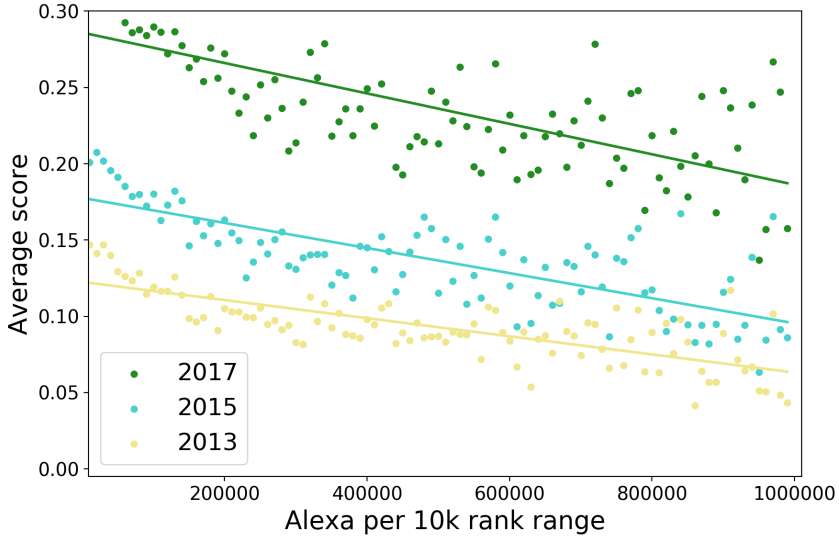

Figure 6.3: The average overall security score for per 10k Alexa ranks

not rely on assumptions that the dataset is drawn from a given probability distribution. The result of Spearman correlation varies between -1 and $+1$, and a positive coefficient implies that as one variable increases, the other variable also increases and vice versa. When using Spearman correlation to test statistical dependence, we set the significance level to 5%. The p-value is calculated using Student's t-distribution. We accept the hypothesis only if the p-value is smaller than the significance level.

As expected from Figure 6.3, there is negative correlation between the *OverallScore* in a website and its Alexa rank (see Table 6.7), and this correlation also holds for all three sub scores. This correlation is consistent with the correlation that higher ranked websites tend to adopt more security features, as we discussed in the previous section.

datasets	coefficient	p-value
Sept. 2017	-0.18	1.1×10^{-108}
Sept. 2015	-0.15	2.9×10^{-86}
Sept. 2013	-0.14	5.4×10^{-95}

Table 6.7: The correlation between the security score of a website and its Alexa rank

6.5.2 Web security score per business vertical in EU

In this section, we compare the security evolution of the websites per business vertical. For the ten most popular business vertical, the average score is calculated for websites that belongs to that business vertical. Figure 6.4 shows the average *OverallScore* for 10 business verticals, sorted by their 2013 security score, to easily identify business verticals that got better than their adjacent peers.

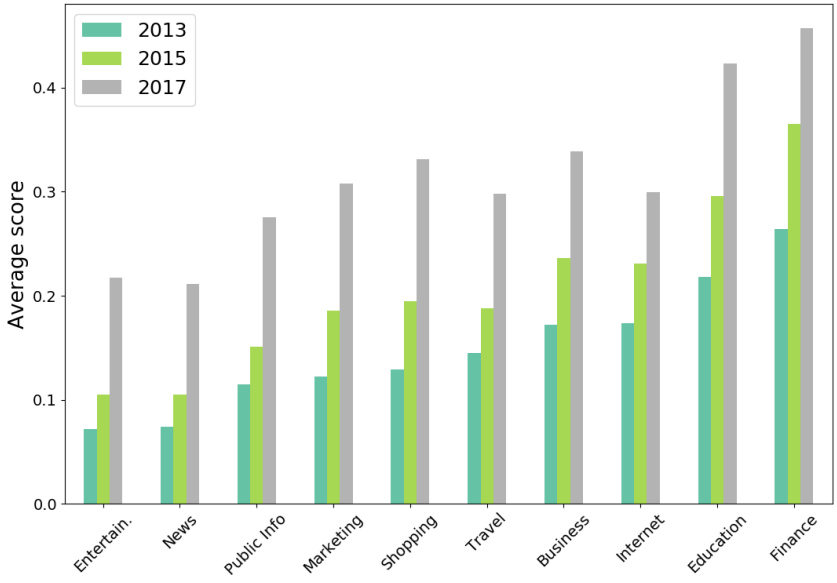

Figure 6.4: The average overall security score for each business vertical

The *Education* and *Finance* verticals are the two best performing categories, which is consistent with the finding that educational and financial organizations tend to adopt more security features (as shown in Figure 6.2 in Section 6.4.2). In addition, we can also observe that the *Business*, *Shopping*, and *Marketing* sectors improved a lot and eventually caught up their neighbors. This improvement might be driven by the fast growth of E-commerce in recent years [13]. As more companies expand their business online, and more people choose online shopping, websites from these sectors are incentivised to provide better web protection.

6.5.3 Web security score per country in EU

In this section, we compare the security evolution of the websites per country. For each EU country, the average score is calculated for websites that belongs to that country. Figure 6.7 shows the average *OverallScore* for 25 EU countries. Cyprus(.cy), Malta(.mt) and Luxemburg(.lu) were removed from the dataset, as the number of websites in these countries were less than 100. The countries in Figure 6.7 are sorted by their 2013 security score, to easily identify countries that got better than their adjacent peers.

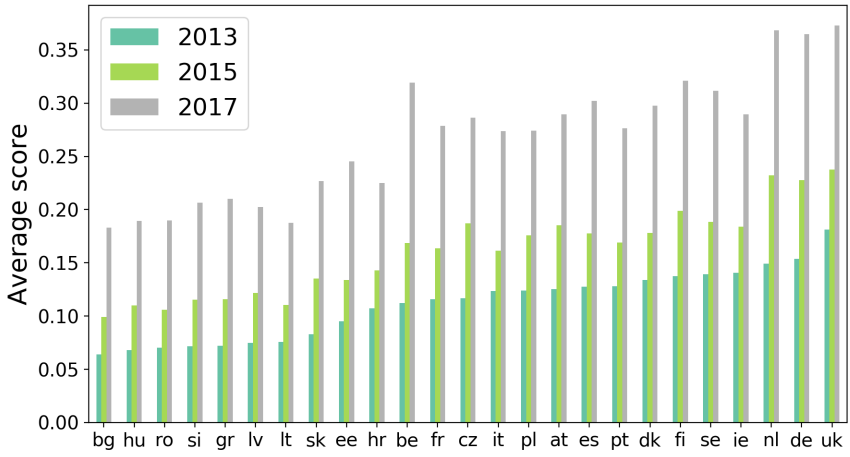

Figure 6.5: The average overall security score for each EU country

One can easily observe that the Netherlands (.nl), Germany (.de) and United Kingdom (.uk) stay ahead over the past few years. Others' positions fluctuated a bit, but there are no big changes except for Belgium (.be) which improved considerably from 2015 to 2017, making it the fourth best EU country in terms of web security performance.

6.6 HTTPS Migration Analysis

HTTPS in the standard solution for securing web traffic nowadays. Although it increases performance overhead and operating costs, the security benefits it brings outweigh these disadvantages. As previously shown in Table 6.5, more than 70% of websites have enabled HTTPS support as of September 2017. In

this section, we investigate the websites that have adopted HTTPS since 2013. We call these websites the newly adopted HTTPS sites.

To understand the types of newly adopted HTTPS sites, we plot the top 10 sectors that have the most percentage of websites with HTTPS support in Figure 6.6. One can see that *Finance* and *Education* sectors are the first movers of HTTPS adoption, with more than 50% of websites have HTTPS support in 2013. They remain the best two verticals over years, but other sectors caught up by 2017. In particular, the past two years have seen substantial improvement of HTTPS adoption in other sectors such as *Shopping* and *Real Estate*.

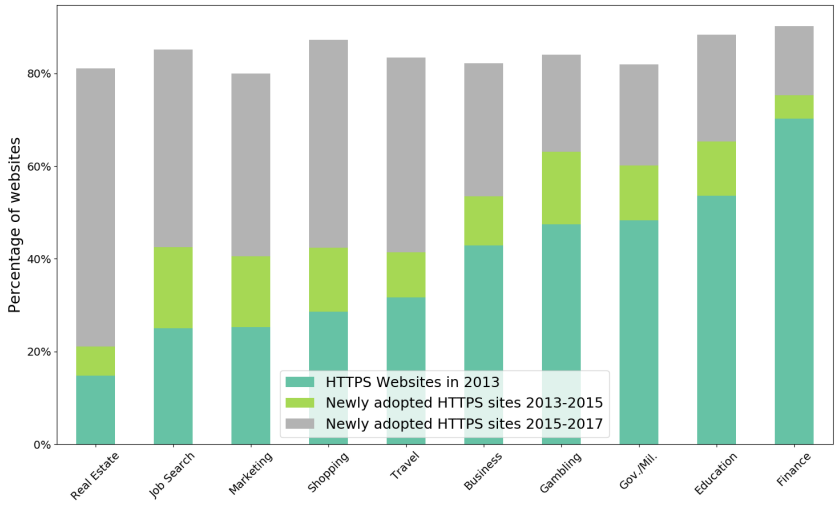

Figure 6.6: Percentage of websites in each business vertical that adopted HTTPS over time

While HTTPS already provides securing communication, it would be better to also implements `HSTS` and `Secure Cookies` to have stronger protection against Man-in-the-middle (MITM) attacks. In this section, we also investigate whether the newly adopted HTTPS sites implement `HSTS` and `Secure Cookies` as well, when migrating to HTTPS.

	HTTPS sites in 2015	Newly adopted sites 2013-2015	HTTPS sites in 2017	Newly adopted sites 2015-2017
`HSTS`	11.4%	14.1%	27.2%	38.1%
`Secure`	24.5%	32.8%	34.7%	48.4%

Table 6.8: Percentage of newly adopted HTTPS sites that enabled Secure Cookies and HSTS features

As shown in Table 6.6, 14% of newly adopted HTTPS sites from 2013 to 2015 have HSTS implemented, which is more than the overall percentage (11% of all HTTPS sites in 2015). And the use of `Secure Cookies` in newly adopted HTTPS sites is also more than the overall percentage. The same pattern can be also found for the newly adopted HTTPS sites from 2015 to 2017.

This indicates that the newly adopted HTTPS sites in recent years tend to be more security conscious than the websites having HTTPS already for a long time. In other words, the use of `Secure Cookies` and `HSTS` features occurs more often on new HTTPS websites.

6.7 Websites that Dropped Out During the Study

While our longitudinal study showed an improvement on the European web, with respect to the adoption of client-side defense mechanisms, this might partly due to the disappearance of websites during the four-year period. To exclude this bias, we selected the websites that appeared in all the three datasets shown in Table 6.1, which results a dataset that contains 13,827 websites, and re-analyzed the trend.

Table 6.9 shows the percentage of websites that adopted security features over time. Compared to figures in Table 6.5, the percentage differences are all less than 1%. And one can still observe an obvious improvement.

Security feature	% of websites		
	Sept. 2013	Sept. 2015	Sept. 2017
HTTPS Support	23.56%	34.37%	71.85%
Secure Cookies	6.07%	8.88%	24.68%
HSTS	0.43%	4.38%	19.71%
HttpOnly Cookies	37.81%	46.01%	54.82%
XCTO	2.47%	7.25%	24.43%
CSP	0.05%	0.43%	5.62%
XFO	5.13%	15.33%	31.98%

Table 6.9: Percentage of European websites that adopted security features over time

As for the web security score and its correlation with websites' popularity, the dropped websites have no impact on our conclusion. We still found a negative correlation between a website's score and its Alexa rank. To better understand the 6,320 websites that dropped out from 2013 to 2017, we plot the distribution of these websites ("Dropped set") against the original 20,147 websites ("Full set") in 2013. As shown in the following figure, the boxplots of the two datasets are very similar. This implies that the set of websites that dropped out has, more or less, the same distribution of the original dataset, with respect to the Alexa rank and web security score.

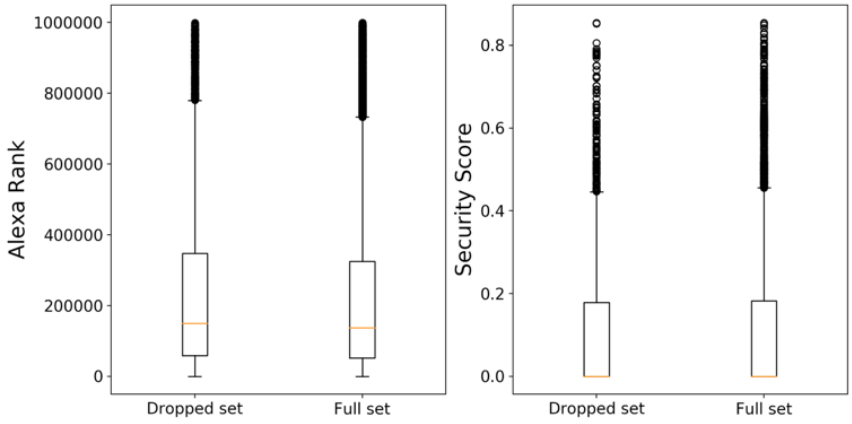

Figure 6.7: Distribution of websites that dropped out the study

As for the dispersion over countries and categories, we also found that the "Dropped set" has similar distribution to the "Full set". Thus, our dataset

remains representative over the four-year study period, despite of the dropout of some websites.

6.8 Limitations

In order to evaluate the general state of security of a website, we designed a scoring system, which is subject to three types of limitations. Firstly, the scoring system focus on the client-side security features, which may not always reflect the actual security level of a website. Although we can validate that the well-known secure websites (e.g., banking sites) have higher score than the most-likely vulnerable websites (e.g., sites with published yet unfixed XSS vulnerability) [140], the score only indicates part of a website's state of security. For a complete analysis, it would be worthwhile to have a more invasive security assessment (with website owners' consent).

Secondly, the selection of security features and the weights given to each feature are based on opinions of a relatively small expert group. While we are trying to be objective, there might be different opinions from readers. An assessor can adapt the scoring system to incorporate his own opinions. Moreover, since the scoring system is built with ECDF (Empirical Cumulative Distribution Function), it cannot give score for a single website, instead it relies on a set of sites and calculates comparative scores. Thus, having a diverse and representative dataset is important for obtaining relevent result.

6.9 Conclusion

To mitigate common web attacks, websites owners can adopt several defensive mechanisms (as introduced in Section 3.1) for protection. This chapter gives an overview of the evolution of the adoption of these security features on European Web for the past few years. More than 8 million webpages of 20,000 websites were crawled for analysis, through which we could observe the following longitudinal trends:

First, the usage of client-side security mechanisms increased over time, especially the past two years (2016 and 2017) which have seen a greater improvement than the previous two-year timeframe.

Second, the most popular websites (according to the Alexa ranking) have a higher web security metric that less popular websites. Moreover, these popular websites were adopting new security features quicker that less popular websites.

Third, by examining the websites based on their business vertical, we can state that *Education* and *Finance* are outperforming other verticals. They were the first movers of adoption of security features, and stayed ahead over the past few years.

To compare the web security level of different websites, we also proposed a web security scoring system. The scoring system can be used to establish a web security baseline among a set of websites, and this might help website operators to consider the adoption of security features.

The proposed web security scoring system is not optimal, since the weights given for each security feature were arbitrarily chosen based on their relative importance. To further understand the importance of these defensive mechanisms, we will present a correlational study in the forthcoming chapter.

Chapter 7

Correlation with Cybercrime Cost

Preamble

The contents of this chapter are adapted from the paper titled "The relationship between the cost of cybercrime and web security posture: a case study on Belgian companies" [73], which was published in the Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings in 2017. This work was done with the collaboration of other authors from KU Leuven. Ping Chen was the lead author of this paper.

This chapter presents a correlational study on the use of security features on a company's website and its cybercrime cost. As a preliminary case study, our work provides some lessons that future research in this area can draw from.

7.1 Introduction

The past few years have seen rapid adoption of client-side security mechanisms on the web (as discussed in the previous chapter). These security features are developed by security community to thwart common web attacks. In addition to reducing attack surfaces, adopting client-side security mechanisms also shows a website's security awareness, which helps to build trust with its customers. Thus, website operators are recommended to adopt these security features.

However, it is unclear whether the adoption can help organisations to reduce the actual cost of cybercrime as well.

It can be argued that the adoption of advanced security practices by a website not only secures the web presence, but is also proxy for the quality of the companies security management practice in general. So organisations with a secure presence should ultimately suffer less losses due to security incidents. To verify the hypothesis, this chapter presents a preliminary correlation study to analyse the relationship between the cost of cybercrime and web security posture.

To the best of our knowledge, this is the first work that studies this correlation. Our analysis shows that companies with better web security defences tend to have less business loss and reputation damage. The finding can motivate companies to focus on web security, and devote more attention to cybersecurity. Practical implications may also be significant - for example, it may also serve as an assessment factor when establishing cyber insurance premiums or audit costs.

7.2 Data Collection

For the analysis, we surveyed 263 Belgian companies about the impact of cybercrime on their business, and gathered the statistics on the usage of security features through website crawling.

7.2.1 Industry survey

To investigate the impact of cybercrime on businesses, we first conducted an online survey. The survey was composed of different parts. We only discuss the parts that are relevant for this paper (The full result and analysis of this industry survey can be found online [122]).

First of all, we asked several general questions to enable the categorization of the businesses based on their size, economic sector, location etc. Secondly, we asked respondents whether the business had been confronted with the cybercrime type in the past 12 months. Five types of cybercrime are surveyed, including unauthorised access to IT systems, incidents resulting in IT failure, cyber extortion, corporate espionage and internet fraud. In this paper, we only focus on unauthorised access and cyber extortion attacks, as they are much more relevant to web security than the others.

In the case of single or multiple victimization, respondents were expected, respectively for the only or the last incident, to give a specification of the incident as well as to assess the harms of the incident. Unlike other studies, the present project draws from the conviction that not all harms of cybercrime can be monetarised or even quantified. Some harms - such as the harms to individuals' dignity or harms to individuals' and entities' reputation and privacy - are inherently not quantifiable; other harms can at least in principle be expressed with a number, but the available data do not support their full monetary or quantitative estimation.

Our conceptualisation of the harms of cybercrime is inspired by Greenfield and Paoli's Harm Assessment Framework [93] and in particular follows the latter's conceptualisation of harm. Specifically, we understand harm as a violation of stakeholders' legitimate interests, thus recognising that the dominant political morality and the underlying socio-economic conditions play a central part in establishing which interests are regarded as legitimate. Following Greenfield and Paoli's Framework, we further assume that businesses - as well as the other 'bearers' identified by their taxonomy - experience harms as damages to one or more 'interest dimensions' [142]. In the case of businesses, these dimensions consist of material support, functional integrity (i.e. services to customers and internal operational activities), reputation, and privacy and autonomy. Following Greenfield and Paoli [93], who build on von Hirsch and Jareborg [142] and Sen [135], we treat these interest dimensions as representing capabilities or pathways to achieving a certain quality of life, referred to as a 'standard of living', or, by analogy, institutional mission.

	Categorical levels
business loss	$Nothing; < 1,000; < 10,000;$ $< 50,000; < 200,000; > 200,000$
reputation damage	No harm; Marginal; Moderate; Serious; Grave; Catastrophic

Table 7.1: The categorical levels of cybercrime cost

In our survey we have included questions intended to develop a monetary estimate of the harms to material support, that is, harms to the businesses' financial and material interests. Aspects of material harm that were questioned were the costs for hardware and software replacement, the regulatory fines and compensation payments, business loss, the value of the lost or stolen assets (with the exception of one cybercrime type) and the money paid to the offender (only for one cybercrime type). The respondents were asked to assess these aspects of material harm on a 6-point scale ranging from nothing to more than

200,000. In line with Greenfield and Paoli's Harm Assessment Framework [93], we have asked the respondents to assess the severity of the harms to the other 'interest dimensions' on a 6-point scale ranging from no harm to catastrophic.

7.2.2 Website crawling

We then manually checked the samples to find out the corresponding website for each survey sample, which results 263 valid websites (sites with less than 10 webpages were excluded from the dataset). After that, we used the Bing search engine [8] to obtain up to 200 webpage URLs for each website, and then a distributed crawler to visit the URLs and retrieve data from webpages (the crawling approach is explained in detail in Section 3.2).

For a comparative analysis, we also crawled more than 18,000 European websites, which are the popular websites from the 28 member states in the EU. The selection is based on the ccTLD (country code top-level domain) of each EU country. All the crawling experiments were all done in September 2016, within a week's time frame.

In order to assess a website's security posture, we analyse the usage of eight client-side security features (as introduced in Section 3.1), which can be grouped into three categories.

- **Category 1: Secure Communication** This category includes four features that contribute to secure communication between a server and a browser: HTTPS support, Secure Cookies, HTTP Strict Transport Security (HSTS), and HTTP Public Key Pining or Certificate Transparency (HPKP/CT).
- **Category 2: XSS Mitigation** This category includes three features that can be used to mitigate XSS attacks: HTTPOnly Cookies, X-Content-Type-Options (XCTO), and Content Security Policy (CSP).
- **Category 3: Secure Framing** This category has one feature X-Frame-Options (XFO) to enable secure framing.

7.3 General Findings

7.3.1 Industry survey result

The target population of the survey consisted of all the businesses based in Belgium. We constructed a sampling frame of more than 9,000 business representatives of which the contact details were provided by the Federation of Enterprises in Belgium (FEB), the largest business consortium in Belgium, and the sector federations Comeos (commerce and services) and Febelfin (banks, stock markets, credit and investment). However, about 10% of them could not be reached, due to undeliverable emails, unavailable mailboxes or expired e-mail addresses. Of the business representatives that could be reached, 453 filled out (entirely or partially) the questionnaire, which brings the initial participation rate to 4.9%. From the 453 responses, we obtained 310 valid samples.

As shown in Table 7.2, of the valid 263 samples, about 44% (118) of websites experienced business loss and reputation damage due to unauthorised access, and about 22% (55) of websites experienced business loss and reputation damage due to cyber extortion. The statistics shows that the threat of cybercrime is a real concern for organisations.

cyber attacks	business loss	reputation damage
unauthorised access	44.9%	43.7%
cyber extortion	22.1%	22.1%

Table 7.2: The cybercrime experience of 263 Belgian organisations

While advanced attackers tend to target large companies that possess strategic resources, cybercrime poses threat to small and medium-sized enterprises (SMEs) as well. As shown in Table 7.3, more SMEs experienced unauthorised access than large companies did. However, large companies had more cyber extortion experience than SMEs, which might due to their higher market value.

Company size	unauthorised access	cyber extortion
Small (< 50)	38.4%	24.2%
Medium ($50 - 250$)	26.5%	21.2%
Large (> 250)	35.1%	54.5%

Table 7.3: The cybercrime experience over different company size

7.3.2 Website crawling result

In order to compare different (groups of) website(s), we calculated ECDF scores for each website. More concretely, for each security feature, we first compute an empirical cumulative distribution function (ECDF) for all EU websites. The ECDF is computed based on the percentage of webpages having that feature on a particular website, as shown in Figure 7.1.

The ECDF score reflects how well a website is doing compared to websites in the EU dataset. For instance, if a website has a score 0.61 for the feature HTTPS, it means the website outperforms 61% of the websites in the EU dataset (i.e. by having a higher percentage of pages over HTTPS). Websites with no pages found to have a security feature are given a zero ECDF score for that feature.

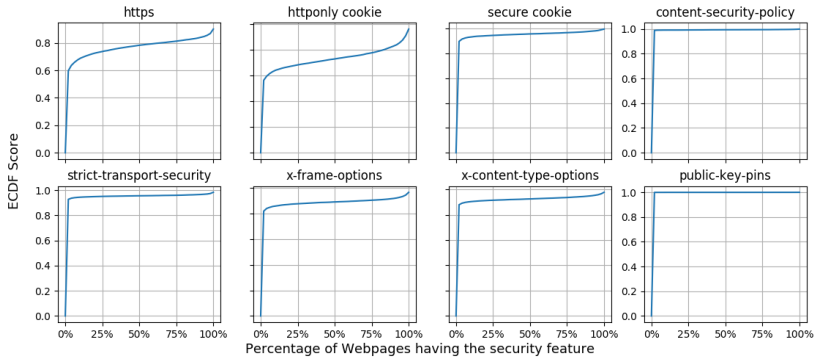

Figure 7.1: ECDFs for each security feature

Table 7.4 gives an overview on the use of security features on 263 surveyed Belgian websites and 18,731 EU websites. The percentages of EU websites that having security features enabled are greater than the surveyed Belgian websites for most features (except `XCTO` and `XFO`), this is probably due to the EU set is comprise of the popular websites, which tend to adopt more security features [69].

Security feature	% of websites		ECDF Avg. & Std. Error	
	BE	EU	BE	EU
HTTPS Support	37.3%	49.7%	0.28, 0.37	0.37, 0.38
Secure Cookies	13.0%	15.0%	0.12, 0.31	0.14, 0.33
HSTS	6.8%	10.0%	0.07, 0.24	0.10, 0.28
HPKP	0.0%	0.2%	0.00, 0.00	0.002, 0.04
HttpOnly Cookies	41.4%	50.2%	0.33, 0.40	0.38, 0.39
XCTO	25.1%	15.9%	0.23, 0.41	0.15, 0.34
CSP	2.3%	2.0%	0.02, 0.15	0.02, 0.14
XFO	27.00%	23.53%	0.24, 0.40	0.21, 0.37

Table 7.4: Overview of the use of security features on European web

However, when comparing the average of ECDF scores with the statistical z-test, there is no significant differences in the use of security features between Belgian websites and EU websites.

7.4 Correlational Analysis

To study the relation between the cost of cybercrime and web security posture, we first try to correlate the cybercrime cost with each security feature separately, and then with a combined web security score.

7.4.1 Correlation with each security feature

We use Spearman's rank correlation coefficient to analyse the relationship between the cost of cybercrime and web security posture. The result of Spearman's correlation varies between -1 and $+1$, and a positive coefficient implies that as one variable increases, the other variable also increases and vice versa. When using Spearman correlation to test statistical dependence, we set the significance level to 5%. A hypothesis is rejected if the p-value is greater than the significance level.

Table 7.5 and Table 7.6 shows the correlation between the use of seven web security features and the impact of unauthorised access and cyber extortion, respectively. The feature `HPKP/CT` is not included, since it is not found on any of the surveyed 263 websites.

	unauthorised access	
	business loss coefficient, p-value	reputation damage coefficient, p-value
HTTPS Support	-0.05, 0.611	-0.03, 0.725
Secure Cookies	-0.11, 0.232	-0.22, 0.017
HSTS	-0.04, 0.658	-0.14, 0.126
HttpOnly Cookies	-0.09, 0.351	-0.002, 0.978
XCTO	-0.13, 0.163	-0.19, 0.041
CSP	0.06, 0.531	-0.18, 0.055
XFO	-0.08, 0.414	-0.04, 0.660

Table 7.5: Spearman's rank correlation between the impact of unauthorised access and web security features

	cyber extortion	
	business loss coefficient, p-value	reputation damage coefficient, p-value
HTTPS Support	-0.24, 0.065	-0.06, 0.658
Secure Cookies	-0.30, 0.023	-0.27, 0.044
HSTS	-0.19, 0.154	-0.19, 0.149
HttpOnly Cookies	-0.27, 0.035	-0.01, 0.951
XCTO	-0.31, 0.016	-0.38, 0.003
CSP	-0.15, 0.255	-0.18, 0.178
XFO	-0.31, 0.018	-0.22, 0.093

Table 7.6: Spearman's rank correlation between the impact of cyber extortion and web security features

For cyber extortion, all the correlation coefficients are significant, which indicates that the use of client-side security features helps reduce the business loss and reputation damage. But many of them have p-values greater than significance level 5%, thus this negative correlation is not significant for some features. The same case goes for unauthorised access.

	business loss			
	β	SE	95% CI	P
Constant	-1.19	0.35	$-1.88 \sim -0.51$	0.001
HTTPS Support	0.31	0.85	$-1.35 \sim 1.98$	0.71
Secure Cookies	-3.71	3.82	$-11.2 \sim 3.78$	0.33
HSTS	-0.02	1.40	$-2.76 \sim 2.71$	0.98
HttpOnly Cookies	-0.83	0.77	$-2.36 \sim 0.68$	0.28
XCTO	-1.66	1.07	$-3.78 \sim -0.45$	0.12
CSP	4.84	3.74	$-2.49 \sim 12.18$	0.19
XFO	0.72	0.99	$-1.17 \sim 2.70$	0.44

Table 7.7: Logistic regression on the business loss due to unauthorised access over web security feature

	reputation damage			
	β	SE	95% CI	P
Constant	1.47	0.38	$0.72 \sim 2.22$	0.000
HTTPS Support	0.50	0.90	$-1.27 \sim 2.27$	0.58
Secure Cookies	-1.25	0.89	$-3.09 \sim 0.38$	0.13
HSTS	-0.23	0.81	$-2.08 \sim 1.62$	0.81
HttpOnly Cookies	0.08	0.67	$-1.22 \sim 1.38$	0.90
XCTO	-1.67	0.86	$-3.37 \sim 0.01$	0.05
CSP	-1.60	1.31	$-4.17 \sim 0.98$	0.22
XFO	1.69	0.99	$-0.25 \sim 3.62$	0.09

Table 7.8: Logistic regression on the reputation damage due to unauthorised access over web security feature

To further analyse this correlation, we dichotomised the business loss and reputation damage variables and run a logistic regression over the different security features. As shown in Table 7.7 and Table 7.8, the coefficient β for most features are negative, although many of them are not significant, they indicate a negative association between the use of security feature and the impact of unauthorised access. Table 7.9 and Table 7.10 give the logistic regression result on the impact of cyber extortion over web security features, which also indicates a negative correlation.

	business loss			
	β	SE	95% CI	P
Constant	-0.11	0.48	$-1.05 \sim 0.84$	0.83
HTTPS Support	-0.44	0.97	$-2.33 \sim 0.64$	0.27
Secure Cookies	-25.2	7.9e4	$-1.6e5 \sim 1.6e5$	1.00
HSTS	-16.7	4.2e3	$-8.3e3 \sim 8.3e3$	1.00
HttpOnly Cookies	-1.12	0.90	$-2.88 \sim 0.64$	0.21
XCTO	-1.95	1.70	$-5.28 \sim 1.38$	0.25
CSP	-5.3	2.6e6	$-5.1e6 \sim 5.1e6$	1.00
XFO	0.66	1.54	$-2.36 \sim 3.70$	0.67

Table 7.9: Logistic regression on the business loss due to cyber extortion over web security feature

	reputation damage			
	β	SE	95% CI	P
Constant	2.54	0.88	$0.81 \sim 4.27$	0.004
HTTPS Support	0.08	1.66	$-3.16 \sim 3.32$	0.96
Secure Cookies	-1.62	1.72	$-4.99 \sim 1.76$	0.35
HSTS	0.12	1.53	$-2.88 \sim 3.13$	0.93
HttpOnly Cookies	-0.38	1.28	$-2.90 \sim 2.13$	0.76
XCTO	-4.53	3.13	$-10.7 \sim 1.60$	0.15
CSP	-0.02	1.47	$-2.90 \sim 2.86$	0.99
XFO	5.07	3.46	$-1.70 \sim 11.8$	0.14

Table 7.10: Logistic regression on the reputation damage due to cyber extortion over web security feature

7.4.2 Correlation with overall security score

In the previous section, we found a negative correlation for some security features in some cases, through a series of detailed analysis. For website owners, it would be better to have a simple conclusion regarding the issue. Thus, we utilise the web security scoring system presented in the previous chapter (Section 6.3), with some changes, to calculate a security score and analyse the overall correlation.

In Section 6.3, we arbitrarily assign the weight to each feature based on maturity and importance. In this section, we reassign the weights according to the correlational findings in the previous section. Higher weights were given to features with more significant correlation with cybercrime cost.

In particular, the following weights are used to calculated the three subscores:

Secure Communication Score. This subscore is measured by applying a weighted average of the HTTPS, HSTS, and Secure Cookies usage.

$$\begin{aligned} SecureCommunicationScore = &\frac{30}{100} \times HTTPS \\ &+ \frac{50}{100} \times SecureCookies \\ &+ \frac{20}{100} \times HSTS \end{aligned}$$

XSS Mitigation Score. This subscore measured by applying a weighted average of the HttpOnly Cookies, XCTO, and CSP usage.

$$\begin{aligned} XSSMitigationScore = &\frac{50}{100} \times HttpOnlyCookies \\ &+ \frac{10}{100} \times XCTO \\ &+ \frac{40}{100} \times CSP \end{aligned}$$

Secure Framing Score. This subscore is measured by the XFO usage.

$$SecureFraming = \frac{100}{100} \times XFO$$

An overall web security score (*OverallScore*) can be obtained by combine the above three subscores:

$$\begin{aligned} \textbf{OverallScore} = &\frac{30}{100} \times SecureCommunicationScore \\ &+ \frac{60}{100} \times XSSMitigationScore \\ &+ \frac{10}{100} \times SecureFramingScore \end{aligned}$$

We then analyse the correlation between the obtained *OverallScore* and the cost of cybercrime, as shown in Table 7.11. The negative coefficients implies that the higher the *OverallScore* is, the lower the costs of cybercrime is. For unauthorised access, the p-values are around 10%, which indicates this correlation is not very significant. However, the correlation between cyber extortion and web security score significantly holds.

Spearman's rank correlation		β	P
unauthorised access	business loss	-0.16	0.089
	reputation damage	-0.15	0.106
cyber extortion	business loss	-0.43	0.0008
	reputation damage	-0.27	0.037

Table 7.11: Correlation between the cost of cybercrime and web security score

7.5 Representativeness of the Samples

For our industry survey, we received 453 responses. The response rate is low, but not much lower than the participation rate of the few other studies that explicitly report such rate[1]: in a survey on computer crime, done by Computer Security Institute [126], for example, 6.4% of the contacted businesses participated. And our sample is bigger than those of two earlier studies from PWC Belgium [80] that provided preliminary data on the impact of cybercrime on Belgian businesses.

The majority of the businesses that took part in the survey have their headquarters in Flanders (62%). Further, Brussels account for 21% of the sample, and Wallonia 14%. The number of businesses whose headquarters is outside Belgium is considerably lower, amounting to 3.6% of the sample. By comparing these figures with the official data [85] from Belgian Ministry of the Economy, we note that the percentage of the Flanders-based businesses taking part in the survey corresponds to the official figure (61%). However, there is an overrepresentation of the Brussels-based businesses, as they effectively count only for 11%, and an underrepresentation of those based in Wallonia.

The businesses taking part in the survey belong to many different economic sectors, but many sectors, and the related sector federations, are only represented once or twice in our sample. The sectors most strongly represented in our sample of respondents are the following: technology (23%), the chemical and life sciences (10%), and commerce and services (10%). Due to the low number of representatives of many sector federations, we could not make the analysis of the incidence or impact of cybercrime per sector.

As for the size of the businesses, we distinguish between small, medium and large businesses, based on staff headcount, following the standard classification of the European Commission [78]. In our sample, around half of the businesses are small (52%), the rest of the sample being almost equally distributed amongst

[1]Most of the studies about the costs or harms of cybercrime do not provide information on the participation/response rate or the number of contacted units, but only report the number of respondents.

medium (22%), or large (27%) businesses. Comparing these figures with the official data [85] from Belgian Ministry of the Economy, we note that our sample is not representative for the size: according to the Ministry's data, 99% of all the persons and entities liable for VAT are small, 0.6% are medium and 0.2% are large.

For the distribution of our samples over Alexa's rank ranges, we observed that 27.4% of them are listed in top 1 million global Internet domains, 31.9% of them are ranked between 1 million and 10 million, and the rest of the samples (40.7%) are either listed beyond top 10 million or not indexed by Alexa. The dataset seems skewed towards lower-ranked sites, this is mainly due to the fact that the majority of the samples are small local businesses in Belgium, which have relatively lower traffic.

7.6 Limitations

As the first attempt to investigate the relationship between the cost of cybercrime and web security posture, this paper has some limitations. Firstly, the dataset is not optimal. It consists of organizations from Belgium only, and the sample size (263 companies) is relatively small. Furthermore, the business loss is measured in absolute amount of money, instead of measuring as the percentage of business revenue and security investment.

Secondly, our study only covers a short time period (one year), which prevents us to investigate the change over time. It might obviously be worthwhile to have a longitudinal study, working more closely with participated organizations to identify trends, and have more conclusive results. Despite these limitations, we hope our preliminary case study can provide some guidance for future research in this area.

7.7 Conclusion

Cybercrime is a growing threat to business [100], but many organisations fail to take it seriously. Anderson et al. [57] provided the first systematic study of the costs of cybercrime, distinguishing different types of cybercrime, and estimating the financial cost for each type. As the web gets rapidly integrated into business, it is also increasingly leveraged by the cybercriminals. While academia has been advocating improving web security, there is little effort in correlating the web security with the cost of cybercrime.

In this chapter, we have investigated the current status of cybercrime impact on Belgian companies, and the usage of web security features. By correlating the use of security features with the cost of cybercrime, we found a negative correlation between them. In other words, companies should adopt more web security features, which helps them to mitigate web attacks and minimise damages caused by cybercrime.

Chapter 8

Conclusion

While the web was originally developed as an information-sharing tool for scholars, it has risen to unprecedented levels over the past two decades, and it is continuously developing. Nowadays, the web has become an essential tool for both business and individuals. More and more people rely on the web for almost all kinds of activities, such as acquiring knowledge, getting information, shopping and networking. Companies are also increasingly depending on the web to conduct various kinds of business (e.g., marketing, e-commerce).

Meanwhile, the past decades have also seen the rise of cyber threats on the web. Cybercriminals have been launching attacks against business and individuals, which inflicted severe damage and harm to our society. The problem of widespread web attacks is partly due to the lack of adequate protection on many websites. Many web attacks can be prevented (or at least be alleviated), if websites owners regularly assess their security posture and implement known defensive mechanisms to improve security protection.

To have a broad view of the current state of security protections on the web, this thesis has focused on assessing websites security by investigating the use of client-side defensive mechanisms. We proposed an approach to measure how well a web application is protected from an outside perspective, and demonstrated its effectiveness through several large-scale assessments. In this chapter, we first revisit our approach and findings with a brief summary, and then give an overview of related works from both academia and industry. Lastly, we present some concluding thoughts.

8.1 Summary

The web is constantly evolving, so do the web attackers. The past decade has seen a shift in web attackers' target. Traditionally, attackers mostly focused on exploiting web servers, aiming to gain control over server machines. Typical examples of such attacks are SQL Injection and Command Injection. As more and more features are deployed at the client side, the attackers started targeting client machines, attempting to obtain sensitive information from a user and performing actions in the name of the user in a web application. Well-known examples of such attacks are Cross-Site Scripting (XSS) and Session Hijacking.

In respond to this trend, security defence also shifts towards the client. Various client-side security mechanisms have been developed to counter client-side attacks. For example, Content Security Policy (CSP) is proposed to prevent XSS. In Chapter 3, we enumerated eight client-side defense mechanisms that websites can adopt to prevent common attacks: HTTPS Support, HTTP Strict-Transport-Security, Public Key Pinning and Certificate Transparency, HttpOnly and Secure Cookies, Content Type Options, and Content Security Policy.

Although these mechanisms are deployed on and enforced by the client side, they are specified and sent by the web server. Thus, the presence of client-side security mechanisms on a website can be used as an indicator of the website's security level. Based on this assumption, we designed a large-scale web crawling approach to assess websites by detecting the use of client-side security mechanisms. As explained in more detail in Chapter 3, our approach leveraged Bing Search [8] to get popular pages of a site, and used a customized headless browser to visit the pages. By doing this, we try to mimic the behavior of a normal user visiting a website.

This large-scale web crawling approach was first used to survey the prevalence of mixed-content issue in Chapter 4. While the issue of mixed HTTP content on HTTPS websites is not new, our work was the first attempt to systematically analyze the issue. Our study showed that almost half of HTTPS websites are vulnerable, and more than half of these vulnerable pages allow the attacker can execute arbitrary JavaScript once successfully exploited. A common mitigation technique employed by web browser is to block the execution of mixed content. When this work was published, most of the mobile browsers lack protection against this issue. As of 2018, all major web browsers (both desktop and mobile versions) block mixed content by default.

We then employed the crawling approach to demonstrate the effectiveness of large-scale external assessment based on the use of client-side security mechanisms. In Chapter 5, we presented an assessment for the top 10,000 Chinese websites, and observed that the majority of Chinese websites lack

support for client-side security policies. Although the situation got better over the past few years (with particular noticeable improvement on HTTPS adoption), the overall adoption rate is still lower when compared to the global web. Our study also identified a severe privacy issue that is unique in China, 6% of websites are inadvertent leaking private identity information.

Next, in Chapter 6, we expand the assessment over a four-year timeframe, examining the longitudinal trend of the adoption of client-side security mechanisms on more than 20,000 European websites. Our assessment showed that the adoption of defence mechanisms increased over time, and the popular websites were adopting new security features quicker that less popular websites. To quantify web security protection, a web security scoring system is proposed. Using the scoring system, we compared the security postures of websites from different countries and sectors. Unsurprisingly, we found that the most popular websites have higher scores and websites from *Education* and *Finance* sector are outperforming others.

In order to optimize the web security scoring system in Chapter 6, and have a further understanding of client-side security mechanisms, we investigated the relationship between a company's cybercrime cost and the adoption of defence mechanisms on its website in Chapter 7. Through a correlational case study on 263 Belgian companies, we found a negative correlation between them. Although the correlation is not very strong, it confirmed that our approach of assessing websites security externally by detecting client-side security mechanisms is useful. It helps to motivate websites owners to invest more on security protection. Additionally, it may also serve as an assessment factor when establishing cyber insurance premiums or audit costs.

Nowadays, the client side (a user's browser) has become increasingly powerful, which takes over many features and functionalities that was conventionally managed by the web server. This shift towards the client side enables a new approach to security, namely using security policies to instruct web browsers to enforcement defence mechanisms. By adopting client-side security mechanisms, website owners are not only protecting their customers, but also demonstrating their security awareness to outsiders. Assessing websites security from the aspect of adopting defensive mechanisms enables efficiency and scalability, which is desirable to outside auditors, such as regulatory authority and supervisory organizations.

8.2 Related Work

As the Web becomes ubiquitous, it is imperative to ensure the security and correctness of Web applications. A variety of methods have been proposed to test web applications in the past two decades [110], and most of them are designed to detect specific vulnerabilities and errors such as SQL injection and XSS attacks. One of the commonly used techniques is active scanning, which automatically finds known vulnerabilities based on defined rules and patterns. As an example, Kals et al. developed the SecuBat tool [104] in 2006, which automatically finds exploitable SQL injection and XSS vulnerabilities.

Nowadays, web scanning tools have become much more sophisticated, and there are various commercial products available on the market [40]. These tools are typically used by penetration testers to assess a specific web application, with permission granted by the website owner. Due to their aggressiveness and laboriousness, vulnerability scanners are not suitable for large-scale assessment. To illustrate the impact of certain vulnerability, security researchers often limited their analyze to just a couple of major popular websites in 2000s [104, 153].

With the advancement of web crawling techniques, the past few years have seen more larger-scale assessment [116, 109, 136, 66, 106]. For instance, Nikiforakis et al. presented a large-scale analysis of remote JavaScript inclusions for more than three million pages of the top 10,000 Alexa websites in 2012 [116]. In 2016, Cahn et al. conducted an empirical study of web cookies for the top 100,000 Alexa websites [66]. Kumar et al. performed a large-scale detection of web dependencies on third-party resources (e.g., images, scripts, etc.) for the Alexa top 1 million websites in 2017 [106].

These large-scale assessments typically explicate certain vulnerability or weakness of web applications. In contrast, this thesis has focused on analysing the client-side defence mechanisms, instead of examining vulnerabilities. The past few years have seen increasing attention towards client-side security in literature. De Ryck et al. provided a detailed introduction on this topic in a book called “Primer on Client-Side Web Security” [131]. The book discussed various client-side vulnerabilities and attacks, and enumerated best practices with existing countermeasures and emerging mitigation techniques. In a recent study by Stock et al. [138], the authors presented a historical perspective on client-side security, by examining a large corpus of archived web documents.

To detect the use of client-side defence mechanisms, one can manually inspect the HTTP responses in a browser. There are also free online tools that allow people to analyse the strength of a server’s SSL/TLS implementation [36], and to check security headers of a specific webpage [4]. This thesis proposed crawling approach for large-scale assessment and presented security evaluation

of websites in China and EU. A similar assessment for a specific demographic area is presented in [55], where Alarifi et al. evaluated the security of popular Arabic websites, by analyzing malicious webpages from 7,000 domains using web scanner APIs like Google Safe Browsing [14]. Another regional web security assessment is presented in [134], where the authors analyzed the malicious servers in the .nz domain.

8.3 Recent Development of Client-side Defenses

As web technology evolves, the landscape of cyber threats also changes over time [87, 121]. Meanwhile, the client-side countermeasures evolve as well. For example, Public Key Pinning is phasing out and being replaced by Certificate Transparency (as introduced in Section 3.1). Since our web security scoring system is based on the use of client-side security mechanisms, it can also be adapted to reflect changes over time [1]. In this section, we discuss the recent development on client-side countermeasures.

While Content Security Policy (CSP) was originally proposed to mitigate Cross-site scripting (XSS), its directives are expanding to cover more security issues. For example, the `frame-ancestors` directive has been proposed to prevent clickjacking attack, which has the same effect as the `X-Frame-Options` (XFO)+ header. And W3C Working Group is considering adding a new `cookie-scope` [145] directive to CSP, in order to achieve similar goals as HttpOnly and Secure Cookies. Since CSP is under active development and some browsers do not yet support all the directives, these new CSP features should only be used as a defense-in-depth to compliment the primary security mechanisms.

CSP allows a website to restrict the loading and execution of malicious resources on its pages. However, it does not apply the restrictions to third-party content loaded in via `<iframe>`. To be able to place restrictions on the embedded content, an embedder can use Embedded CSP [146] to negotiate a policy with the embedded-content provider. Embedded CSP proposes a Content Security Policy as an attribute on the `<iframe>` element. This policy is transmitted along with the HTTP request for the framed content in an `Embedding-CSP` header. A browser renders the embedded content only if the embedded-content provider accept the proposed policy (by returning the policy in a `Content-Security-Policy` header along with the response). As of August

[1] The longitudinal study presented in this thesis covers a four-year period, but our scoring system remained the same. This is because we have to maintain consistency when comparing the security performance of websites over time.

2018, Embedded CSP is still a W3C Working Draft [146], with support from Chrome (version 61+ [51]) only.

Referrer Policy [86] is a new mechanism that controls the referrer information sent in the Referer header. It can be set via the `Referrer-Policy` HTTP header, or the referrer keyword in a `<meta>` element, or the referrer policy attribute in an `<a>` element. It can prevent sensitive information in the URL, for example, the user's session identifier in the URL, from leaking via referrer headers. Currently, Referrer Policy is a W3C Candidate Recommendation [86]. It has already received full support from Chrome and Firefox, and partial support from Edge and Safari [10].

Similar to HTTP Public Key Pinning (HPKP), which pins a cryptographic hash to a server, Subresource Integrity (SRI) [54] pins a cryptographic hash to a resource. While HPKP ensures the server is genuine (no attackers in between), it cannot guarentee the fetched resource has been delivered without unexpected manipulation. If an attacker gains access to the server, it can manipulate content with impunity. To address this issue, SRI allows a website to specify the expected hash as integrity attribute in the `<link>` and `<script>` element. The browser will load and execute the resource only if the fetched script matches the expected hash. SRI is a W3C Recommendation, and it has already been supported in most browsers (except IE).

Modern browsers provide an expanding set of features and APIs for websites to offer richer functionality. For example, the `geolocation` feature allow a web application to obtain users' location information. Feature Policy [75] is mechanism that allows websites to selectively enable and disable various browser features and APIs. Feature Policy can be set in the `Feature-Policy` HTTP header and the `allow` attribute in `<iframe>` element. By specifying a feature policy to disable access to certain browser features, a website can prevent own and third-party content from introducing unwanted behaviors within its application. Currently, Feature Policy is still a W3C Working Draft. It is supported in Chrome, and partially supported in Safari.

HTTP cookie is an important mechanism for web applications to store users' session information. By default, cookies are automatically sent with every request to a website. This default behaviour might be abused by attackers to perform cross-site request forgeries (CSRF) attacks, i.e., to force a user to perform unwanted actions on the site where they are logged in. To prevent CSRF attacks and mitigate the risk of cross-origin information leakage, a website can use the `Same-Site` [149] attribute. There are two possible values for the `Same-Site` attribute: "Lax" and "Strict". In the strict mode, the cookie is withheld with any cross-site usage. As of August 2018, Same-site cookies has been supported in the latest version of major browsers.

8.4 Concluding Thoughts

This thesis proposed a simple yet effective approach to assess websites security from an outside perspective. The proposed approach allows people to compare the security levels of a group of websites. As ensuring web security is becoming more and more important, this approach come in handy for both website operators and supervisory bodies. They can employ it to regularly monitor a large number of websites and have a broad comparative view. This kind of large-scale assessments will spur websites to put more effort on security.

In our work, we tried to validate that the efforts on security are worthwhile by correlating cybercrime cost with websites security. While we found some facts to support this view, our case study is limited. It only provides an correlational analysis at a certain point in time. Further research can focus on finer-grained and longitudinal studies, i.e., to have a continuous follow-up over the surveyed companies in order to see changes over time. Moreover, as web technologies evolve, new defensive features might appear. They should also be included as part of the web scoring metrics.

By quantifying a website's security level as a web security score, we compared websites based on country, sector and popularity (Alexa ranking). For future work, we can include other properties of a website for comparison, such as a site's domain age and a company's annual revenue. Furthermore, as different countries have different regulations on a company's responsibility to protect customers' data, it might also be worthwhile to look for regional differences between EU, US and China.

Despite these limitations, our work shed some light on efficient large-scale web security assessment. It has practical applicability for government and companies, and it also provides valuable lessons for further research in this area.

Bibliography

[1] The 20 million hotel reversation records (in chinese). http://net.chinabyte.com/2/12850502.shtml.

[2] 2000 students' id number and bank acount leaked (in chinese). http://tech.sina.com.cn/i/2012-03-23/07326867582.shtml.

[3] Alexa top 1 million sites. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[4] Analyse your HTTP response headers. https://securityheaders.com/.

[5] Baidu SEO Guide V2.0 (in Chinese). http://baiduseoguide.com.

[6] Baidu starts to index HTTPS websites (in Chinese). http://ziyuan.baidu.com/wiki/392.

[7] BeEF - The Browser Exploitation Framework Project. http://beefproject.com/.

[8] Bing Web Search API. https://azure.microsoft.com/en-us/services/cognitive-services/bing-web-search-api/.

[9] Blink - The Chromium Projects. https://www.chromium.org/blink.

[10] Can I use Referrer Policy. https://caniuse.com/#feat=referrer-policy.

[11] Celery: Distributed Task Queue. http://www.celeryproject.org/.

[12] Data privacy: Public pushback. https://www.economist.com/node/21735613.

[13] E-commerce: The new bazaar. https://www.economist.com/node/21730546.

[14] Google Safe Browsing. https://safebrowsing.google.com/.

[15] HSTS Preload List Submission. https://hstspreload.org/.

[16] HTML Purifier. http://htmlpurifier.org/.

[17] HtmlUnit. http://htmlunit.sourceforge.net/.

[18] IE8 Security Part V: Comprehensive Protection. https://blogs.msdn.microsoft.com/ie/2008/07/02/ie8-security-part-v-comprehensive-protection/.

[19] Iframes security summary. http://www.thespanner.co.uk/2007/10/24/iframes-security-summary/.

[20] Intent To Deprecate And Remove: Public Key Pinning. https://www.certificate-transparency.org/.

[21] Internet Explorer Architecture. https://msdn.microsoft.com/en-us/library/aa741312(v=vs.85).aspx.

[22] KNET Trusted Website Database (in Chinese). http://t.knet.cn/.

[23] Let's Encrypt - Free SSL/TLS Certificates. https://letsencrypt.org/.

[24] Mcafee trustedsource web database. https://www.trustedsource.org/en/feedback/url.

[25] Mixed content checker online. https://mixed-content.info/mixed-content-checker-ligne/.

[26] MongoDB. https://www.mongodb.com/.

[27] NCSA MOSAIC. http://www.ncsa.illinois.edu/enabling/mosaic.

[28] NoScript - JavaScript/Java/Flash blocker for a safer Firefox experience! https://noscript.net/.

[29] "only secure content is displayed" notification in internet explorer 9 or later. http://support.microsoft.com/kb/2625928.

[30] OWASP Java Encoder Project. https://www.owasp.org/index.php/OWASP_Java_Encoder_Project.

[31] Phantomjs: Headless webkit with javascript api. https://www.phantomjs.org/.

[32] Preloading HSTS. https://blog.mozilla.org/security/2012/11/01/preloading-hsts/.

[33] RabbitMQ. https://www.rabbitmq.com/.

[34] Ssl check - scan your website for non-secure content. https://www.jitbit.com/sslcheck/.

[35] SSL Pulse. https://www.trustworthyinternet.org/ssl-pulse/.

[36] SSL Server Test. https://www.ssllabs.com/ssltest/.

[37] sslyze. https://github.com/iSECPartners/sslyze.

[38] The birth of the web. https://home.cern/topics/birth-web.

[39] Understanding the TLS Renegotiation Attack. http://www.educatedguesswork.org/2009/11/understanding_the_tls_renegoti.html.

[40] Vulnerability Scanning Tools. https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools.

[41] WebKit - Open Source Web Browser Engine. https://webkit.org/.

[42] World Internet Users in the World. http://www.internetworldstats.com/stats.htm.

[43] Directive 95/46/EC on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Union* (1995).

[44] Internet Explorer 8 Mixed Content Handling. http://msdn.microsoft.com/en-us/library/ee264315(v=vs.85).aspx, 2009.

[45] New Chromium security features, June 2011. https://blog.chromium.org/2011/06/new-chromium-security-features-june.html, 2011.

[46] Ending mixed scripting vulnerabilities. http://blog.chromium.org/2012/08/ending-mixed-scripting-vulnerabilities.html, 2012.

[47] Mixed content blocking enabled in firefox 23! https://blog.mozilla.org/tanvi/2013/04/10/mixed-content-blocking-enabled-in-firefox-23/, 2013.

[48] Is HTTP Public Key Pinning Dead? https://blog.qualys.com/ssllabs/2016/09/06/is-http-public-key-pinning-dead, 2016.

[49] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union* (2016).

[50] Using security features to do bad things. https://scotthelme.co.uk/using-security-features-to-do-bad-things/, 2016.

[51] CSP: Embedded Enforcement - Chrome Platform Status. https://www.chromestatus.com/feature/5750241810710528, 2017.

[52] Intent To Deprecate And Remove: Public Key Pinning. https://groups.google.com/a/chromium.org/forum/#!msg/blink-dev/he9tr7p3rZ8/eNMwKPmUBAAJ, 2017.

[53] Akhawe, D., Barth, A., Lam, P. E., Mitchell, J., and Song, D. Towards a formal foundation of web security. In *Computer Security Foundations Symposium (CSF), 2010 23rd IEEE* (2010), IEEE, pp. 290–304.

[54] Akhawe, D., Braun, F., et al. Subresource Integrity. *W3C Recommendation* (2016).

[55] Alarifi, A., and AI-Salman, A. Security analysis of top visited arabic web sites. In *15th International Conference on Advanced Communication Technology* (2013), IEEE.

[56] Amrutkar, C., Traynor, P., and van Oorschot, P. C. Measuring ssl indicators on mobile browsers: extended life, or end of the road? In *Proceedings of the 15th International Security Conference* (2012), ISC '12, Springer, pp. 86–103.

[57] Anderson, R., Barton, C., Böhme, R., Clayton, R., van Eeten, M. J. G., Levi, M., Moore, T., and Savage, S. *Measuring the Cost of Cybercrime*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 265–300.

[58] Barth, A. HTTP State Management Mechanism. *IETF RFC 6265* (2011).

[59] Barth, A., Jackson, C., and Mitchell, J. C. Robust defenses for cross-site request forgery. In *Proceedings of the 15th ACM conference on Computer and communications security* (New York, NY, USA, 2008), CCS '08, ACM, pp. 75–88.

[60] Barth, A., Jackson, C., and Mitchell, J. C. Securing frame communication in browsers. *Communications of the ACM 52*, 6 (2009), 83–91.

[61] Belshe, M., and Peon, R. Hypertext Transfer Protocol Version 2 (HTTP/2). *IETF RFC 5280* (2015).

[62] Berners-Lee, T., and Connolly, D. Hypertext Markup Language - 2.0. *IETF RFC 1866* (1995).

[63] Berners-Lee, T., Fielding, R., and Masinter, L. Uniform Resource Identifier (URI): Generic Syntax. *IETF RFC 3986* (2005).

[64] Berners-Lee, T., Masinter, L., and McCahill, M. Uniform Resource Locators (URL). *IETF RFC 1738* (1994).

[65] Butler, E. Firesheep. `https://codebutler.github.io/firesheep/`, 2010.

[66] Cahn, A., Alfeld, S., Barford, P., and Muthukrishnan, S. An empirical study of web cookies. In *Proceedings of the 25th International Conference on World Wide Web* (2016), International World Wide Web Conferences Steering Committee, pp. 891–901.

[67] CERNET. Evolution of Internet in China. `http://www.edu.cn/introduction_1378/20060323/t20060323_4285.shtml`, 2001.

[68] Chen, P., Desmet, L., and Huygens, C. *A Study on Advanced Persistent Threats*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 63–72.

[69] Chen, P., Desmet, L., Huygens, C., and Joosen, W. Longitudinal study of the use of client-side security mechanisms on the european web. In *Proceedings of the 25th International Conference Companion on World Wide Web* (2016), WWW '16 Companion, International World Wide Web Conferences Steering Committee, pp. 457–462.

[70] Chen, P., Huygens, C., Desmet, L., and Joosen, W. Advanced or not? a comparative study of the use of anti-debugging and anti-vm techniques in generic and targeted malware. In *IFIP International Information Security and Privacy Conference* (2016), Springer, pp. 323–336.

[71] Chen, P., Nikiforakis, N., Desmet, L., and Huygens, C. Security analysis of the chinese web: How well is it protected? In *Proceedings of the 2014 Workshop on Cyber Security Analytics, Intelligence and Automation* (New York, NY, USA, 2014), SafeConfig '14, ACM, pp. 3–9.

[72] Chen, P., Nikiforakis, N., Huygens, C., and Desmet, L. A Dangerous Mix: Large-scale analysis of mixed-content websites. In *16th Information Security Conference* (2013), ISC '13.

[73] Chen, P., Visschers, J., Verstraete, C., Paoli, L., Huygens, C., Desmet, L., and Joosen, W. The relationship between the cost of cybercrime and web security posture: A case study on belgian companies. In *Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings* (New York, NY, USA, 2017), ECSA '17, ACM, pp. 115–120.

[74] Clark, J., and van Oorschot, P. C. SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements. In *IEEE Symposium on Security and Privacy* (2013), SP '13, pp. 511–525.

[75] Clelland, I. Feature Policy. *W3C Draft Community Group Report* (2018).

[76] CNNIC. The 40th China Statistical Report on Internet Development (in Chinese), 2017.

[77] CNNIC, and APAC. Global Chinese Phishing Sites Report. `http://www.cnnic.cn/gywm/xwzx/rdxw/rdxx/201305/W020130531616450986485.pdf`, 2013.

[78] Commission, E. Commission recommendation of 6 may 2003 concerning the definition of micro, small and medium-sized enterprises, 2003.

[79] Cooper, D., Santesson, S., , Farrell, S., et al. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. *IETF RFC 5280* (2008).

[80] Coopers, P. Redefining the security culture a better way to protect your business. `https://www.pwc.be/en/documents/20170315-Information-security-breaches-survey.pdf`, 2017.

[81] Dacosta, I., Chakradeo, S., Ahamad, M., and Traynor, P. One-time cookies: Preventing session hijacking attacks with stateless authentication tokens. *ACM Transactions on Internet Technology (TOIT) 12*, 1 (2012), 1.

[82] Dierks, T., and Allen, C. The TLS Protocol Version 1.0. *IETF RFC 2246* (1999).

[83] Duong, T., and Rizzo, J. *Here Come The ⊕ Ninjas*, 2011.

[84] Durumeric, Z., Kasten, J., Adrian, D., Halderman, J. A., Bailey, M., Li, F., Weaver, N., Amann, J., Beekman, J., Payer, M., et al. The matter of heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (2014), ACM, pp. 475–488.

[85] Economie, F. Aantal actieve btw-plichtige ondernemingen volgens werknemersklasse en plaats maatschappelijke zetel, meest recente jaar. https://bestat.economie.fgov.be/bestat/crosstable.xhtml?view=9d19ebe2-f35a-4b51-ac1a-c153e6d77d67, 2016.

[86] Eisinger, J., and Stark, E. Referrer Policy. *W3C Candidate Recommendation* (2017).

[87] ENISA. Enisa threat landscape report 2017. https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2017, 2017.

[88] Evans, C., Palmer, C., and Sleevi, R. Public key pinning extension for HTTP. *IETF RFC 7467* (2015).

[89] Fazzini, M., Saxena, P., and Orso, A. Autocsp: automatically retrofitting csp to web applications. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* (2015), vol. 1, pp. 336–346.

[90] Fielding, R., Gettys, J., Mogul, J., et al. Hypertext Transfer Protocol – HTTP/1.1. *IETF RFC 2616* (1999).

[91] Finley, K. Half the Web Is Now Encrypted. That Makes Everyone Safer. https://www.wired.com/2017/01/half-web-now-encrypted-makes-everyone-safer/, 2017.

[92] Garrett, J. J. Ajax: A New Approach to Web Applications. http://adaptivepath.org/ideas/ajax-new-approach-web-applications/.

[93] Greenfield, V. A., and Paoli, L. A framework to assess the harms of crimes. *The British Journal of Criminology 53*, 5 (2013), 864.

[94] Grosskurth, A., and Godfrey, M. W. Architecture and evolution of the modern web browser. *Preprint submitted to Elsevier Science 12*, 26 (2006), 235–246.

[95] Guardian, T. Edward Snowden. https://www.theguardian.com/us-news/edward-snowden, 2013.

[96] Heiderich, M., Niemietz, M., Schuster, F., Holz, T., and Schwenk, J. Scriptless attacks: stealing the pie without touching the sill. In *Proceedings of the 2012 ACM conference on Computer and communications security* (New York, NY, USA, 2012), CCS '12, ACM, pp. 760–771.

[97] Hodges, J., Jackson, C., and Barth, A. HTTP strict transport security (HSTS). *IETF RFC* (2012).

[98] Hoffman, P., and Schlyter, J. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. *IETF RFC 6698* (2012).

[99] Hoogstraaten, H., Prins, R., et al. Black Tulip: Report of the investigation into the DigiNotar Certificate Authority breach. *FOX IT* (2012).

[100] Hyman, P. Cybercrime: It's serious, but exactly how serious? *Commun. ACM 56*, 3 (Mar. 2013), 18–20.

[101] IEBlog, M. Declaring Security. `http://blogs.msdn.com/b/ie/archive/2009/06/25/declaring-security.aspx`, 2009.

[102] Jackson, C., and Barth, A. Forcehttps: protecting high-security web sites from network attacks. In *Proceedings of the 17th international conference on World Wide Web* (2008), ACM, pp. 525–534.

[103] Jacobs, I., and Walsh, N. Architecture of the World Wide Web, Volume One. *W3C Recommendation* (2014).

[104] Kals, S., Kirda, E., Kruegel, C., and Jovanovic, N. Secubat: a web vulnerability scanner. In *Proceedings of the 15th international conference on World Wide Web* (2006), ACM, pp. 247–256.

[105] Klein, A. DOM Based Cross Site Scripting or XSS of the Third Kind. `http://www.webappsec.org/projects/articles/071105.shtml`, 2005.

[106] Kumar, D., Ma, Z., Durumeric, Z., Mirian, A., Mason, J., Halderman, J. A., and Bailey, M. Security challenges in an increasingly tangled web. In *Proceedings of the 26th International Conference on World Wide Web* (2017), International World Wide Web Conferences Steering Committee, pp. 677–684.

[107] Lamprakis, P., Dargenio, R., Gugelmann, D., Lenders, V., Happe, M., and Vanbever, L. Unsupervised Detection of APT C&C Channels using Web Request Graphs. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (2017), pp. 366–387.

[108] Laurenza, G., Aniello, L., Lazzeretti, R., and Baldoni, R. Malware triage based on static features and public apt reports.

International Conference on Cyber Security Cryptography and Machine Learning (2017), 288–305.

[109] Lekies, S., Stock, B., and Johns, M. 25 million flows later: Large-scale detection of dom-based xss. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (2013), ACM, pp. 1193–1204.

[110] Li, Y.-F., Das, P. K., and Dowe, D. L. Two decades of Web application testing - A survey of recent advances. *Information Systems 43*, 0 (2014), 20 – 54.

[111] Lie, H. W., and Bos, B. Cascading Style Sheets, level 1. *W3C Recommendation* (1996).

[112] Marlinspike, M. New tricks for defeating ssl in practice. *Blackhat* (2009).

[113] Martin, B., Brown, M., Paller, A., and Kirby, D. 2011 cwe/sans top 25 most dangerous software errors. `http://cwe.mitre.org/top25/`, 2011.

[114] Möller, B., Duong, T., and Kotowicz, K. This poodle bites: exploiting the ssl 3.0 fallback.

[115] Naylor, D., Finamore, A., Leontiadis, I., Grunenberger, Y., Mellia, M., Munafo, M., Papagiannaki, K., and Steenkiste, P. The Cost of the "S" in HTTPS. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies* (New York, NY, USA, 2014), CoNEXT '14, ACM, pp. 133–140.

[116] Nikiforakis, N., Invernizzi, L., Kapravelos, A., Van Acker, S., Joosen, W., Kruegel, C., Piessens, F., and Vigna, G. You are what you include: large-scale evaluation of remote javascript inclusions. In *Proceedings of the 2012 ACM conference on Computer and communications security* (New York, NY, USA, 2012), CCS '12, ACM, pp. 736–747.

[117] Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., and Vigna, G. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Security and privacy (SP), 2013 IEEE symposium on* (2013), IEEE, pp. 541–555.

[118] O'Reilly, T. What Is Web 2.0. `http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html`.

[119] OWASP. Transport Layer Protection Cheat Sheet. https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet.

[120] OWASP. The Ten Most Critical Web Application Security Vulnerabilities. *OWASP Top 10* (2007).

[121] OWASP. The Ten Most Critical Web Application Security Risks. *OWASP Top 10* (2017).

[122] Paoli, L., Visschers, J., Verstraete, C., and van Hellemont, E. The impact of cybercrime on belgian businesses. https://bcc-project.be/newsandpublications/Industry-survey-Final-report-2017, 2017.

[123] Quintero-Bonilla, S., del Rey, A. M., and Queiruga-Dios, A. New perspectives in the study of advanced persistent threats. *practical applications of agents and multi agent systems 619* (2017), 242–244.

[124] Rafique, M. Z., Chen, P., Huygens, C., and Joosen, W. Evolutionary algorithms for classification of malware families through different network behaviors. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation* (2014), ACM, pp. 1167–1174.

[125] Rescorla, E. HTTP Over TLS. *IETF RFC 2818* (2000).

[126] Richardson, R. 15th annual 2010/2011 computer crime and security survey. https://cours.etsmtl.ca/gti619/documents/divers/CSIsurvey2010.pdf, 2011.

[127] Ristić, I. Internet SSL Survey 2010. In *Black Hat USA 2010* (2010).

[128] Rizzo, J., and Duong, T. Crime: Compression ratio info-leak made easy. In *ekoparty Security Conference* (2012).

[129] Ross, D., and Gondrom, T. HTTP Header Field X-Frame-Options. *IETF RFC 7034* (2013).

[130] Rubio, J. E., Alcaraz, C., and Lopez, J. Preventing advanced persistent threats in complex control networks. In *European Symposium on Research in Computer Security* (2017), vol. 10493, pp. 402–418.

[131] Ryck, P. D., Desmet, L., Piessens, F., and Johns, M. *Primer on Client-Side Web Security*. Springer, 2014.

[132] Security, Q. Chinese websites security vulnerability situation analysis report (in Chinese), 2017.

[133] Security, W. Web Applications Security Statistics Report, 2016.

[134] Seifert, C., Delwadia, V., Komisarczuk, P., Stirling, D., and Welch, I. Measurement Study on Malicious Web Servers in the .nz Domain. In *Information Security and Privacy*, vol. 5594. Springer, 2009.

[135] Sen, A., and Hawthorn, G. *The Standard of Living - The tanner lectures*. Cambridge University Press, 1988.

[136] Son, S., and Shmatikov, V. The postman always rings twice: Attacking and defending postmessage in html5 websites. In *NDSS* (2013).

[137] Stamm, S., Sterne, B., and Markham, G. Reining in the web with content security policy. In *Proceedings of the 19th international conference on World wide web* (New York, NY, USA, 2010), WWW '10, ACM, pp. 921–930.

[138] Stock, B., Johns, M., Steffens, M., and Backes, M. How the web tangled itself: Uncovering the history of client-side web (in) security.

[139] Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., and Cranor, L. F. Crying Wolf: An Empirical Study of SSL Warning Effectiveness. In *Proceedings of the 18th Usenix Security Symposium* (2009), pp. 399–416.

[140] van Goethem, T., Chen, P., Nikiforakis, N., Desmet, L., and Joosen, W. *Large-Scale Security Analysis of the Web: Challenges and Findings*. Springer International Publishing, Cham, 2014, pp. 110–126.

[141] Ventures, C. 2017 Cybercrime Report, 2017.

[142] von Hirsch, A., and Jareborg, N. Gauging criminal harm: A living-standard analysis. *Oxford Journal of Legal Studies 11*, 1 (1991), 1–38.

[143] W3C. Document Object Model (DOM). `https://www.w3.org/DOM/`.

[144] Weichselbaum, L., Spagnuolo, M., Lekies, S., and Janc, A. Csp is dead, long live csp! on the insecurity of whitelists and the future of content security policy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), pp. 1376–1387.

[145] West, M. Content Security Policy: Cookie Controls. *W3C Working Group Note* (2016).

[146] West, M. Content Security Policy: Embedded Enforcement. *W3C Working Draft* (2016).

[147] West, M. Content Security Policy Level 3. *W3C Working Draft* (2016).

[148] West, M., Barth, A., and Veditz, D. Content Security Policy Level 2. *W3C Recommendation* (2016).

[149] West, M., and Goodwin, M. Same-site Cookies. *IETF Draft* (2016).

[150] Wikipedia. Resident Identity Card. http://en.wikipedia.org/wiki/Resident_Identity_Card_(PRC).

[151] Wilson, K. Revoking Trust in one CNNIC Intermediate Certificate. https://blog.mozilla.org/security/2015/03/23/revoking-trust-in-one-cnnic-intermediate-certificate/, 2015.

[152] Zalewski, M. Postcards from the post-xss world. http://lcamtuf.coredump.cx/postxss/, 2011.

[153] Zeller, W., and Felten, E. W. Cross-site request forgeries: Exploitation and prevention. *The New York Times* (2008), 1–13.

List of publications

Papers at international conferences and workshops, published in proceedings

- Ping Chen, Jonas Visschers, Cedric Verstraete, Letizia Paoli, Christophe Huygens, Lieven Desmet, Wouter Joosen. "The relationship between the cost of cybercrime and web security posture: A case study on Belgian companies". In *Companion Proceedings of the 11th European Conference on Software Architecture*, pages 115-120, Canterbury, UK, 2017.
- Ping Chen, Christophe Huygens, Lieven Desmet, Wouter Joosen. "Advanced or not? A comparative study of the use of anti-debugging and anti-VM techniques in generic and targeted malware". In *31st International Conference on ICT Systems Security and Privacy Protection (IFIP SEC 2016)*, Springer, pages 323-336, 2016.
- Ping Chen, Christophe Huygens, Lieven Desmet, Wouter Joosen, Longitudinal study of the use of client-side security mechanisms on the European web. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 457-462, Montreal, Canada, 2016.
- Ping Chen, Nick Nikiforakis, Lieven Desmet, Christophe Huygens. "Security analysis of the Chinese web: how well is it protected?" In *Proceedings of the 2014 Workshop on Cyber Security Analytics, Intelligence and Automation (SafeConfig 2014)*, pages 3-9, ACM, 2014.
- Ping Chen, Lieven Desmet, Christophe Huygens. "A study on Advanced Persistent Threats". In *Proceedings of the 15th IFIP TC6/TC11 Conference on Communications and Multimedia Security*, pages 63-70, Aveiro, Portugal, 2014

- Ping Chen, Nick Nikiforakis, Christophe Huygens, Lieven Desmet. "A dangerous mix: Large-scale analysis of mixed-content websites." In *Proceedings of the 16th International Conference on Information Security (ISC 2013)*, pages 354-363, Dallas, USA, 2013.

- M. Zubair Rafique, Ping Chen, Christophe Huygens, Wouter Joosen. "Evolutionary algorithms for classification of malware families through different network behaviors." In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, pages 1167-1174, Vancouver, Canada, 2014

- Tom Van Goethem, Ping Chen, Nick Nikiforakis, Lieven Desmet, Wouter Joosen. "Large-scale security analysis of the web: Challenges and findings." In *Trust and Trustworthy Computing*, volume 7, pages 110-125, Heraklion, Greece, 2014.

- Sebastiaan de Hoogh, Berry Schoenmakers, Ping Chen, Harm op den Akker, "Practical secure decision tree learning in a teletreatment application", In *18th International Conference on Financial Cryptography and Data Security*, pages 179-194, Springer, 2014.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
IMEC-DISTRINET
Celestijnenlaan 200A box 2402
B-3001 Leuven
https://distrinet.cs.kuleuven.be