Clubbing Seals: Exploring the Ecosystem of Third-party Security Seals

Tom Van Goethem[‡], Frank Piessens[‡], Wouter Joosen[‡], Nick Nikiforakis[†]
[‡]iMinds-Distrinet, KU Leuven, 3001 Leuven, Belgium
firstname.lastname@cs.kuleuven.be

[†]Department of Computer Science, Stony Brook University
nick@cs.stonybrook.edu

ABSTRACT

In the current web of distrust, malware, and server compromises, convincing an online consumer that a website is secure, can make the difference between a visitor and a buyer. Third-party security seals position themselves as a solution to this problem, where a trusted external company vouches for the security of a website, and communicates it to visitors through a security seal which the certified website can embed in its pages.

In this paper, we explore the ecosystem of third-party security seals focusing on their security claims, in an attempt to quantify the difference between the advertised guarantees of security seals, and reality. Through a series of automated and manual experiments, we discover a real lack of thoroughness from the side of the seal providers, which results in obviously insecure websites being certified as secure. Next to the incomplete protection, we demonstrate how malware can trivially evade detection by seal providers and detail a series of attacks that are actually facilitated by seal providers. Among other things, we show how seals can give more credence to phishing attacks, and how the current architecture of third-party security seals can be used as a completely passive vulnerability oracle, allowing attackers to focus their energy on websites with known vulnerabilities.

Categories and Subject Descriptors

K.6.5 [**Security and Protection**]: Unauthorized access; H.3.5 [**Online Information Services**]: Web-based services; K.4.4 [**Electronic Commerce**]: Security

Keywords

Web applications; security seals; web-based attacks

1. INTRODUCTION

Nowadays, it has become rather uncommon for an entire month to go by without some news of a major security incident. Whether by bugs in cryptographic libraries [11, 23], malware installed on points-of-sale terminals [21], the exploitation of web application vulnerabilities [22, 27, 29], or social engineering [1, 5], databases with credentials and personal details of millions of users seem to be finding their way into the hands of attackers, on a regular basis.

The monetary losses due to these events and due to cybercrime in general, are sizable. According to a recent report on the economic cost of cybercrime and cyber espionage, the cost of these activities reach 100 billion dollars each year, just for the United States alone [25]. Note that these losses are not just direct monetary losses, but also indirect ones. One interesting indirect loss is the opportunity cost to businesses due to reduced trust in online services.

Trust has always been a barrier to the adoption and use of new technology. In the context of the web, people are called to trust the websites of online companies, many of which do not have an offline brick-and-mortar presence, with their sensitive personal and financial information. In a 2008 survey, 75% of online shoppers did not like the fact that they had to provide their credit card and personal information online [13], while in a 2013 survey, only 61% of U.S. Internet users were banking online [9].

In such an environment of distrust, companies can distinguish themselves from others by not only securing their infrastructure but also convincing the user that they did so. This is even more so for smaller companies which do not enjoy the implicit trust afforded by the wide recognition of logos and household company names.

One way of achieving this goal is through the use of third-party security seals. A third-party security seal is an image that a website can embed in its HTML code which signals to consumers that the website has been scanned by a security company and has been found to be without issues, i.e., without vulnerabilities and without malware. Seals are typically provided by large security companies, like McAfee and Symantec, and are meant to be recognizable by the user and lend the credibility and trust associated with these large companies, to the certified site. Depending on the seal provider, security seals can cost anywhere between hundreds to thousands of dollars per year which can be a significant reoccurring security investment, especially for smaller companies.

Prior research on the topic of third-party security seals has mostly focused on whether a user recognizes their presence on certified websites and whether they result in higher confidence and thus increased sales [4, 10, 15, 19].

CCS'14, November 3–7, 2014, Scottsdale, Arizona, USA.

http://dx.doi.org/10.1145/2660267.2660279.

In this paper, we perform a three-pronged analysis of third-party security seal providers. Instead of measuring whether users trust security seals, we want to know whether these seals *should* be trusted in the first place. First, we compile a list of ten popular seal providers and analyze their certification methods. We discover that, among others, the security checking of seal-using websites is done almost entirely in a black-box fashion where the seal providers try a list of automated attacks on their clients. By designing and deploying a known vulnerable web application, we witness the inaccuracy and haphazardness of these scanners, with the best one detecting less than half of the known vulnerabilities.

Second, we turn our attention to existing businesses that use third-party security seals, discovering more than 8,000 websites certified to be secure by the studied seal providers. By gathering and comparing features that are telling of a website's security hygiene, e.g., the use of HttpOnly cookies and Anti-CSRF tokens in HTML forms, we show that seal-using websites are *not* more prudent than other websites of an equivalent nature and popularity. Moreover, by obtaining explicit permission for a manual penetration test on nine seal-using websites, we demonstrate how a moderately motivated attacker can discover high-risk vulnerabilities in most certified websites, in less than one working day.

Finally, taking into account the workings of seal providers, we detail a series of attacks on websites that are actually facilitated by the use of third-party security seals. Among others, we describe the architecture of a completely passive vulnerability oracle that allows an attacker to discover easily exploitable websites by monitoring the appearance and disappearance of third-party security seals on seal-using websites.

Our main contributions are:

- We perform the first study of third-party security seals that tests the claimed security guarantees, exposing a lack of thoroughness and sophistication
- We show that seal-certified websites do not have different security practices when compared to other equivalent non-seal-using websites
- We demonstrate that even moderately motivated attackers have no problem finding critical vulnerabilities in websites that are certified to be secure
- We describe how attackers can abuse third-party security seals proposing, among others, a completely passive vulnerability oracle that takes advantage of the way seal providers react when one of their clients has been detected as vulnerable

2. SECURITY SEALS

In this section, we describe the general workings of third-party security seals and list the ten seal providers that we analyzed, together with their features and deviations from the generic model of a third-party security seal.

2.1 General Architecture

In general, third-party security seals follow the architecture shown in Figure 1 which is comprised of two distinct components. We arrived at this architecture through the analysis of all ten investigated providers, and the recording of common features.

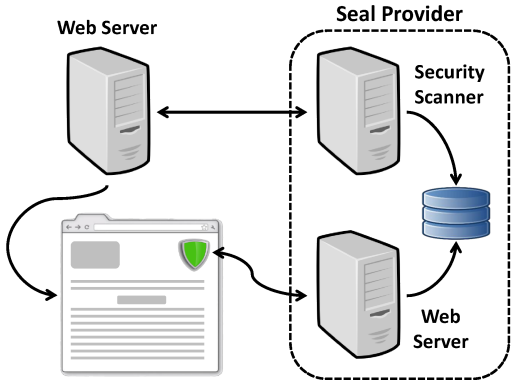

Figure 1: High-level view of the architecture and delivery of third-party security seals

First, if seal providers are to give meaningful security certifications to other websites, they must have a way to assess the security stance of those websites. This is done through the use of an automated scanner which periodically checks the websites of their clients for the presence of security issues, usually involving vulnerabilities and, in some cases, malware. While the exact scanning frequency depends on each seal provider and the version of the seal a customer has bought, the scans are usually repeated either on a daily or a weekly basis. The clients of seal providers are given access to a web-based "control panel" where they can inspect the findings of the latest security scan. If the scan discovered vulnerabilities, the client is given a description of the vulnerability, its exact location and nature (e.g. reflected XSS found on parameter `page` of `index.php`), and some generic guidance for its remediation, usually in the form of links towards whitepapers and other security resources. When a vulnerability is discovered, the owners of certified websites are typically given a "grace period" of a few days which they can use to correct the discovered vulnerability before it affects their certification, i.e., the visual component displayed on their website.

The second component of a seal-provider is the seal itself. When a website owner subscribes her website for a third-party security seal, she is given a snippet of HTML code (or JavaScript code which creates an HTML snippet) which she is instructed to place in her page, at the position where she wants the third-party seal to appear. The general format of seal snippets is as follows:

```
<a href="https://seal-provider.com/info?
        client=example.com">
 <img src="https://seal-provider.com/shield?
        client=example.com"/>
</a>
```

This snippet always involves an HTML image tag which dynamically requests an image from the web server of the seal provider. If a website is found to be secure, the server will respond with an image of a seal, as shown in Figure 1, which typically includes the logo of the seal provider and is meant to be recognizable by the visitors of the seal-using website and lend it the credibility associated with the seal-provider. Usually, the image is wrapped in an anchor tag

#Clients	Name	Yearly Cost	Vulnerability Scan	Malware Scan	Server-side access	Server AuthN	Disappearing seal	Grace Period (Days)
3,980	Norton Secured	$995	✓	✓	–	–	–	NA
3,029	McAfee SECURE	$300	✓	✓	–	–	✓	Unknown
463	Trust-Guard	$697	✓	–	–	–	✓	7
459	SecurityMetrics	$120	✓	✓	–	–	–	0
281	WebsiteProtection (GoDaddy)	$84	✓	✓	✓	–	✓	0
118	BeyondSecurity	$360	✓	–	–	✓	✓	0
109	ScanVerify	$100	✓	–	–	–	✓	2
68	Qualys	$495	✓	✓	–	–	✓	3
48	HackerProof	$2,295	✓	–	–	–	✓	5
4	TinfoilSecurity	$2,388	✓	–	–	✓	–	NA

Table 1: Overview of evaluated seal providers and their features

which is clickable, leads to the domain of the seal provider, and shows the visitor of a certified website more information about the seal and the seal provider, e.g., the coverage of the scan that produced the seal, and the day of the last scan. Apart from providing more information, this linked page creates an obstacle for website owners who wish to create the illusion of being certified, e.g., by showing a fake security seal on their website, possibly scraped by the website of a seal provider's paying client. Since the extra information is provided in a page under the seal-provider's domain, a non-paying user cannot mimic that part of the certification.

Interestingly, for the majority of seal providers, when a vulnerability is discovered and the certified website fails to correct it within its allotted grace period, the security seal merely becomes invisible. That is, even when a website is vulnerable, a visitor of that website will never see a "negative" security seal. The only way that a visitor can discover this fact is by the examination of the HTML source code of a page and the discovery of the aforementioned HTML snippet – a task well out of reach of common users of the web. As such, the vast majority of web users are not able to distinguish between a website that does not use a third-party security seal, and one that does but is vulnerable.

2.2 Seal Providers

To discover third-party seal providers, we searched in a popular search engine for phrases such as "site security seal" and "site safety seal". For each result, we manually examined the website of each seal vendor to ensure that the seal coverage included the detection of vulnerabilities, as opposed to other types of seals that verify a site's identity and the proper use of SSL. Due to the extensive labor involved with the evaluation of each seal provider and its clients, we limited ourselves to ten providers of security seals.

Table 1 lists the ten investigated third-party security seal providers ordered by the number of their clients that we could discover (we describe the process of client discovery in Section 3). One can see that the services vary greatly in terms of popularity as well as in terms of cost. Interestingly, there seems to be no correlation between the popularity of a seal provider and the price of seal. Since the two, by far, most popular seal providers are also two large, recognizable antivirus companies, it is likely that the popularity of seal providers is related more to brand recognition and less to other factors, such as price and word-of-mouth.

Five out of the ten seal providers support malware scanning in addition to vulnerability scanning, yet only one provides the option of scanning the server for malware over the FTP protocol. As such, the other services can only discover malware on the indexable pages and directories of a website. Only two out of the ten investigated services have the option of server authentication, i.e., scanning the part of a website that is behind a registration wall. In these two cases, website owners can give the credentials of a user to the seal provider and the location of the login page. This means that the vulnerability scanners of the majority of seal providers will not be able to find vulnerabilities and malware that reside on the authenticated part of a website.

The investigated seal providers exhibit varied behavior when it comes to how they react in the presence of a discovered vulnerability. The majority of seal providers returns an invisible image when the client fails to mitigate the vulnerability within the grace period. However, there are two types of deviation from this behavior: the Norton Secured seal will always be shown, even when vulnerabilities were found, and similarly, for SecurityMetrics and TinfoilSecurity, the seal will also remain visible, but the status page on the seal provider website will no longer show that there is a passing certification.

Finally, we would like to stress that although security seals have been around for more than a decade, it is still a market that is actively being developed. During the course of our research, McAfee and Godaddy rebranded their security seal products. The McAfee SECURE seal, which previously offered both a malware scan and vulnerability analysis as a single package, was split and now only offers a security seal for passing the malware scan. In our research, we evaluated the combination of the malware scan and vulnerability scan, as was originally offered. WebsiteProtection, a GoDaddy product, was rebranded as SiteLock, which most likely reflects a change in the third-party that GoDaddy relies on for their security seal product. As in the case of the McAfee security seal, our research is mainly based on the original product that was offered by GoDaddy.

3. ADOPTION

As explained in Section 2.1, when website owners subscribe their websites to seal providers, they are given a small HTML snippet that they have to include in their websites. This snippet is responsible for fetching and displaying the security seal to the visitors of the certified site. In this section,

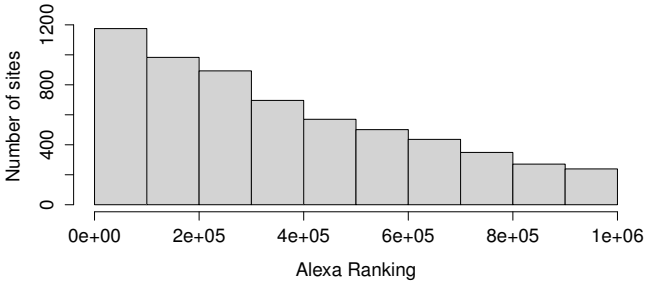

Figure 2: Distribution of seal-using websites in the Alexa top 1 million websites

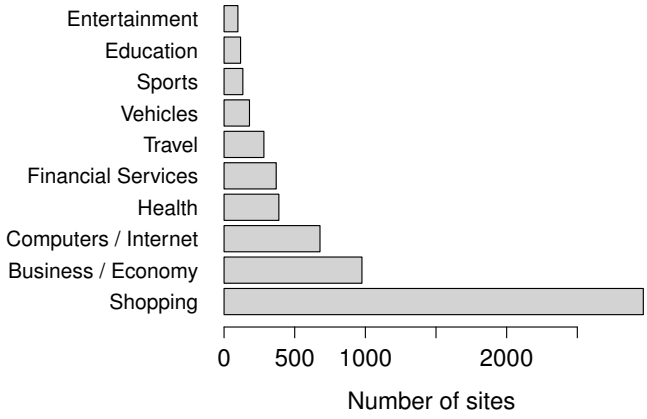

Figure 3: Ten most popular categories of seal-using websites

we take advantage of this snippet in order to automatically detect seal-using websites, in an effort to understand the nature of websites that choose to certify themselves using security seals.

More specifically, using a crawler based on PhantomJS, we performed a shallow crawl of the top 1 million websites according to Alexa, searching for inclusions of specific remote images and anchor tags from each of the ten studied seal providers. To account for seal-using websites that are not part of the top 1 million Alexa websites, we used some advanced search features of Google's search engine, which we based on the same HTML snippets. For example, the following search query `site:scanverify.com/siteverify.php` returns a list of websites using seals by ScanVerify. Using these processes we were able to discover a total of 8,302 seal-using websites.

From the 8,302 websites, 73.64% was part of Alexa's top 1 million websites ranking. Figure 2 shows the distribution of these websites across the ranks of Alexa. The distribution is right-skewed where the usage of third-party security seals decreases together with the ranking. Our findings indicate that websites that are already popular, still choose to use seals as a mechanism of convincing users that they do take security seriously.

To identify the nature of these 8,302 websites, we used TrendMicro's public website categorization database [32]. Figure 3 shows the ten most popular categories of seal-using websites. As one can see, the "Shopping" category is by far the most popular with 35.74% of the entire dataset being categorized as such. Given the motivation for using third-party security seals that the seal providers themselves use, this result makes intuitive sense. Security seals are advertised as capable of increasing a user's trust for any given website which, in turn, translates to increased sales. As a matter of fact, most seal providers' advertising campaigns heavily rely on testimonials where existing clients claim to have seen a significant increase of sales (in the range of 5%-20%) after the adoption of their security seal. As such, shopping websites are the primary target audience for buying security seals from seal providers. This result is also interesting from a security point of view. Websites belonging to e-shops are highly dynamic in nature, with frontends and backends, and various e-commerce modules. As such, their extended attack surface, together with the prospect of exfiltrating financial and personal data upon compromise, make shopping websites more attractive targets over other categories of websites, such as blogs and news websites.

4. SECURITY EVALUATION

Seal vendors claim that a security seal increases the trustworthiness of a certified website and leads to increased sales. In this section, we seek to understand whether a user's trust *should* increase in the presence of a third-party security seal.

In order to assess the thoroughness of service provided by the ten investigated seal providers, we conducted the following three experiments. First we compare the security practices of seal-using websites to other equivalent websites which do not make use of seals. Second, by obtaining permission for penetration testing, we investigate whether a moderately interested attacker would be able to find a vulnerability in a supposedly secure website, i.e., a website bearing a security seal. Third, we set up a webshop including multiple vulnerabilities, in order to understand which vulnerabilities are discoverable by seal providers and how easy it is for a vulnerable site to obtain a clean bill of health.

4.1 Comparison to non-certified websites

When a website uses a third-party security seal, it may seem reasonable to assume that the administrators of that website are, in general, interested in the security of their site and are thus taking all the necessary precautions to secure their services and protect their users.

In this section, we test this assumption by comparing the adoption of popular security mechanisms by seal-using websites against the adoption of the same mechanisms by equivalent websites which do not use third-party seals. Next to security mechanisms we also test for issues that can be detected in a non-intrusive way.

Comparison Dataset

To have meaningful comparisons between seal-certified and non-certified websites, the second set of websites must be similar to the first one, in all matters except for the adoption of a security seal. While this is virtually impossible to establish from the client-side without full knowledge of an application's codebase and environment, we provide an approximate solution as follows.

For every seal-utilizing website, we attempt to identify another site of the same category within ten places in the Alexa ranking of the top 1 million websites. If, for instance, `buyfromhome.com` is a seal-utilizing e-shop ranked at the

100th place, we search for another e-shop site either ten ranks above, or ten ranks below the 100th place. The direction of our search is decided probabilistically using the probability distribution of a fair coin. As before, the categories of each site are determined using TrendMicro's public website categorization engine [32].

Using this process we were able to match 2,238 seal-using websites to 2,238 other websites of equivalent rank and category that do not use third-party security seals.

Security Indicators

In recent years, as a response to a continuous battery of exploitations of web application vulnerabilities, browser vendors and the research community introduced a series of client-side security mechanisms that are today available in the vast majority of modern browsers. These client-side mechanisms are usually guided by server-side policies, where the web server expresses its security desires through HTTP headers and the browsers are responsible for enforcing them at the side of the user. In addition, web application programmers have come up with various "best practices" that should always be followed, e.g., an anti-CSRF token in all forms.

While the presence or absence of such security mechanisms does not equate to proof of the presence or absence of exploitable vulnerabilities, they still can be used as indicators of the *security hygiene* of any given website. In addition, these mechanisms can be detected in a completely passive fashion thereby not incurring any unnecessary stress on web applications. In prior research, Nikiforakis et al. [26] combined some of these indicators in a Quality-of-Maintenance metric and applied it on servers offering remote JavaScript libraries. Vasek and Moore investigated whether certain server characteristics can be used as risk factors for predicting server compromise [33], and found that HttpOnly cookies can, for some types of compromise, be negative risk factors, i.e., their presence is correlated more with non-compromised websites rather than compromised ones.

For our experiment, we searched for the presence or absence of the following, passively discoverable, security mechanisms and best practices:

- HTTP Strict Transport Security (HSTS)
- Secure Cookies
- HttpOnly Cookies
- Content Security Policy (CSP)
- X-Frame-Options (XFO)
- iframe sandboxing
- Anti-CSRF Tokens
- X-Content-Type-Options
- SSL-stripping Vulnerable Form

A brief description of each of these mechanisms can be found in this paper's Appendix.

Results

For each of the 4,476 websites, we used a crawler based on PhantomJS to automatically visit a site's main page and ten other pages within the same domain, which were obtained using a shallow crawl. To simplify comparisons, we counted the present security mechanisms and issues in a binary fashion. If, for example, one page of the domain `example.com` used an HttpOnly cookie, then we credited the entire `example.com` domain with that security mechanism.

Security Mechanism or Issue	Sites w/ Seals (%)	Sites w/o Seals (%)	Significantly different? (p-value)
HSTS	1.05	1.06	✗ (1.00)
Secure Cookies	1.83	0.42	✗ (0.06)
SSL Stripping	15.45	15.64	✗ (0.99)
X-Frame-Options	3.71	5.14	✓ (0.02)
HTTP-Only Cookies	42.27	29.98	✓ (<0.01)
CSP	0.00	0.00	– (NA)
Anti-CSRF Tokens	6.39	11.89	✓ (<0.01)
X-Content-Type	0.00	0.00	– (NA)
iframe sandbox	0.18	0.04	✗ (0.37)

Table 2: Comparison of the discovered issues and security mechanisms between websites with and without seals. Highlighted entries denote the better value, in the statistically significant cases.

Table 2 shows the percentage of security mechanisms and issues discovered in the seal-utilizing websites and in our gathered set of equivalent websites. To compare the adoption between our two sets of websites, we conducted a two-sided hypothesis test for the comparison of two independent proportions. For each row in Table 2, our null hypothesis (H_0) was that the true proportion of that mechanism is the same between the two sets, while the alternative hypothesis (H_A) was that the true proportions of that mechanism for the two sets are *different* from each other. The last column of Table 2 shows the results of each hypothesis test, i.e., whether the adoption of each mechanism is different in a *statistically significant* way, and reports the p-value of the hypothesis test. Following standard practices in hypothesis testing, 0.05 was the cut-off point for the computed p-value, over which the null hypothesis is maintained.

As one can see, only one third of the measured proportions were different in a statistically significant way, while in two out of the three cases, the difference was credited in favor of the websites *without* security seals. The lack of more significant differences between the adoption of security mechanisms can be interpreted as a lack of systematic difference between the security hygiene of seal-using websites and the hygiene of equivalent websites that do not use seals. This, in turn, hints towards the absence of a holistic security strategy by the adopters of third-party security seals. In other words, if the intuitive notion that bad security hygiene is correlated with increased probability of compromise is true, seal-using websites are *not* more secure than their non-seal equivalents.

4.2 Penetration Testing

Security seal providers claim to protect websites by periodically scanning for the presence of thousands of vulnerabilities. One could assume that this would result in the detection of most easily discoverable vulnerabilities, thus making the discovery of a vulnerability by an attacker, a long and arduous task.

To test this hypothesis, we contacted 1,000 seal-using websites asking for explicit permission to conduct a manual penetration test. While the vast majority of websites never responded to our request, nine websites granted us permission to proceed. In order to avoid bias in our experiment we simulated a moderately motivated attacker who only has eight hours (one working day) to find a vulnerability before proceeding to the next target.

In our manual penetration test, we evaluated websites for several common vulnerabilities, such as SQL injection, cross-site scripting and CSRF, as well as for application-logic attacks. Despite the limited time available for evaluating each website, we were able to find several severe vulnerabilities, such as XSS and SQL Injection, for seven out of the nine evaluated websites. Additionally, we found HTTP parameter tampering vulnerabilities, as well as application-flow vulnerabilities, in three out of the four webshops that we analyzed, allowing us to order products and services for arbitrary prices.

These results clearly indicate that the hypothesis that attackers with limited resources are unable to find vulnerabilities in sealed websites, is false. In our manual analysis, we could register on the websites under evaluation, access content that requires authentication, and reason about prices in shopping carts – actions that are not supported by the automated scanners of most seal providers. Besides these general limitations that render seal providers unable to find vulnerabilities which require a series of coherent actions, we also found easily discoverable vulnerabilities, which were missed by the seal providers, in six out of nine websites. These vulnerabilities consist of cross-site scripting vulnerabilities where a GET or POST parameter was reflected without proper encoding, and even a "textbook" SQL injection.

We also want to mention one incident that demonstrates the haphazardness of the certification of certain seal providers. In one specific case, an e-shop, certified to be secure by a third-party seal provider, gave us an SQL error while contacting them to ask for permission for a manual penetration test. When reviewing the case, we realized that we had used a contraction in our message to the website where we inadvertently introduced a single quote, e.g., the phrase "do not" contracted to "don't". The SQL error was generated by that single quote in the message body.

4.3 Vulnerable Webshop Experiment

Being able to accurately assess the state of security of a website, is one of the key requirements of a seal provider, if the seal is to be trusted by consumers. That is, if trivially vulnerable websites can be certified as secure, then the certification becomes void of meaning. To evaluate the accuracy of the tools used by seal providers to verify a website's security, we set up a webshop which included a number of severe vulnerabilities. More specifically, we used an outdated version of PrestaShop, a popular open source e-commerce application, which suffers from a cross-site scripting vulnerability, and expanded the attack surface by including several other vulnerabilities spanning many typical web application vulnerability classes.

The vulnerable webshop reflects a realistic website containing vulnerabilities we encountered in real-world websites during our penetration testing. Next to a number of well known vulnerabilities, such as SQL injection and XSS, we also included vulnerabilities that are less popular, such as a remote JavaScript inclusion from a stale domain and a CSRF issue with OAuth login. In total, the following twelve vulnerabilities (V_1 to V_{12}) were present in our vulnerable web application:

- **SQL Injection (V_1):** A textbook example of SQL injection. One of the GET parameters was not properly sanitized and a user-controlled SQL statement could be executed.
- **SQL Injection - Ajax (V_2):** Similar to V_1, but only Ajax requests were made to this endpoint. Consequently, this endpoint could only be discovered by executing JavaScript code.
- **Sensitive files (V_3):** We uploaded a `phpinfo.php` file, which discloses sensitive information of a PHP installation, and a `.git` folder, which can leak the contents of several sensitive files.
- **Stale remote JavaScript inclusion (V_4):** On each page, a JavaScript file was included from an unregistered domain. An attacker could register that domain, and serve malicious JavaScript from it, which would be executed on all pages of the vulnerable webshop.
- **OAuth - CSRF parameter (V_5):** A "Login with Facebook" link was added, which points to Facebook's OAuth endpoint. This link did not contain a `state` parameter, which would allow an attacker to perform a CSRF attack and make a victim log in as the attacker.
- **Malware (V_6):** Every page of the webshop contained a link to a malicious executable. This link was made invisible using CSS in order to prevent infecting casual visitors of our test website.
- **Directory listing (V_7):** One of the directories that stored images, allowed directory listing. This directory also contained malicious executables.
- **Reflected XSS (V_8):** A GET parameter on one page of our web application was reflected without any encoding.
- **Reflected XSS - form action (V_9):** Similar to V_8, but the endpoint was located in the `action` attribute of a `form` element, as opposed to the `href` attribute of a link as in V_8.
- **Reflected XSS - additional parameter (V_{10}):** For this vulnerability, the query parameters were reflected without encoding. This only happened when the `controller` parameter was set to a certain value, so a GET parameter needed to be added for the discovery of this vulnerability.
- **Reflected XSS - JavaScript context (V_{11}):** In this case, again a GET parameter was reflected in a page, but the "<" and ">" characters were properly encoded. However, since the parameter was reflected in a JavaScript string context, an attacker could terminate the string with a quote, which was not escaped, and inject arbitrary JavaScript code.

- **DOM-based XSS (V_{12}):** On several pages, the fragment of the URL (the portion after the # sign in the URL), was written to the document, without any encoding.

From the ten seal providers listed in Table 1, we were able to purchase seals (or get free trials) from eight of them. The two missing providers implemented strict checks for the existence of a valid business which we did not attempt to bypass.

Table 3 shows the vulnerabilities that were discovered by each seal provider. The names of the seal providers have been anonymized, as our study is not meant to promote one product over another, but rather to show the coverage, or lack thereof, by all companies. The first thing that one can notice is that *all* seal providers found less than half of the vulnerabilities. Even more worrisome is that two seal providers did not manage to find any vulnerabilities. By analyzing the requests made to our web server, we found that these seal providers merely ran Nmap and Nessus scans, which listed open ports and checked for the presence of certain files. These scanners are not meant to discover vulnerabilities in web applications, thus they are far from sufficient to evaluate the security of a website.

V_8 and V_9, the two "standard" reflected XSS vulnerabilities, were found by the majority of seal providers. Even though V_1 was a textbook example of SQL injection, only half of the seal providers managed to find this vulnerability. Moreover, only Seal Provider 6 and 7 were able to find the SQL injection vulnerability in the Ajax endpoint (V_1). Interestingly, these were also the only two seal providers whose scanners executed the JavaScript code of our web application. Since more and more websites rely on JavaScript for delivering functionality to end users, a lack of support for JavaScript will certainly limit the number of vulnerabilities that a vulnerability scanner is able to find.

From the four seal providers that claim to check for the presence of malware, only two managed to find the malware present on our vulnerable web application. Note that we purposefully uploaded malware that was detected as such by the vast majority of antivirus engines on VirusTotal, to ensure that the malicious executables would be flagged by any proper antivirus product. Interestingly, one of these seal providers only managed to find the malware binary *after* they were given FTP access to our server, an optional feature they provide. The inability to find publicly-reachable malware by browsing our webshop, is another indication that the security-scan employed by seal providers is incomplete.

Overall, it is clear that the coverage of security seals leaves much to be desired. Even if a website would employ multiple security seals, certain classes of vulnerabilities would still remain fully undetected. While one can argue that some of the vulnerabilities present in our webshop are of a more exotic nature, such as V_4 and V_5, the fact is that these vulnerabilities are known by the security community, and can be easily discovered by an automated scanner as long as the detection logic is present. The absence of support for them, indicates that automated scanners may have trouble detecting newer vulnerabilities, even if those are easier to detect than traditional ones.

Finally, we compared the coverage of the seal providers with that of three popular web application vulnerability scanners. Web application vulnerability scanners typically work in a fully automated way and, despite their shortcomings [7], are a popular option for discovering vulnerabilities in websites. As Table 4 shows, the coverage of the scanners is higher than all but one of the tools employed by the security seal providers, which again shows that the scans required to obtain a security seal are far from rigorous.

Seal provider	Vulnerabilities						
	V_1	V_2	V_3	V_4	V_5	V_6	V_7
Seal Provider 1	–	–	–	–	–	N/A	–
Seal Provider 2	–	–	–	–	–	N/A	–
Seal Provider 3	–	–	✓	–	–	✓	✓
Seal Provider 4	–	–	–	–	–	✓	–
Seal Provider 5	✓	–	–	–	–	N/A	–
Seal Provider 6	✓	✓	–	–	–	–	–
Seal Provider 7	✓	✓	–	–	–	–	–
Seal Provider 8	✓	–	✓	–	–	N/A	–

	V_8	V_9	V_{10}	V_{11}	V_{12}	Coverage
Seal Provider 1	–	–	–	–	–	0/11 (00.0%)
Seal Provider 2	–	–	–	–	–	0/11 (00.0%)
Seal Provider 3	–	–	–	–	–	3/12 (25.0%)
Seal Provider 4	✓	✓	–	–	–	3/12 (25.0%)
Seal Provider 5	✓	✓	–	–	–	3/11 (27.3%)
Seal Provider 6	✓	✓	–	–	–	4/12 (33.3%)
Seal Provider 7	✓	✓	–	–	–	4/12 (33.3%)
Seal Provider 8	✓	✓	✓	–	–	5/11 (45.5%)

Table 3: Vulnerability detection results. For each seal provider, the vulnerabilities that were discovered are indicated with a checkmark.

Vulnerability Scanner	Vulnerabilities						
	V_1	V_2	V_3	V_4	V_5	V_6	V_7
Acunetix	–	✓	–	–	–	N/A	✓
HP WebInspect	–	✓	–	–	–	N/A	✓
Burp Suite	✓	✓	–	–	–	N/A	✓

	V_8	V_9	V_{10}	V_{11}	V_{12}	Coverage
Acunetix	✓	✓	✓	–	–	5/11 (45.5%)
HP WebInspect	✓	✓	✓	–	–	5/11 (45.5%)
Burp Suite	✓	✓	✓	–	–	6/11 (54.5%)

Table 4: Vulnerability detection results for Web Application Vulnerability Scanners. For each vulnerability scanner, the vulnerabilities that were discovered are indicated with a checkmark.

5. ATTACKS

In the previous section, we showed that seal providers perform very poorly when it comes to the detection of vulnerabilities on the websites that they certify. One, however, could still argue that such products operate in a "best effort" manner and that, despite our findings, they still provide some tangible security benefits. In this section, we show that, paradoxically, third-party security seals can assist attackers in identifying vulnerable targets, and even provide them with the exact vulnerability. In addition to attacks against seal-using websites, we also show how an attacker can, in some cases, use seal providers to attack non-seal websites and how an interested party (attacker, or sketchy webmaster of a vulnerable website) can trivially evade detection by a seal provider.

5.1 Security Seal as an Oracle

Being able to accurately determine whether a website contains a vulnerability is incredibly useful for attackers since it allows them to focus their attention on websites that are likely to provide some yield.

The way security seals are currently displayed on websites, enables an adversary to pick the easiest prey from a herd of seal-utilizing websites. More precisely, security seals, which are generally hosted on the servers of the seal providers, should only be visible when a website is found to be secure. This means that if a vulnerability is found on a website, and the webmaster fails to mitigate it within the allotted grace period, the seal will stop showing. During our vulnerable webshop experiment, as discussed in Section 4.3, we discovered that when a seal provider wanted the seal to stop showing, they would either make the image transparent or provide an image with 1x1 dimensions.

Because of the difference in image size or content, it is possible to determine the security status of seal-using websites in an automated way. Thus, an attacker could set up a crawler to daily visit seal-using websites and be alerted whenever the image of a seal changes. To evaluate the feasibility of this attack scenario, we conducted the following experiment: for a two-month period, we visited the set of 8,302 seal-utilizing websites on a daily basis and extracted the security seal that was shown on each website. In addition, we also stored the webpage of the security seal provider that results when a user clicks on the image of each seal.

Table 5 shows the results of this two-month-long experiment. From all the seal-using websites, we discovered that 333 websites (Column 2 of Table 5) were given an invisible security seal for at least one day of our experiment. That is, in 333 cases, a website's seal transitioned from a showing seal to an invisible one, or vice versa. This indicates that either a website went from secure to vulnerable for at least one day, or was vulnerable for a series of days and went back to secure when the webmaster mitigated the discovered vulnerabilities. Related to this, for 189 websites in our dataset (Column 3 of Table 5), the seals were constantly invisible for the entire monitored period.[1]

This could either be due to an expired contract between the seal-using website and the seal provider, or due to a website being constantly vulnerable. In any case, from an attacker's point of view, an invisible seal should provide more than enough motive to start attacking a website.

Apart from the side-channel of seals appearing and disappearing, we discovered that for three seal providers, the combination of seal appearance and status page was different when a website was no longer a client of the seal provider, to when a website was vulnerable. For instance, for one seal provider, the seal would remain intact but the status page was indicating that the date of the last successful scan was prior to the current date. As such, for these three cases, shown in the last column of Table 5, we can relatively safely conclude that the seal-using website was vulnerable during our monitored period.

While a pointer towards a vulnerable website is already of great help to attackers, a disappearing seal does not pinpoint the exact vulnerability necessary to exploit a website. In the context of security seals, however, attackers can, in some cases, elicit the exact vulnerability out of a seal provider.

[1]Note that the sum of these seals in Table 5 is 190, since one website was including seals from two different providers.

Seal Provider	# Sites w/ changing seals	# Sites w/ blank seals	# Sites most likely vulnerable
McAfee SECURE	260	64	-
HackerProof	3	11	-
WebsiteProtection	19	52	46
Qualys	27	15	20
Trust-Guard	5	23	-
BeyondSecurity	19	24	38
ScanVerify	0	1	-

Table 5: The number of websites whose security seals never appeared, or disappeared and reappeared during our two-month-long experiment.

In our evaluation of the thoroughness of the security checks done by seal providers, we noticed that for each scan a very similar set of requests were made. These requests checked, among others, for the presence of certain files, whether a parameter was reflected without encoding, or whether a certain SQL statement would be executed. By setting up a website and purchasing a security seal (or getting a free trial), the attacker can collect the series of requests and replay them to the vulnerable website, thus discovering the exact vulnerability that caused the victim's seal to disappear. In the cases where the scan of a seal provider is dependent on the web application discovered, the attacker could set up the same web application as his victim and thus collect relevant, probing requests. An attacker could even try to replay the requests that he receives on his server directly on the victim website and extract discovered vulnerabilities in a MitM fashion. Note that once an attacker collects a trace of attack requests, he can reuse them an "infinite" number of times against vulnerable websites of that seal provider.

It could be argued that if attackers had in their possession a tool that could check the security of websites, they could run that tool against an arbitrarily large number of websites. However, we experienced that the security scanners often made a substantial number of requests, in one case up to 180,000, to the probed web server. Running such a scan on a large number of websites would require access to a considerable amount of resources, something the average wrongdoer may not have. As such, it would not come as a surprise if an attacker would prioritize attacking a website known to contain a specific exploitable vulnerability. Lastly, it is worth reminding the reader that, as shown in Section 3, more than a third of all seal-utilizing websites are e-shops, thus holding the promise of personal and financial information that are not typically present on an average website.

5.2 Cloaking

When an attacker compromises a website, it is in his best interest to keep this hidden from the website owner. In case the adversary uses the website to host malware and infect the site's visitors, the task of hiding the compromise becomes more difficult. Not only could a change in the website alarm the website administrators, but regular crawls by various search engines will also look for the presence of malware, in order to protect the users of these search engines.

To prevent detection, attackers make use of *cloaking*, where the malware distinguishes between visits of crawlers and human users and only exposes itself to the latter. Among others, attackers can use implementation-specific JavaScript code to distinguish between JavaScript engines (and thus their housing browsers), as well as cloaking at an IP-address level [28].

For some seal providers, we noticed that their scanners do not execute JavaScript. As such, an attacker can simply hide the presence of malware by testing for JavaScript support. Alternatively, attackers can, unfortunately, always resort to cloaking at a network level. During the tests described in Section 4.3 we witnessed that the scanning requests of seal providers were always originating from the same IP range, often a block that is registered to the seal provider. It would thus be straightforward for an attacker to only expose his malware in case a request does not originate from an IP address related to a seal-provider. This way, an attacker could easily compromise a seal-utilizing website, while the website owner would remain under the impression the website was still secure as a consequence of the daily or weekly successful seal scans. Note that this detection can be done in a straightforward manner, and is already used by attackers for conducting blackhat SEO [18].

Next to attackers, website owners could also be interested in hiding weaknesses from a seal provider. In case a seal provider finds vulnerabilities on a website, it may take a considerable amount of time and resources for the website administrator to mitigate them. If this does not happen within the grace period provided by the seal provider, the security seal – a product the webmaster paid for – will disappear. Hence, in some cases, it could prove very useful for a webmaster, if the security provider is not able to find any vulnerabilities. As such, a webmaster may be tempted to also employ a cloaking technique to deceive seal providers. In our vulnerable webshop experiment, we managed to circumvent the detection of vulnerabilities by rerouting all traffic originating from seal providers to a static web page. For all seal providers, this could be done by merely adding two lines to our webserver's configuration file. This, unfortunately, requires much less effort than continuously mitigating vulnerabilities, and could thus be employed by sketchy website owners who just want to convince their customers their website is secure.

5.3 Abusing security seal services

In earlier sections, we showed how seal providers can be abused to attack the websites that they certify as secure. In this section, we describe how they can also be weaponized against users, as well as against third-party websites.

Phishing

In an attempt to acquire sensitive credentials for websites, adversaries often create phishing pages which typically resemble the original website which the phisher is targeting. To trick a user in entering her credentials, attackers can try a series of techniques to make the victim believe that she is on the legitimate website. For instance, by registering a domain that looks similar to the original domain, attackers can often convince users they are on the genuine website. Additionally, when the original website contains a security seal, the attacker could replicate this seal on his phishing webpage to increase his credibility. Moreover, if the claims from seal providers are correct, i.e., that the appearance of a seal leads to an increase of trust in the webpage, the victim will feel safe on the phishing page.

To counter this type of attack, seal providers should only allow a seal to be included from the authentic website which they certify. In our evaluation, we found that two seal providers would not display a security seal in case the `Referer` header did not match the sealed website. The seal, however, would appear if the referrer header of a user's HTTP request was absent, which can be trivially achieved by the use of the appropriate value for the `meta referrer` HTML tag. As a result, an attacker can currently include a security seal on his phishing page from *all* ten seal providers that we evaluated. Note that these seals are fully functional in the sense that the potential phishing victims can click on them and be assured, by a page hosted the seal-provider's domain, that the seal is legitimate (not just a copy) and that the seal-bearing website is secure. If a user is already considering a phishing page enough to click on the security seal, it is unlikely that he will spot the fact that the domain mentioned in the seal provider's page is different than the one of the phishing page.

Attacking third-party websites

Since seal providers search for vulnerabilities on websites, it is obvious that only the webmaster of a specific website should be able to request a vulnerability scan for that website. This is especially important since the attacks will not be launched directly by the attacker, thus making him harder to trace by the victim website at a later time.

Even though seal providers do attempt to verify ownership of a website, we found that often their methods are bypassable. For instance, several seal providers performed owner verification through the upload of a specific file on the website that requested a seal. The uploaded file should have a specific randomly-generated filename and contain a randomly-generated string. For three seal providers, we discovered that the contents of the uploaded file did not have to be an exact match as the ones provided. Consequently, if an attacker would be able to partially control the content on a URL containing the requested filename, he would be able bypass the ownership verification and get a security report for that domain. While this may seem unlikely, for several websites this can be easily achieved. On Twitter, for example, users are appointed their own URL containing their username, so a user named `foobar` is reachable at `http://twitter.com/foobar`. As such, an attacker can register an account with a username equal to the filename requested by the seal provider, and include the contents of the file in his first or last name. The seal provider will then successfully discover the "uploaded file" and proceed to perform a security scan and report the discovered issues to the attacker.

6. DISCUSSION

In previous sections we showed that, given the current state-of-practice, third-party security seals are not only of limited value, but that they can also be used to attack seal-using websites, as well as their users, and third-party websites. In this section, we briefly describe the ways in which seal providers can substantially better their services.

In terms of vulnerability discovery, we witnessed that the vulnerability scanners of some seal providers were not ex-

ecuting client-side JavaScript. Given the ubiquitousness of JavaScript, we argue that JavaScript support is a necessary feature of any modern vulnerability scanner. For the seal providers that did find some vulnerabilities, we believe that their tools can be bettered if they are tried against web applications with known vulnerabilities so that the developers can quantify the coverage of their tools and prioritize the development of support for the missing functionality. For the seal providers that found no vulnerabilities whatsoever, it is clear that either their tools are fully ineffective, or that they are trying to combine incompatible technologies, e.g., searching for web application vulnerabilities using a network-level scanner.

For the problem of cloaking, a straightforward solution is to employ, from time to time, the use of VPN or cloud services, to ensure that the IP addresses of the scanners are not publicly traceable back to the seal provider. The resulting pages can then be compared to pages retrieved from the seal provider's usual IP block using a wide range of techniques, such as text shingles [6], screenshot comparison, or comparison of the HTML structure between the two pages. Pages with large differences can be manually inspected by a human analyst who can then reach out to the webmaster of the seal-using website. To avoid being abused by phishing pages, seal providers can stop showing the security seal when the referrer header is absent, or does not match the certified website. All browsers allow, by default, the sending of the referrer header, thus the change will not affect the majority of web users who do not alter the default configuration of their browsing software.

The problem of seals being used as a vulnerability oracle is, unfortunately, not straightforward to solve. When a seal provider hides a seal as a reaction to a discovered vulnerability, this event is detectable and abusable by an attacker, as discussed in Section 5.1. Alternatively, if a seal provider does not hide, or in some way alter, the presence of a seal when a vulnerability is detected, then the certification power of a seal is compromised because any website can acquire it regardless of the presence or absence of vulnerabilities. Thus, given the current architecture of security seals, the honesty of a seal provider, and the security of a seal-using website seem to be contradictory goals.

The only solution to this conundrum appears to be a vigilant webmaster. If a webmaster is able to fix a discovered vulnerability within the grace period allotted by the seal provider, then the vulnerable website can avoid detection by attackers, while the seal provider can maintain its certification honesty. As such, the companies that do not currently provide a grace period – possibly thinking that this results to higher security standards – are actually doing a disservice to their clients.

7. RELATED WORK

While there has been some evidence of the improper certification done by security seal providers, this evidence is either anecdotal or gathered in an ad-hoc manner for a handful of websites [16, 34]. To the best of our knowledge, this paper is the first to systematically evaluate the certification of popular third-party security seals, at a large scale and from a security point of view. In this section, we review prior work on all aspects of third-party seals and the effectiveness of automated vulnerability scanners.

Third-party Seals

Third-party seals originally received attention from the user-interface and economics community, where researchers tried to identify whether a seal is recognizable and whether it leads to an increase of trust and thus increased sales for the seal-bearing website. In 2002, Head and Hassanein [10] discovered that the awareness and influence of third-party seals was low and calling for more research on the placement and appearance of seals that would increase their awareness level. Belanger et al. [4] studied the role of various factors in how trustworthy a website appears to consumers, including third-party security seals. The authors discovered that the users valued security features (such as the presence of encryption) more than security seals. Kimery et al. discover that while a consumer's attitude in online merchants is positively correlated with the consumer's trust of those merchants, the presence of third-party seals does not affect the consumer's trust [20].

Interestingly, not all researchers have reached the same conclusions. Houston et al. [14] survey 106 students and discovered a positive correlation between the presence of a seal and the perceived quality of a product which is in turn correlated with a consumer's willingness to buy. Hu et al. [15] empirically examined the effects of various seals and, contrary to the aforementioned research, concluded that most seals that deal with trust *do* in fact increase a consumer's intent to purchase. Later research by Kim et al. [19] showed that seals have a positive effect of reducing security-related concerns of consumers, but only *after* the consumers are educated about security and privacy threats, as well as the role of third-party security seals.

The seal providers themselves rely mostly on testimonials by existing clients in order to convince businesses to adopt third-party seals. The majority of the testimonials, as well as much of the seal-provider's own advertising, revolve more around the increased sales and monetary gains resulting from the adoption of a security seal, rather than the number of vulnerabilities discovered.

Edelman [8] approached third-party seals from a different angle. Instead of investigating the effect of seals on consumers, he investigated the trustworthiness of sites that employ third-party trust seals. Trust seals are different from the security seals that we investigated in this paper, in that they certify a merchant's trustworthiness, instead of their security. These trust seals are typically focusing on the privacy policy of online merchants and attempt to ensure that the merchant does not abuse the details collected by their customers. Edelman argued that a trustworthy company does not need an external entity to certify its trustworthiness. By using "site safety" information provided by SiteAdvisor, he found that websites using these seals are more likely to be untrustworthy than websites that do not use trust seals.

James and MacDougall investigated the security seals of McAfee and Trust-Guard and also reached the conclusion that the vanishing seals could be used as a security oracle [17]. The authors implemented a tool called Oizys which would automatically discover missing seals and report the potentially vulnerable websites. Our attack described in Section 5.1 expands upon their work by, not only locating vulnerable websites, but also discovering the exact vulnerabilities of each website via the use of a malicious proxy.

Effectiveness of Automated Vulnerability Scanners

A core component of a third-party security seal vendor is the automated vulnerability scanner that dictates whether a website should be "awarded" the seal or not.

Doupé et al. conducted the most recent survey of the thoroughness of black-box automated vulnerability scanners, by evaluating eleven different scanners against a known vulnerable web application [7]. The authors discovered that many of the evaluated scanners could not cope with modern web application technologies, such as the presence of JavaScript, and thus could not fully explore websites, much less discover all known vulnerabilities in a test web application.

One difference between security seals and generic web application scanners that is worth mentioning, is the latter's higher tolerance for false positives. In a common penetration test, the tester can go through a relatively large number of false positives without any adverse effects for the tested service. Contrastingly, security seal vendors do not have this luxury, as false positives that stop the seal from appearing on their clients websites will not be tolerated by the clients who pay for the certification of their websites. As such, it is possible that the scanners of third-party seal providers choose to err on the safe side in order to lower or altogether avoid false positives which, almost unavoidably, will result in less true positives.

8. CONCLUSION

Providers of security seals claim that websites that make use of their services will appear more trustworthy to the eyes of consumers and will thus have an increase in their sales. In this paper, we put the security guarantees of seal providers, i.e., the guarantees that indirectly influence a consumer's feelings of trust, to the test. Through a series of automatic and manual experiments, we discovered that third-party security seals are severely lacking in their thoroughness and coverage of vulnerabilities. We uncovered multiple rudimentary vulnerabilities in websites that were certified to be secure and showed that websites that use third-party security seals do not follow security best practices any better than websites that do not use seals. In addition, we proposed a novel attack where seals can be used as vulnerability oracles and describe how an attacker can abuse seal providers to discover the exact exploit for any given vulnerable seal-using website.

Overall, our findings show that current state-of-practice of third-party security seals is far from ideal. While we propose steps that seal providers can take that will substantially increase the accuracy and effectiveness of their security certification, the issue of inadvertently creating a vulnerability oracle seems to be central to the current architecture of security seals and appears to not have a technical solution which does not sacrifice, either the honesty of a seal provider, or the security of the certified website.

Acknowledgments: We want to thank the anonymous reviewers for the valuable comments. This research was performed with the financial support of the Prevention against Crime Programme of the European Union (B-CCENTRE), the Research Fund KU Leuven, and the EU FP7 projects NESSoS and STREWS.

9. REFERENCES

[1] L. Arsene. Xbox Live Accounts of Microsoft Employees Hacked Using Social Engineering. `http://www.hotforsecurity.com/blog/xbox-live-accounts-of-microsoft-employees-hacked-using-social-engineering-5716.html`.

[2] A. Barth. HTTP State Management Mechanism. *IETF RFC*, 2011.

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

[4] F. Belanger, J. S. Hiller, and W. J. Smith. Trustworthiness in electronic commerce: the role of privacy, security, and site attributes. *The Journal of Strategic Information Systems*, 11(3):245–270, 2002.

[5] P. Bright. Anonymous speaks: the inside story of the HBGary hack. `http://arstechnica.com/tech-policy/2011/02/anonymous-speaks-the-inside-story-of-the-hbgary-hack/`.

[6] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic Clustering of the Web. *Computer Networks and ISDN Systems*, 29(8-13), Sept. 1997.

[7] A. Doupé, M. Cova, and G. Vigna. Why Johnny Can't Pentest: An Analysis of Black-box Web Vulnerability Scanners. In *Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, Bonn, Germany, July 2010.

[8] B. Edelman. Adverse Selection in Online "Trust" Certifications. In *Proceedings of the 11th International Conference on Electronic Commerce*, ICEC '09, pages 205–212, 2009.

[9] S. Fox. Pew Research : 51% of U.S. Adults Bank Online. `http://www.pewinternet.org/2013/08/07/51-of-u-s-adults-bank-online/`.

[10] M. M. Head and K. Hassanein. Trust in e-commerce: evaluating the impact of third-party seals. *Quarterly Journal of Electronic Commerce*, 3:307–326, 2002.

[11] OpenSSL 'Heartbleed' vulnerability (CVE-2014-0160). `https://www.us-cert.gov/ncas/alerts/TA14-098A`.

[12] J. Hodges, C. Jackson, and A. Barth. HTTP Strict Transport Security (HSTS). *IETF RFC*, 2012.

[13] J. Horrigan. Pew Research : Online Shopping. `http://www.pewinternet.org/2008/02/13/online-shopping/`.

[14] R. W. Houston and G. K. Taylor. Consumer Perceptions of CPA WebTrust assurances: Evidence of an Expectation Gap. *International Journal of Auditing*, 3(2):89–105, 1999.

[15] X. Hu, Z. Lin, and H. Zhang. Trust promoting seals in electronic markets: an exploratory study of their effectiveness for online sales promotion. *Journal of Promotion Management*, 9(1-2):163–180, 2002.

[16] T. Hunt. Why I am the world's greatest lover (and other worthless security claims). `http://www.troyhunt.com/2013/05/why-i-am-worlds-greatest-lover-and.html`.

[17] J. James and S. MacDougall. Usine McAfee secure/trustguard as attack tools. In *DerbyCon*, 2012.

[18] J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi. deSEO: Combating Search-result Poisoning. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, 2011.

[19] D. J. Kim, C. Steinfield, and Y.-J. Lai. Revisiting the role of web assurance seals in business-to-consumer electronic commerce. *Decision Support Systems*, 44(4):1000–1015, 2008.

[20] K. M. Kimery and M. McCord. Third-party assurances: the road to trust in online retailing. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*. IEEE, 2002.

[21] B. Krebs. A First Look at the Target Intrusion, Malware. `http://krebsonsecurity.com/2014/01/a-first-look-at-the-target-intrusion-malware/`.

[22] B. Krebs. Adobe Breach Impacted At Least 38 Million Users. `http://krebsonsecurity.com/2013/10/adobe-breach-impacted-at-least-38-million-users/`.

[23] A. Langley. Apple's SSL/TLS Bug. `https://www.imperialviolet.org/2014/02/22/applebug.html`.

[24] M. Marlinspike. New Tricks for Defeating SSL in Practice. *Blackhat*, 2009.

[25] McAfee. The Economic Impact of Cybercrime and Cyber Espionage. `http://www.mcafee.com/sg/resources/reports/rp-economic-impact-cybercrime.pdf`.

[26] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. You are what you include: large-scale evaluation of remote javascript inclusions. In *Proceedings of the 2012 ACM conference on Computer and Communications Security (CCS)*, pages 736–747, 2012.

[27] N. Olivarez-Giles. Snapchat Data Breach Exposes Millions of Names, Phone Numbers. `http://blogs.wsj.com/digits/2014/01/01/snapchat-alleged-leak-4-million-users/`.

[28] M. Rajab, L. Ballard, N. Jagpal, P. Mavrommatis, D. Nojiri, N. Provos, and L. Schmidt. Trends in circumventing web-malware detection. *Google, Google Technical Report*, 2011.

[29] D. Reisinger. Syrian Electronic Army hacks Forbes, steals user data. `http://www.cnet.com/news/syrian-electronic-army-hacks-forbes-steals-user-data/`.

[30] D. Ross and T. Gondrom. HTTP Header X-Frame-Options. *IETF RFC*, 2013.

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

[32] Trend Micro Site Safety Center. `http://global.sitesafety.trendmicro.com/`.

[33] M. Vasek and T. Moore. Identifying Risk Factors for Webserver Compromise. In *Proceedings of the Eighteenth International Conference on Financial Cryptography and Data Security*, FC' 14, 2014.

[34] J. Vijayan. 'Hacker Safe' seal: Web site shield, or target? `http://www.computerworld.com/s/article/9057878/_Hacker_Safe_seal_Web_site_shield_or_target_?`

[35] M. West. Play safely in sandboxed IFrames. 2013.

APPENDIX

Client-side Security Mechanisms

Here we present brief descriptions of the client-side security mechanisms that we used as indicators of an overall "security hygiene" when comparing seal-using websites, to websites of an equivalent ranking and category that do not use seals.

- **HTTP Strict Transport Security (HSTS):** HSTS is a security policy mechanism where a web server can force complying browsers to interact with it using only HTTPS connections [12]. By sending out the HSTS policy via the appropriate HTTP header, a web server specifies a period of time during which the browser is instructed that all requests to that website need to be sent over HTTPS, regardless of what a user requests.
- **Secure Cookies:** Using the `Secure` flag on `Set-Cookie` headers limits the scope of a cookie to only secure channels [2], making the cookie less likely to be stolen by a MitM attacker.
- **HttpOnly Cookies:** Cookies are, by default, accessible to JavaScript code, which can lead to the theft of cookies in an XSS attack. To defend against this, a website operator can use the `HttpOnly` flag on cookies, making them unavailable to client-side JavaScript.
- **Content Security Policy (CSP):** To mitigate a wide range of injection vulnerabilities, such as Cross-Site Scripting (XSS), a website operator can make use of the CSP mechanism. CSP provides a standard HTTP header that allows website owners to declare approved sources of content that browsers should be allowed to load on any given webpage [31]. Whenever a requested resource originates from a source that is not defined in the policy, it will not be loaded.
- **X-Frame-Options (XFO):** When an attacker is able to load a website, or part of a website in a `frame` or `iframe` element, the website might be vulnerable to Clickjacking attacks. The XFO response header can be used to instruct a user's browser whether a certain page is allowed to be embedded in a frame [30].
- **iframe sandboxing:** The `sandbox` attribute for the `iframe` element, introduced in HTML5, enables a set of extra restrictions on any content loaded in a frame, which can be used to limit the capabilities given to untrusted framed pages [35].
- **Anti-CSRF Tokens:** The "best practice" defense for Cross-Site Request Forgery (CSRF) attacks is the inclusion of a secret long random token (also known as *nonce*) with each request, and validation of that token at the server side [3]. To check for nonces, we searched for forms that contained a hidden form element that was most likely used as a nonce. More specifically, form elements were marked as nonces when their name contained the keywords "token", "nonce", or "csrf", and when their value was a long alpha-numerical string.
- **X-Content-Type-Options:** Internet Explorer has a MIME-sniffing feature that will attempt to determine the content type for each downloaded resource. This feature, however, can lead to security problems for servers

hosting untrusted content. To prevent MIME-sniffing, a web server can send the `X-Content-Type-Options` response header with the `nosniff` value.

- **SSL-stripping Vulnerable Form:** For performance reasons, some websites only implement HTTPS for certain webpages that contain sensitive information (such as a log-in page), which may result in forms vulnerable to SSL stripping [24]. As a result, a MitM attacker can replace all HTTPS form links on the HTTP page to HTTP links, which will allow the attacker to later intercept sensitive form data.