Copyright Root Services, 2014

# Rise in Web Application Intrusions 2013-2014

Alexander Chopra[1]*, Michael Kaufman[2]

**Abstract**
This article goes into detail about the recent Verizon report on breech investigations as well as Verizon's PCI report regarding intrusions. The article goes into detail regarding the various vulnerabilities as well as provides a high level understanding of some of the issues with the modern web in regards to web applications. Precautions are explored in ways that organizations and individuals can try and prevent these types of intrusions.

**Keywords**
Web Intrusions — Application Intrusions — Payment Card Industry — Web Application Services

[1] *Research and Development, Root Services, Rockville, Maryland, United States*
[2] *Research and Development, Root Services, Rockville, Maryland, United States*

## Contents

| | | |
|---|---|---|
| | Introduction | 1 |
| 1 | Popularity of Web Apps | 1 |
| 2 | Recent Security Issues | 1 |
| 3 | Verizon PCI Report | 2 |
| 4 | Vulnerabilities in the Web Application Layer | 2 |
| 5 | Organizational Responsibility | 3 |
| 6 | Precautions | 3 |
| 7 | Conclusion | 4 |
| | Acknowledgments | 4 |
| | References | 4 |

## Introduction

Web applications have increased in popularity since the introduction of the first graphical web browser in 1993 known as Mosaic (later renamed to Netscape). The ability for a web application to exist without direct install on a large number of cross-platform client computers through a web browser, spurred the popularity of web applications. The concept of web applications were introduced in 1999 in the Java Servlet Specification version 2.2[1]. This new concept allowed resources to be run together and respond to requests representing an application. The continual development of standardization between browsers led to the finalization of HTML5 which improved upon the capabilities of the web browser to present various media without client side installs. Corporations are increasingly utilizing web applications internally as well as for their clients. Web applications are heavily used today, ranging from email applications such as Gmail or Outlook to more advanced applications such as project management software (ex. C-BRAT), point of sale software (ex. Imonggo) or Collections Software (TalionisLaw).

## 1. Popularity of Web Apps

The increased popularity surrounding web applications and the industry also shifted the focus of attackers to web applications. The security behind web applications should be considered as a result of the vastly increased amount of consumer and private data that is being stored through these applications. Web applications are generally separated into three sections: the presentation layer, the application layer and the storage layer. The web browser is generally the presentation layer and works in conjunction with dynamic content languages such as ASP or PHP and the storage layer to create a rich web based application for the user. The interaction between the client and the server must be controlled in such a way that the client is unable to gain access or manipulate server-side information. Vulnerabilities exist that have been proven effective to gain access to sensitive information within a web application as a result of not following industry best practices in relation to security in the development of the application.

## 2. Recent Security Issues

Recently there have been many breaches of security of web applications, allowing attackers to gain access to sensitive enterprise information as well as consumer information of those businesses, resulting in public outrage in regards to privacy

and expected security of their data. On December 18, 2013, a major retailing company, Target Corporation, announced that they were investigating a major data breach involving over 110 million customers[2]. In the case of Target, credentials from Fazio Mechanical Services, a HVAC servicing company for Target, were stolen, allowing access to Target's network and internal infrastructure[3]. With access to Target's network, the attackers uploaded a piece of malware called "BlackPOS" to their POS systems for the purpose of stealing credit card information. A corporation as large as Target did not take proper precautions in regard to their network and POS system, failing to separate their payment processing systems from the rest of their network. With the emergence of more and more systems that exist in web applications, proper security should be employed to best thwart attacks upon the system to best preserve customer and internal data from attackers.

An increasing number of web applications are becoming compromised as a result of the increased use of them by organizations utilizing the new power it provided. The allocation of resources and funding towards security is often difficult to justify because the benefits that are derived from such expenditures are only seen in the case when there is a breach. If there are no breaches in the system it would be difficult to estimate the amount of actual losses that are avoided by this investment[4]. In the previously discussed case of Target Corporation, the damage experienced by the company included far more than just loss of sales but also the loss of consumer trust. The "lost customers and potential liabilities" that are associated with a data breach are difficult to predict; Corporations rather increase profits than decrease costs[4]. The investment in proper allocation of resources towards security is often not rewarded as it should until a breach actually occurs.

## 3. Verizon PCI Report

Verizon's 2014 PCI (Payment Card Industry) Compliance Report shows that less than 15 percent of corporations are appropriately holding card holder data[5]. Their Data Breach Investigations Report for 2014 indicates that over 85 percent of the security breaches that are seen are associated with large corporations[6]. The Data Security Standard put forward by the PCI Security Standards council attempt to focus an organization on objectives ranging from encryption of stored customer data to proper configuration of access controls. In the case of Target Corporation, proper access controls should have prevented the stolen credentials to be utilized in the manner they were, eventually allowing access to the entirety of Target's network.

Verizon notes that over 85 percent of internal issues surrounding web applications deals with access control and privilege misuse. Users with access to the system often have much more access than they realistically need to carry out their duties, making the possibility for exploitation through those accounts much more feasible. Over half of all breaches and incidents that occur deal with point of sale systems and web application attacks. From 2011 to 2013 the number of attacks on point of sale systems in comparison with web application attacks were inversely proportional; attackers are shifting their focus to exploiting web applications and focusing less and less on point of sale systems[6]. In the report produced by Verizon, recommendations are made towards implementation of two-factor authentication, straying away from dynamic code execution (often included in various CMS), input validation, lockout policies and active monitoring or filtering of network resources[6]. Web applications are generally attacked either through a vulnerability that is found within the web application itself or through credential misuse.

## 4. Vulnerabilities in the Web Application Layer

It is estimated by information technology research and adviser Gartner, that over 70 percent of all vulnerabilities are seen in the web application layer. The process of launching a web attack begins through analysis of the given remote system that is targeted for this attack, monitoring traffic between the client and the server and evaluating dynamic pages and associated validation issues[7]. Two popular methods of exploitation of web applications are SQLIAs and XSS. SQLIAs, or structured query language injection attacks, deal with the user interjecting additional SQL keywords and operators into the query being carried out. This attack method can be carried out in a multitude of ways including: SQL injection through the client's input into the web application, SQL injection through the client's stored cookies on their local system, manipulation of server variables and second-order injection[8].

Often SQLIAs that are crafted through the input of the client target various form submission fields that are seen on the web application. If the data that is entered into these fields is not properly sanitized, users would be able to inject their own SQL commands that the server would carry out as its own. SQLIAs can also be launched through manipulation of cookies that are left on the client's device from the server. Often SQL queries are built based on the contents of a cookie (restoring state information for example) and can be utilized to launch an attack if manipulated to inject their own queries into the system. Attackers often forge network headers in such a way that an SQLIA could be launched directly from the headers. Second-order injections deal with the possibility of an attacker launching SQLIAs to be executed at a later time, which creates particular difficulties in tracking down the attack as the injection is often different than the attack point[8]. SQLIAs can result in many differing outcomes ranging from extracting data from the database to executing remote commands resulting in a denial of service.

Direct input could be used by an attacker in a SQLIA to be returned results in addition to the results that are expected to be displayed. If the application is not sanitizing the input, an attacker could execute a SQL query to list the various tables in the application. If the attacker has knowledge of names of

other tables being perused and is able to directly pass queries to the server, could gain access to additional information not intended through the application itself. Queries can be made by an attacker that attempt to trick the web application into ending the intended query and attaching their own query to the original. Often it is seen that attackers attempting to launch SQLIAs utilize alternate encoding to assist in avoiding detection. The utilization of alternate encoding allows the attacker to pass an expression, not encoded as expected by the web application without meaning in the application, to be later interpreted by another layer, resulting in the interpreter assisting in the attack[8].

XSS attacks deal with tricking the client into thinking what is being displayed back to them from a form is actually what is being served from the server handling the form but in reality it is coming from themselves. Often attackers utilize this to have the browser parse the data after submission and display their work for them as opposed to accessing the server. There are a few ways currently proposed to prevent XSS attacks, the most common being output filtering. Output filtering allows the server to filter the data that is submitted by the user before being re-displayed to the user ensuring that the browser does not interpret the input as code to be displayed back to the client. An attacker could have their own pre-loaded links or emails forwarding users to pass their own code to the application, making the client see the data they passed through on interpretation. The client in this case would think that the return is coming from the server they are trying to fill, but rather it is coming from themselves. The code that is put forward is not left on the remote server because it was never there, it was only on the client's computer at which was accessing the form. A client for example could be misled into thinking data they are entering is going straight to their intended target (the server), but in reality is going to a third party collecting this information. XSS attacks are often seen in anticipation of holiday sales or promotional materials where newsletters or other information from a website is expected that can be easily forged, resulting in a perfect opportunity to launch such an attack[9]. Malicious file includes are often utilized by attackers who take advantage of servers that do not properly validate user defined parameters[10].

## 5. Organizational Responsibility

Organizations should heed the warning of recent previous high profile attacks to allocate the time and resources needed to more adequately protect their web applications. There is a growing trend seen in attacks on web applications and this can be seen happening at an increased rate from 2013-2014 in comparison with previous years. Ensuring system software is up to date and patched to the most recent version is often a tedious task with the amount of clients and services to keep up to date, but will have a large impact on the number of vulnerabilities that are seen. Attackers take advantage of slow patch times and often launch attacks based on the vulnerabilities that are exposed in patch notes. Keeping an organization's servers as up to date as possible minimizes the chances of an exploit happening that has been already realized. Active monitoring of the network and infrastructure from an organizational perspective will allow irregularities to be more easily identified and dealt with before the issue becomes too large. Consistent analysis of web application updates and associated vulnerabilities that become apparent are needed to ensure a high level of security within the organization's infrastructure.

## 6. Precautions

Often Apache is not configured as well as it could be in defense of possible attacks, leaving the server open to more vulnerabilities than necessary. Something as simple as ensuring that a separate group and user are created for Apache processes. In case of a breach, it would be beneficial to limiting the user running those processes to a separate user than root. It is commonly seen for an Apache server to have far more modules than which are required for the job, creating potential vulnerabilities in the system unnecessarily. With the rise of attacks on web applications hosted on Apache servers, it is needed to take precautions against possible attacks. Modules such as httpd-core, mod-access, mod-auth, and other required modules for basic functionality should remain enabled and all other modules disabled. A module that would also be beneficial and easily accessible for most would be Mod-Security.

Mod-Security acts similarly to Snort or Bro in addition to being classified as a web application firewall because it is designed specifically to secure web applications[11]. Mod-Security has the ability to perform very unique filtering on requests and traffic, various payload analysis options as well as detailed logging. Mod-Security is so similar in its usage of filtering and rule sets as Sort or Bro, a simple PERL script can be used to directly convert Snort rules to Mod-Security formatted rules. Snort has a very large public collection of rules and conversation available for download from their community portal, making the process of transition to Mod-Security and easily setup previously trusted rules much simpler. Creating rules that are unique for the web application that is running on the Apache server being protected must take consideration in evaluating what is attempted to be looked for. Analysis and ensuring a vast logging process created by the Mod-Security module would allow one to have adequate information to understand what happened in the event of an incident.

Active monitoring requires the use of resources that are often not allotted to secure web applications. Ensuring that there is an active reviewing process of activities and processes on the server is important. It is possible to bypass the processes attempting to secure the server altogether and attack the Apache binaries themselves, creating the possibility for further exploit without notice[12]. The Blackhole exploit targeted Apache servers by replacing their core binaries with compromised binaries. The process sat in shared memory to avoid detection. These sort of intrusions are very diffi-

cult to detect because of how they are created and often are only found through careful examination of the system and resources are required for that. It is common for web applications to have some sort of maintenance feature, allowing an administrator to log into the application to make changes or updates. Often the administrative credentials and accounts are not held to the same standard as other accounts because they were thought to be fully trusted to begin with[13].

## 7. Conclusion

The rise of attacks on web applications should have been foreseen so that measures to combat the attacks could have been enacted. Unfortunately, as seen recently, large corporations such as Sony, PBS, Epsilon and Target (to name a few) have not allocated the necessary resources to properly secure their systems against various types of web application intrusions, leading to data breaches and loss of customer data. The prevalence of web application based attacks will not slow down as a result of the increased shift towards web application services. Security needs to be considered a primary objective in the development process for web applications and best practices should be followed to ensure as secure of an environment as possible when handling consumer and corporate data.

## Acknowledgments

Verizon for providing detailed statistics and reports based on the 2013-2014 data collected.

## References

[1] Alex Chaffee. What is a web application? *jGuru*, 1, may 2012. http://www.jguru.com/faq/view.jsp?EID=129328.

[2] Samantha Sharf. Target shares drop after ceo gregg steinhafel's resignation. *Forbes*, 1, may 2014. http://www.forbes.com/sites/samanthasharf/2014/05/05/target-shares-drop-after-ceo-gregg-steinhafels-resignation/.

[3] Jaikumar Vijayan. Target breach happened because of a basic network segmentation error. *Computer World*, 1, feb 2014. http://www.computerworld.com/article/2487425/cybercrime-hacking/target-breach-happened-because-of-a-basic-network-segmentation-error.html.

[4] L. A. Gordon. Incentives for improving cybersecurity in the private sector: A cost-benefit perspective. *Testimony for the House Committee on Homeland Security's Subcommittee on Emerging Threats, Cybersecurity, and Science and Technology*, 1, 2007. 20071031155020-22632.

[5] Verizon. Verizon 2014 pci compliance report. *Verizon Research Report*, 1, jan 2014.

[6] Verizon. 2014 data breach investigations report. *Verizon Research Report*, 1, apr 2014.

[7] Kyle Adams. Insider's view into web application abuse: 5 phases of an attack. *Network World*, 1, jun 2011. http://www.networkworld.com/news/tech/2011/060611-cyberattack-phases.html.

[8] W. G. Halfond, J. Viegas, and A. Orso. A classification of sql-injection attacks and countermeasures. *Proceedings of the IEEE International Symposium on Secure Software Engineering*, 1:13–15, mar. Arlington, VA, USA.

[9] K. Spett. Cross-site scripting. *SPI Labs*, 1:1–20, may 2005.

[10] Imperva. Remote and local file inclusion vulnerabilities. *acker Intelligence Initiative*, 1(8), apr 2012.

[11] I. Ristic. Modsecurity handbook. *Feisty Duck*, 1, may 2010.

[12] Dennis Fisher. Attack using backdoored apache binaries to lead to blackhole kit. *Threatpost*, 1, apr 2013. http://threatpost.com/attack-using-backdoored-apache-binaries-to-lead-to-blackhole-kit/99973.

[13] Y. W. Huang, S. K. Huang, T. P. Lin, and C. H. Tsai. Web application security assessment by fault injection and behavior monitoring. *Proceedings of the 12th international conference on World Wide Web*, 1:148–159, may 2003. ACM.