IMPERVA®

WAAR 2015

2015 Web Application Attack Report (WAAR)

Table of Contents

1. Introduction

Welcome to the annual Web Application Attack Report #6 (WAAR) from Imperva. This report contains a thorough analysis of attack and alert data sourced from the many deployments of Imperva Web Application Firewalls (WAF). If all the security measures deployed ahead of the WAF were truly effective in protecting the application, there would be no need for a Web Application Attack Report. But even with all those layers protecting the endpoint, the network, and everything in between user and application, threats still manage to sneak through to the application, proving once again that Imperva SecureSphere WAF is the last line of defense for web applications. So, until all security products are perfect, and applications can protect against all attacks, there will be a WAAR. Imperva Application Defense Center is a proud contributor of valuable cyber crime trends and shares information with the security community of customers, vendors and partners to defend better against existing and new threats.

1.1 Our Approach

In this report, our Application Defense Center (ADC) group analyzed 297,954 attacks and 22,850,023 alerts on 198 of the applications protected by Imperva Web Application Firewalls. This was done over a period of six months—from January 1, 2015 to June 30, 2015—to provide an accurate picture of today's application threat landscape, as seen in the wild, by deeply analyzing attacker behavior. The report also defines techniques used by hackers at the time of publishing. The report provides fact driven data to businesses and consumers to help strengthen their security posture and better protect their applications and infrastructure against known malicious attacks.

Key Findings:

1. **Threat Growth**
    a. The number of specific malicious attacks increased year-over-year
        i. A typical application suffered 3 times more SQL Injection attacks
        ii. A typical application suffered 2.5 times more Cross-Site Scripting attacks
    b. Everyone's at risk
        i. At least 3 out of 4 applications were targeted by each of the 8 attack types analyzed
        ii. The Shellshock Mega Trend – Shellshock attacks were detected in 100% of the applications in very similar numbers, indicating blind scanning of the Internet
        iii. Higher magnitude and frequency of attacks as soon as a vulnerability is made public
2. **Mitigation Classes**
    a. Reputation based detection becomes more effective with 78% of the total malicious requests being caught by detect-by-reputation mechanisms
3. **Targets**
    a. Content Management Systems (CMS) were attacked 3 times more often than non-CMS applications (and WordPress was attacked 3.5 times more than non-CMS applications)
    b. WordPress was targeted 7 times more for Spam and RFI attacks than non-CMS applications
    c. XSS attacks were very popular in Health applications, 10 times more than other applications

2. Key Findings Explanation

In last year's WAAR Report (WAAR #5), we noted the following trends: 1) an increase in attacks on web applications containing some form of consumer information, 2) attacks threatening more applications and persisting for a longer duration, 3) retail and financial sectors leading the pack in number of targets, 4) the United States hosting the majority of attack hosts, and 5) WordPress suffering more attacks than other CMS applications.

This year has some of the trends continuing from last year's report, such as increased SQL Injection (SQLi) and Cross-Site-Scripting (XSS) attacks and more attacks on WordPress. But this year also has a newcomer with the mega trend of Shellshock Remote Code Execution (RCE) attacks, scanning web applications on an equal basis. We conclude that the increasing availability of web attack tools and services–with computational power becoming less expensive and ubiquitous–are driving the new wave of volumetric malicious attacks. The evolution of attacks against web applications has continued with increased sophistication, magnitude, and velocity. However, there is hope thanks to the growing effectiveness of reputation-based detection mechanisms, and their ability to identify attacks by tracking previously identified malicious activity to its origins.

Threat Growth

When we compared this year's data to last year's data, we saw that the total number of attacks this year was significantly higher than last year (see below). Conventional web attacks from XSS and SQLi rose by 200% and 150% respectively continuing the trend from last year, with larger numbers and larger volumes of scanning campaigns across the Internet.

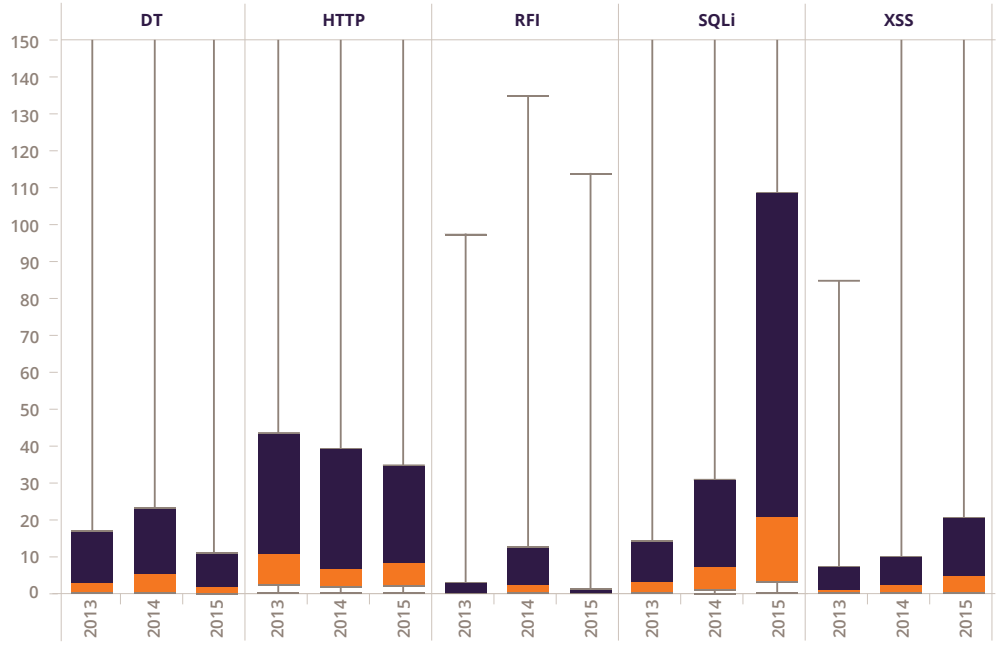


Figure 1: Comparison of Number of Incidents Between Years

Half of the applications analyzed were the target of more than 20 SQLi attacks within a six-month period. In terms of attack magnitude, the typical SQLi attack included 72 malicious requests, with the most intensive SQLi attack detected by our sensors amounting to 400,000 malicious requests. Additionally we found that Remote Code Execution attacks, in particular Shellshock, were launched against all of the applications in the research, with the typical application getting RCE-attacked 112 days over the report period, averaging more than four days per week. The volume and persistency of attacks indicate industrialization of and automation behind organized efforts. The Verizon DBIR 2015 report agrees with us, "This year, organized crime became the most frequently seen threat actor for Web App Attacks."

Attacks have become more widespread with at least three out of four applications exposed to attack attempts of each type during the report period. The most prevalent was Shellshock RCE attack, with attempts on every single application examined.

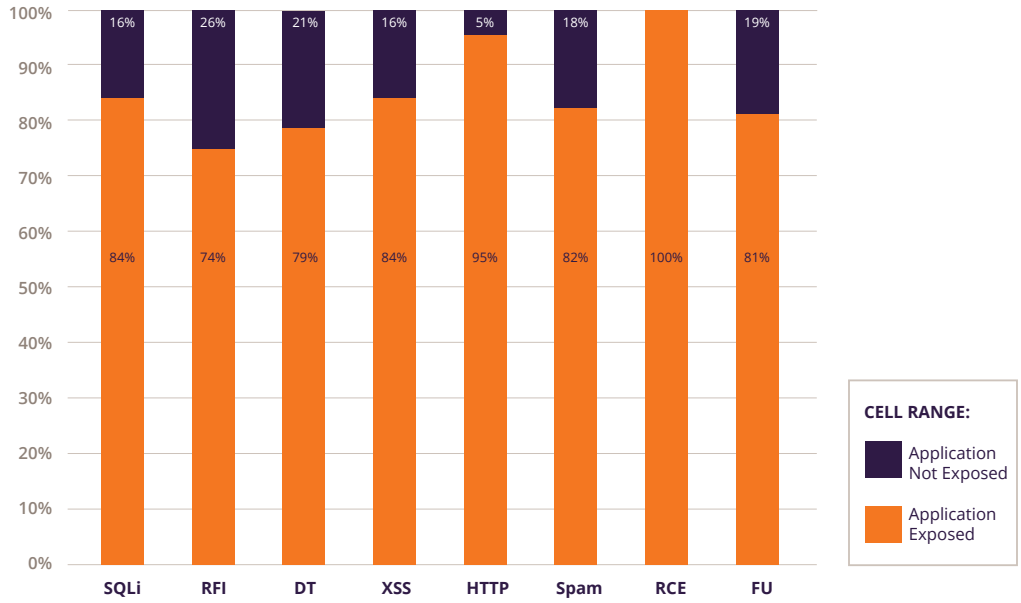


Figure 2: Web Application Malicious Traffic Exposure Ratio

Shellshock vulnerability was first made public in late September 2014. With very quick turnaround of vulnerability-to-attack, Shellshock attack vectors were integrated into exploit kits and applied in the wild by attackers within hours. Another Shellshock wave, which our sensors detected in April 2015 (captured in this report), showed similar broad and high intensity campaigns persistently attacking all of the applications analyzed. These waves also made Shellshock the main contributor for the significant increase in the number of application battle days. The RCE attack patterns in this year's report–which resembles the Heartbleed OpenSSL vulnerability–appear to have become the new norm, with quick turnaround, automation, and large campaigns.

The Verizon DBIR 2015 report also highlights that the command-and-control (C&C) type of Crimeware pattern remains the leader. We attribute part of this trend to activating C&C in Shellshock-infected applications.

Mitigation Classes

In this WAAR 6 report, detect-by-reputation alerts comprised 78% (up from 40% in WAAR 5) of the total number of alerts, or approximately 100,000 malicious requests that an application sees every month that were blocked without the application's involvement or WAF investing computational resources. There is no abnormal increase in total number of attacks, but the trend is indicating that blocking-by-reputation is yielding good results.

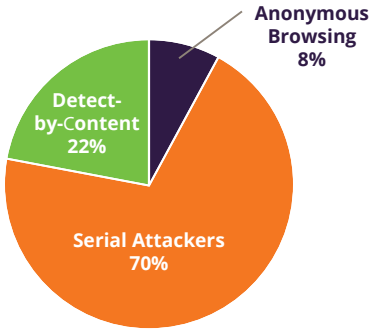


Figure 16: Contents Alerts Versus Reputation Alerts Distribution

More intensive attacks mean that the IPs from which the attacks originate are more likely to get into reputation blacklists and be classified as a Serial Attacker, and more requests from these sources being blocked using detect-by-reputation mechanisms.

This increase in automated attacks is a direct result of a higher than expected rise in profit driven cybercriminals. There have been multiple research results, highlighting organized crime growth like the one from Europol IOCTA, which concludes, "The Crime-as-a-Service (CaaS) business model, which grants easy access to criminal products and services, enables a broad base of unskilled, entry level cybercriminals to launch attacks of a scale and scope disproportionate to their technical capability and asymmetric in terms of risks, costs and profits."

Targets

The attraction of attackers to CMS applications (which are attacked 3 times more often than non-CMS applications) and in particular to WordPress is not new. CMS frameworks are mostly open source, with communities of developers continuously generating sequences of plugins and add-ons, without concerted focus towards security. This developer model constantly increases the vulnerabilities in CMS applications, especially for WordPress which is also PHP based. We found that WordPress was attacked 3.5 times more often than non-CMS applications. Typically, WordPress and other CMS applications are derived from a common template, enabling automated scanning attacks that work effectively on multiple sites.

Application Type	DT	FU	HTTP	RCE	RFI	SQLi	Spam	XSS	Total
WordPress	39	178	117	768	393	54	1893	55	3497
Other CMS	119	134	280	1031	74	83	833	239	2793
All CMS	90	150	221	935	190	72	1219	172	3049
Non CMS Applications	29	65	111	418	53	48	254	32	1010

Figure 4: Attack Incidents Average per Application for CMS Slices

Unsurprisingly, the number of RFI attacks in WordPress (on average 393 per application during the report period) is significantly higher than the non-CMS applications, most probably due to WordPress being PHP based. This trend was first observed in our 2013 "CMS Hacking" research and was also confirmed by the Verizon DBIR 2014 report.

Healthcare web applications suffer substantially more XSS attacks than other sectors. When excluding SPAM and RCE, 57% of the attacks are XSS, significantly more than other sectors (with only 1%–16%). The average number of XSS attacks in the healthcare sector is almost 10 times higher than the other industries. Healthcare is possibly attractive for hijacking sessions through XSS, for the sake of stealing Personal Identifiable Information (PII).

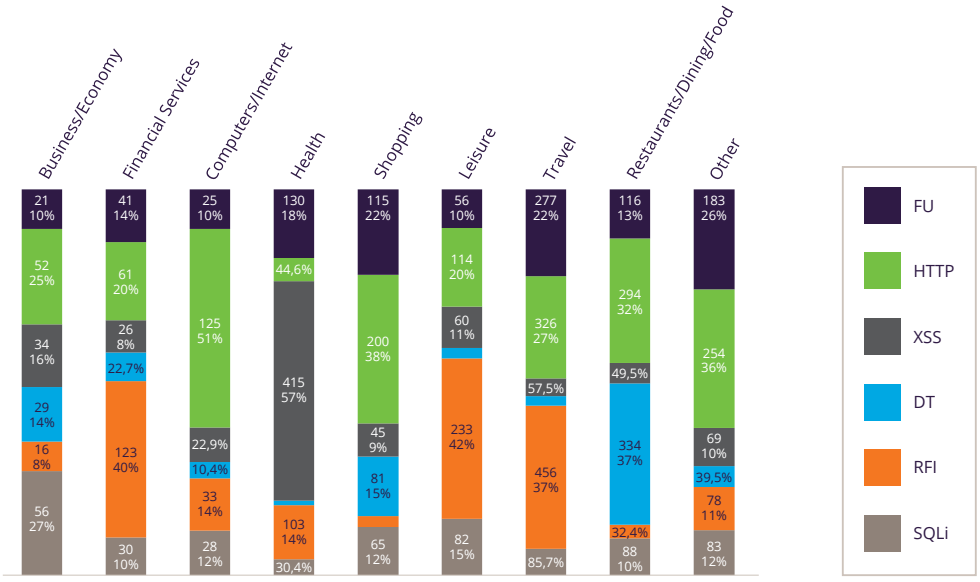


Figure 5: Alerts Proportion by Industry Verticals (Excluding Spam and RCE)

3. Analysis Methodology

3.1 Data Corpus

This report is based on data gathered from monitoring and analyzing 198 web applications from January 1, 2015 through June 30, 2015. Automated tools recorded the web applications' traffic and malicious events were documented in log files. ADC Security experts analyzed this data using special-purpose software and knowledge base.

3.2 Analysis Principles

The analysis and presentation methodology in this report follows previous reports, with several changes which are explained below. Similar to previous reports, the main notion in the report is the ***Attack Incident***, which is a burst of malicious requests of the same attack type that exceeds a threshold number within a five-minute period. These threshold numbers are specific per attack type. SQLi incidents are bursts of at least 20 requests, HTTP Attack incidents are bursts of at least 10 requests, and XSS and Directory Traversal (DT) incidents include at least five requests. For the other attack types—Remote File Inclusion (RFI), Spam, Remote Code Execution, and File Upload (FU) attack—a single request is sufficient for an attack incident. An attack incident ends when a five-minute period lasts without malicious requests seen. In order to avoid double counting of distributed attacks and attacks originating from dynamic IPs, the aggregation of requests into incidents is IP agnostic.

We also define a broader concept of ***"Battle Days"***, which are days in which an application experienced at least one attack incident.

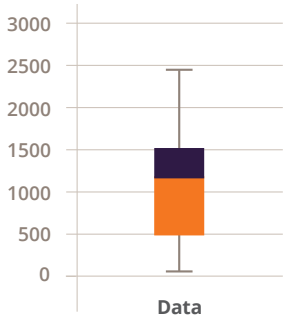
We analyze attacks against web applications in different ways to address questions with practical security implications:

- How many attack campaigns occur in a given period of time?
- How long does each attack last?
- How intense is an attack campaign; that is, how many malicious requests are issued as part of the attack?
- If an application suffered an attack incident yesterday, how likely will they suffer an attack today?

We use statistical analysis to answer these questions, and provide descriptive statistical measures for representing the attack trends, including mean, median, and quartiles.

In order to show year-to-year attack trends, we compare trend results to those from the previous year.

Graphically, we present the resulting numbers using box-and-whisker plots.



Box-and-Whisker Plots

The Box-and-Whisker plot is a convenient way to present variations of statistical data. The bottom and the top of the box represent the first and third quartiles; the middle line represents the median. We chose the end of the whisker to represent the 95% value of the data, and the lower whisker to represent the lowest value.

3.2.1 Differences from the Previous Reports

The analysis methods used in WAAR 6 have some differences from the WAAR 5.

- The data used for WAAR 6 was collected during the six months from January 1, 2015 to June 30, 2015—shorter than the nine month period from August 1, 2013 to April 30, 2014 used for WAAR 5.
- The classification of malicious requests to attack types is continuously evolving with the advance in the detection technology. In particular, we updated the list of attack types, adding Remote Code Execution and File Upload attacks, and removing Local File Inclusion (LFI). The attacks are explained in Section 10 below.
- We refined the definition of Attack Incident by replacing the generic threshold of 30 requests in five minutes, which we used in previous reports, with thresholds that are specific per attack type.

Whenever possible, we solve for the differences in analysis methods year-to-year by repeating last year's analysis on this year's data, or using proper normalization. If these were not possible, we have omitted the year-to-year comparison.

3.3 Terminology

TERM	DESCRIPTION
Attack Request	A single, malicious HTTP request.
Attack Incident	A burst of attack requests that exceeded an attack-type-specific threshold within a five-minute period.
Attack Incident Magnitude	The number of attack requests per attack incident.
Attack Incident Duration	The length, in minutes, of an attack incident.
Battle Day	A day in which an application experienced at least one attack incident.
Report Period or This Year (2015)	The time period of the current report from January 1, 2015 to June 30, 2015.
Previous Period or Last Year (2014)	From August 1, 2013 to April 30, 2014 – the time period of the previous report.
SQLi	SQL Injection Attack (see Section 10.1)
RFI	Remote File Inclusion Attack (see Section 10.2)
DT	Directory Traversal Attack (see Section 10.3)
XSS	Cross-Site-Scripting Attack (see Section 10.4)
Spam	Comment Spam Attack (see Section 10.5)
RCE	Remote Code Execution Attack (see Section 10.6)
FU	File Upload Attack (see Section 10.7)

3.4 Using the Report

Based on our analysis of application attacks during the report period, we uncovered some results that reinforced existing trends and statistics, and other results that deviated dramatically from previous reports. Based on our examination of web attack methods, attack sources, and incident intensity and duration, security teams can prioritize their efforts and develop plans to improve their security posture.

For example, a vendor of a Health application can learn from the report that: (1) attackers tend to mount Cross-Site Scripting (XSS) attacks on Health applications significantly more than other applications, (2) Health applications suffered on average 415 XSS attempts during the report period, and (3) that three out of four applications (including Health applications) fell victim to an attack for each of the attack types analyzed during the report period.

3.5 Structure of the Report

We begin with analysis of Web Attacks in Section 4, where we describe new attack trends and compare it to previous years. In Section 5, we analyze the effectiveness of Reputation based mitigation. In Section 6, we examine different industries, Content Management Systems, and Hosting services. In Section 7, we examine the trends in the geographic distribution of attackers. We present Shellshock Mega-Trend in Section 8.1 and additional interesting case studies in Section 8. Finally, we conclude with recommendations in Section 9.

4. Analysis of Web Attacks

4.1 Web Application Participation

In this section, we measure the popularity of web attacks by examining the percentage of web applications that were victims to each attack type—and learn that no application goes by without being attacked.

In Figure 6, we present the ratio between web applications that were exposed to malicious traffic to the ones that were not exposed for different attack types.

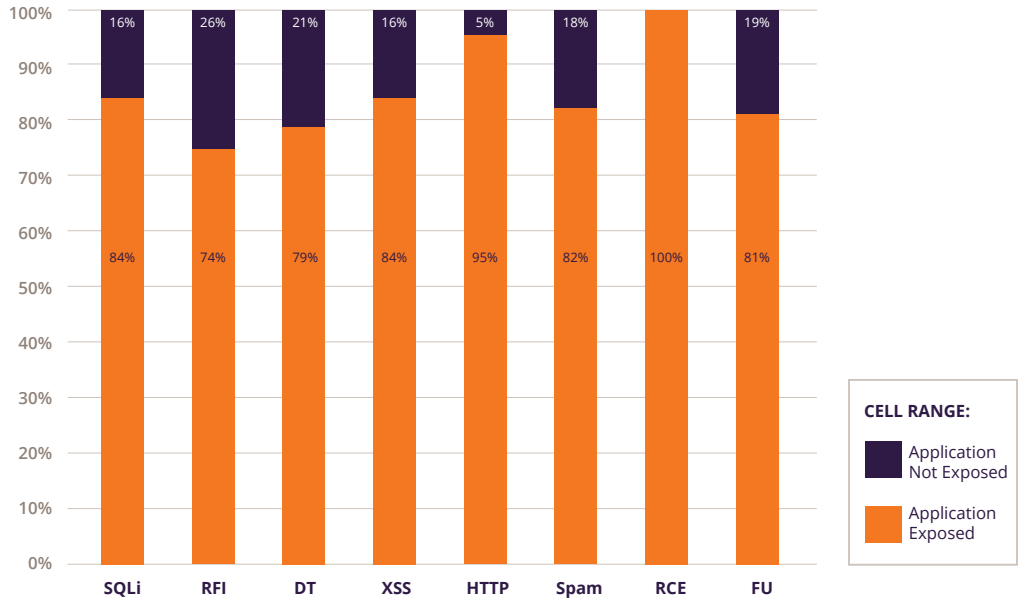


Figure 6: Web Application Malicious Traffic Exposure Ratio

Each of the attack types has at least 74% popularity, i.e., essentially at least three out of four applications were exposed to attack attempts of this type during the report period.

The most popular attack was Remote Code Execution, with attack attempts on every single application we examined. We attribute this fact at least partially to the Shellshock mega-trend, which is discussed in Section 8.1 below.

In Figure 7, we present a comparison of the popularity of attack for this year to the popularity of attacks from the previous year.[1] The diagram shows the likelihood of an application to get malicious traffic of a certain type along a period of six months. We omit RFI and Spam attacks from the diagram, since they were significantly changed in this report, and only mention that both show a decrease in popularity—in particular RFI attacks which had a cliff drop in 2015 (this trend is visible in the incident quantity analysis in Section 4.2 below).

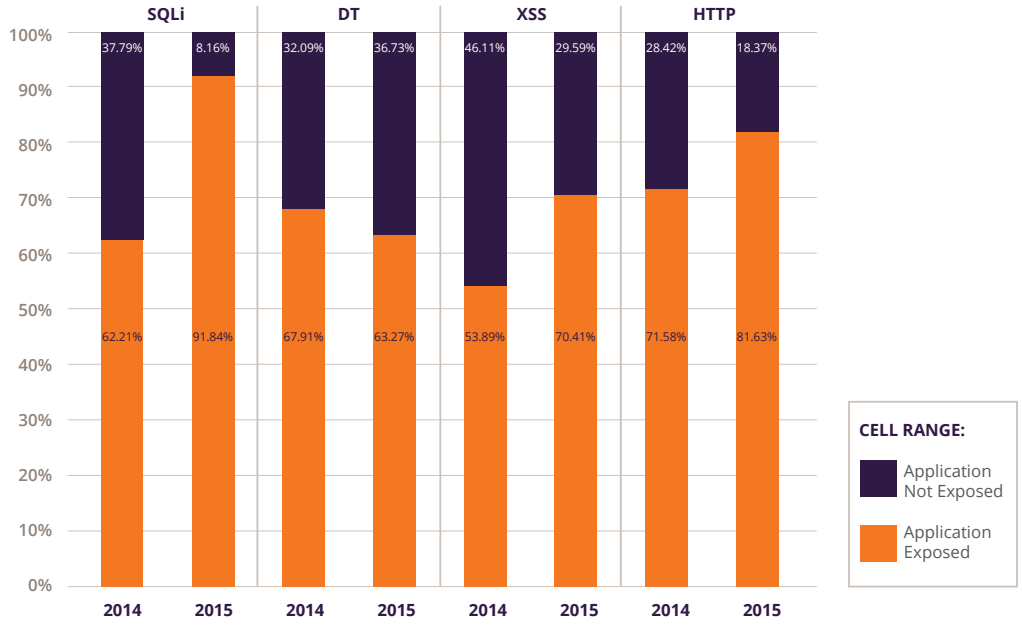


Figure 7: Web Application Malicious Traffic Exposure Ratio Between Years

Figure 7 shows different trends for different attack types. Several attack types became more popular, including SQLi, HTTP, and XSS attacks. The increase in XSS and SQLi attacks continues the trend we have seen in 2014. HTTP attacks, which in many cases can be attributed to reconnaissance attempts, also showed a small increase.

On the other hand, RFI, Spam, and Directory Traversal attacks were seen in fewer applications than the previous year.

Spam attacks are still widely used, but they are not spread among all applications. Instead, they focus on specific segments. This phenomenon is clearly seen in the analysis of attack trends on different industries in Section 6.1 and Content Management Systems in Section 6.2 below.

The decrease in RFI attacks can be attributed to the Human.txt mega-trend, which included massive RFI campaigns during the WAAR 5 period in 2014.[2]

We haven't seen an actual reason for the decrease in the number of Directory Traversal attacks. But we attribute this trend to the actual decrease in popularity of this attack.

[1] For the sake of comparison, we used the attack definitions from the previous report and normalized the numbers from last year to a six month period as in the current report.

[2] http://www.imperva.com/docs/HII_Web_Application_Attack_Report_Ed5.pdf

Humans.txt

Humans.txt is a static file Google keeps on its servers. In the last few years, we witnessed many RFI reconnaissance attempts that used this file as the included object. This mega-trend was very prominent throughout 2014, but faded towards the end of the year and was rarely seen in 2015.

4.2 Quantity of Attacks

In this section, we measure the number of attacks an application suffers for different attack types. In order to do that, we introduced the incident notion as explained in Section 3.2).

In Table 1 and Figure 8, we present the numbers of incidents per attack type during the report period. For each attack type, we show the average number of incidents, the median, the 25th and 75th percentiles, and finally the maximum and mean number of incidents we saw for a single application.

	SQLi	RFI	DT	XSS	HTTP	Spam	RCE	FU
1st Q	2	0	1	2	9	1	156	2
Median	13	5	6	12	34	24	273	23
3rd Q	55	25	20	38	109	276	591	84
Max	903	4,051	3,038	5,762	2,044	8,986	8,012	1,537
Mean	55	94	47	74	143	539	569	90

Table 1: Number of Attack Incidents

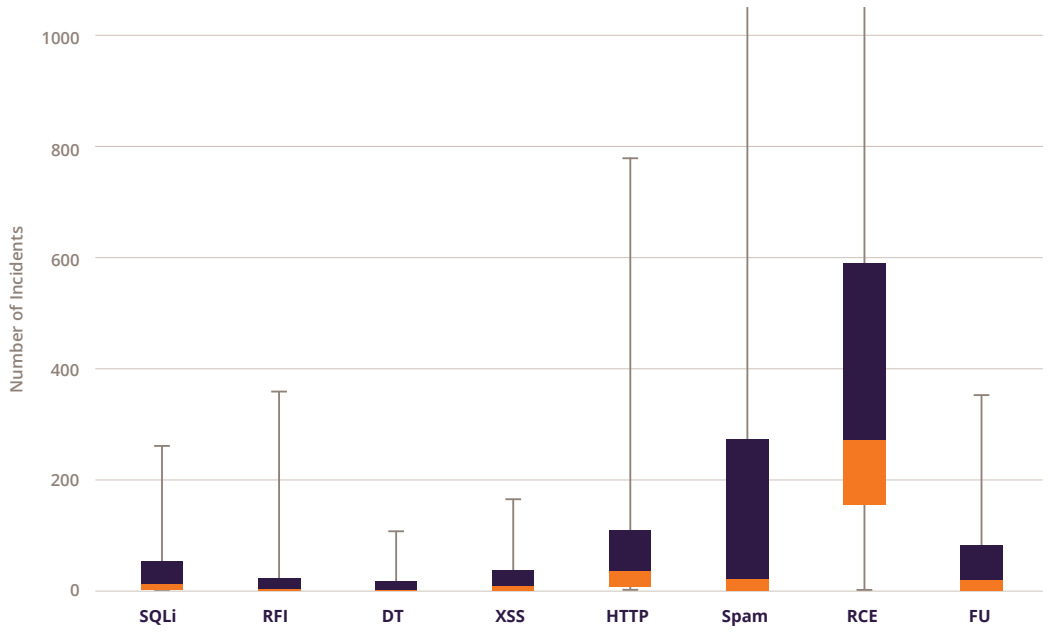


Figure 8: Number of Attack Incidents

The most intensively used attack was Remote Code Execution, which stands out in all measures. 75% of the applications had experienced 156 RCE incidents in the report period, which averages to about an incident a day.

We were surprised to discover more malicious File Upload attempts than XSS and SQLi attack incidents.

We compared the results to those of the previous years (Figure 9). The trends presented are identical to the ones presented in Section 4.1.

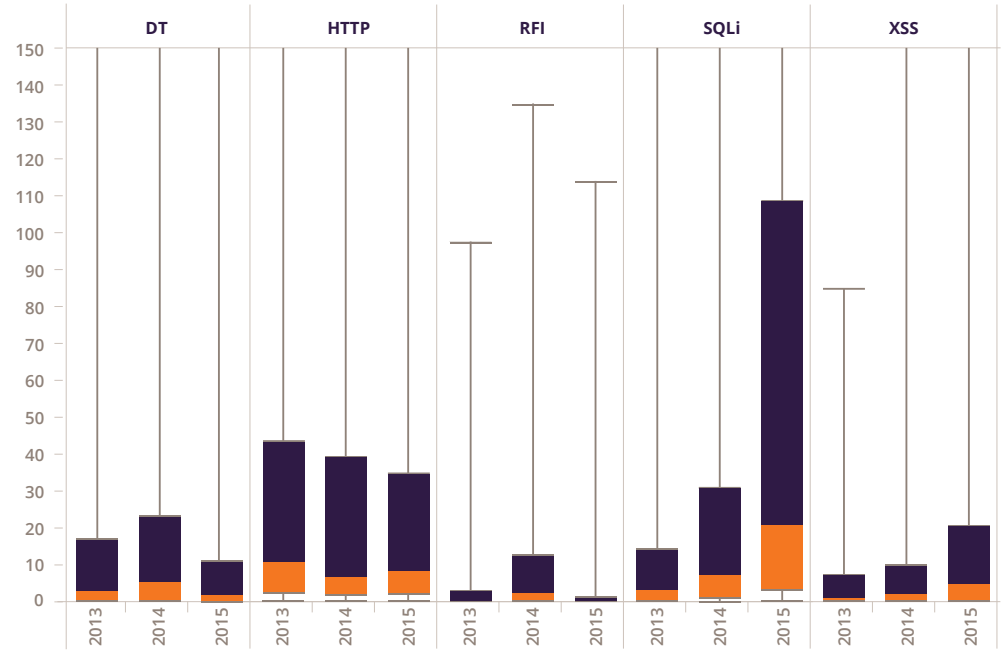


Figure 9: Comparison of Number of Incidents Between Years

The conclusions from Figure 9 vary according to the attack type. SQLi and XSS are gaining more popularity throughout the years as both distributions show a constant growth. RFI shows an obvious decrease, reflecting the humans.txt 2014 mega-trend. HTTP attack remains stable over the years, whereas Directory Traversal attacks show a decrease in popularity.

Next we analyze the disparity in the quantity of attacks on different applications by attacks type. We do that by observing the ratio between the number of incidents per attack type during the report period at the 75th percentile and the median.

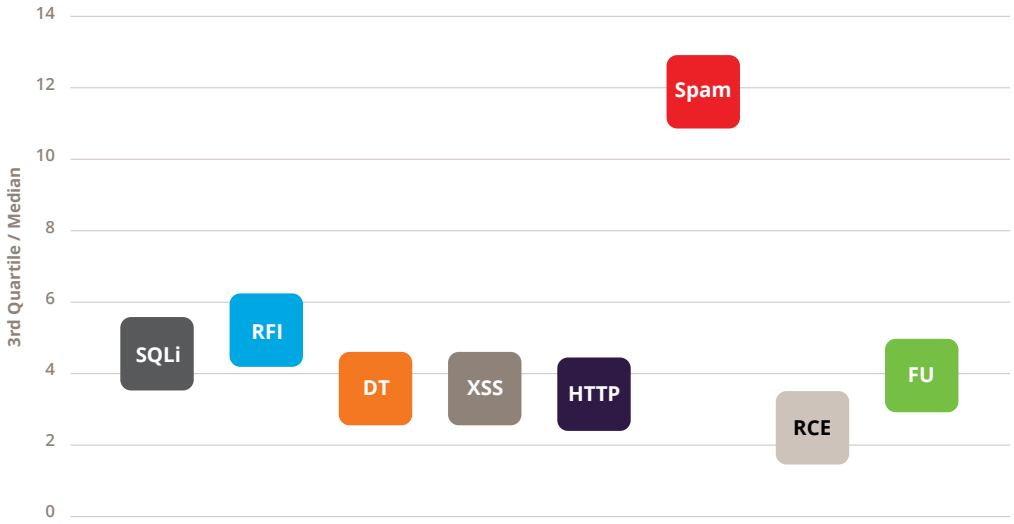


Figure 10: Disparity in Attack Quantity per Attack Type

One interesting observation is the "Fairness" of RCE attacks with a ratio of 2.2, significantly smaller than the other attacks (between 3.2 and 11.7). We attribute this phenomenon to blind Shellshock scanning campaigns that spanned most of the Internet during 2015.

On the other end of the scale, we find Spam attacks with a ratio of 11.7–far above the other attacks (max 4.9)–hinting of massive campaigns focused on relatively small number of applications. We attribute this phenomenon to the fact that Spamming is applicable only to applications of certain types, such as those allowing user content. This phenomenon is demonstrated clearly in Section 6 below, where we analyze vertical industries. The ratio of 4.9 for RFI attacks is attributed to the fact that RFI attacks are applicable only to PHP applications, and are rarely (and mistakenly) used against non-PHP applications (see Section 6.2 below).

4.3 Incident Intensity

Attack incidents, which are bursts of malicious HTTP requests, vary significantly in their ***Magnitude*** (number of requests), and their ***Duration***. In this Section, we measure the magnitude and duration of attacks for the different attack types.

4.3.1 Attack Magnitude

In Table 2 and Figure 11, we show the distribution of the magnitude of attack incidents per attack type.

	SQLI	RFI	DT	XSS	HTTP	SPAM	RCE	FU
1st Q	32	1	7	16	12	1	1	1
Median	72	1	12	29	19	1	1	1
3rd Q	204	4	34	85	50	3	6	4
Max	297,972	8,272	15,214	89,643	395,099	7,942	26,452	12,740
Mean	838	12	111	238	301	4	25	6

Table 2: Magnitude of Attack Incidents

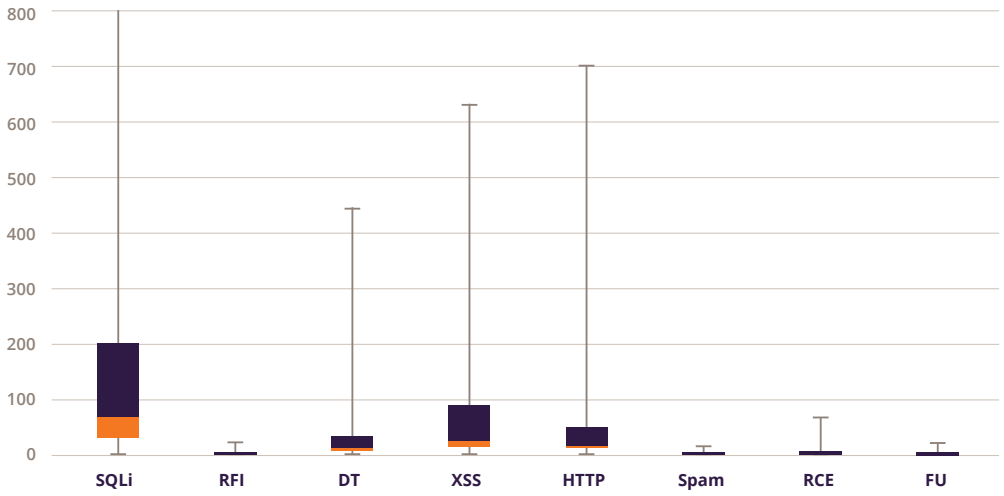


Figure 11: Magnitude of Attack Incidents

SQLi attacks, and to some extent XSS attacks, are characterized by massive attack incidents, with half of the SQLi incidents including 72 requests or more, and half of XSS incidents including 29 requests or more.

The 75th percentile row not only shows bursts of malicious traffic in Directory Traversal and HTTP attacks, but also shows that RFI, Spam, RCE, and File Upload are very rarely used with more than a few requests.

The Max row shows SQLi attack incident and HTTP attack incident of 400,000 and 300,000 requests correspondingly. The most intensive XSS attacks drag behind with 90,000 requests.

Directory Traversal is less popular than RFI, Spam, RCE, and FU attacks, but its incidents are more intensive. We attribute this phenomenon to the nature of the attack: File Upload is an attempt to load a corrupted file to the server, usually to a known destination, requiring a single request. On the other hand, in a Directory Traversal attack, the attacker tries various DT patterns in various URLs in the application.

In Figure 12, we compare these results with the previous year, and show that while the number of incidents have changed, the magnitude of an incident per attack type remain rather similar. The only exception to that trend is for RFI incidents, which was biased in 2014 due to the Humans.txt mega-trend—a trend that was characterized with massive RFI campaigns.

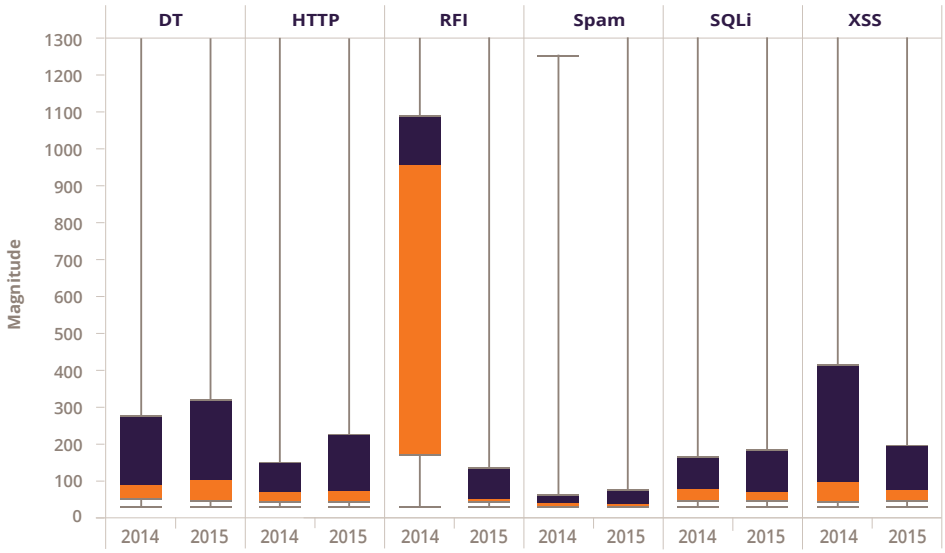


Figure 12: Comparison of Magnitude of Incidents Between Years

Figure 12 demonstrates the stability in the magnitude of incidents for DT, HTTP, Spam, and SQLi, which emphasizes the decline in the magnitude of RFI attacks. XSS attacks are also narrower this year—a phenomenon for which we found no obvious explanation.

4.3.2 Duration Analysis

In Table 3 below, we show the distribution of the duration of incidents in minutes per attack type[3].

	SQLI	RFI	DT	XSS	HTTP	SPAM	RCE	FU
1st Q	5	5	5	5	5	5	5	5
Median	5	5	5	5	5	5	5	5
3rd Q	10	5	10	10	10	10	5	5
Max	3,700	625	1,865	1,270	6,010	2,175	665	360
Mean	20	10	15	10	20	10	5	5

Table 3: Duration of Attack Incidents

[3] The duration is rounded to units of 5 minutes.

A vast majority of the attacks last less than 10 minutes. SQLi attacks, which stand out in their intensiveness when compared to other attack types, have a duration that resembles other attacks, such as XSS, HTTP, DT, and Spam. We attribute this to the nature of SQLi attacks, usually carried out using automated tools that scan the application for SQLi vulnerabilities in bursts, and characterized by high magnitude in short period of time.

The Max values in extreme cases of SQLi and HTTP attack incidents last longer than other attack types as expected from such high magnitude attacks. On the other side of the scale, File Upload longest incident is 2 to 15 times shorter than the longest incidents of the other attack types.

4.4 Battle Days

The statistical analysis of attack incidents may be significantly affected by a few campaigns that include consecutive sequences of attacks. In order to reduce the impact of such campaigns, we use the notion of battle days.

Table 4 and Figure 13 show the number of days in which a web application suffered attack incidents during the report period.

	SQLI	RFI	DT	XSS	HTTP	SPAM	RCE	FU
1st Q	2	0	1	2	7	1	78	2
Median	9	4	4	8	21	17	112	13
3rd Q	23	13	11	18	58	83	150	36
Max	155	181	181	181	177	181	181	177
Mean	19	17	10	17	39	48	111	25

Table 4: Battle Days

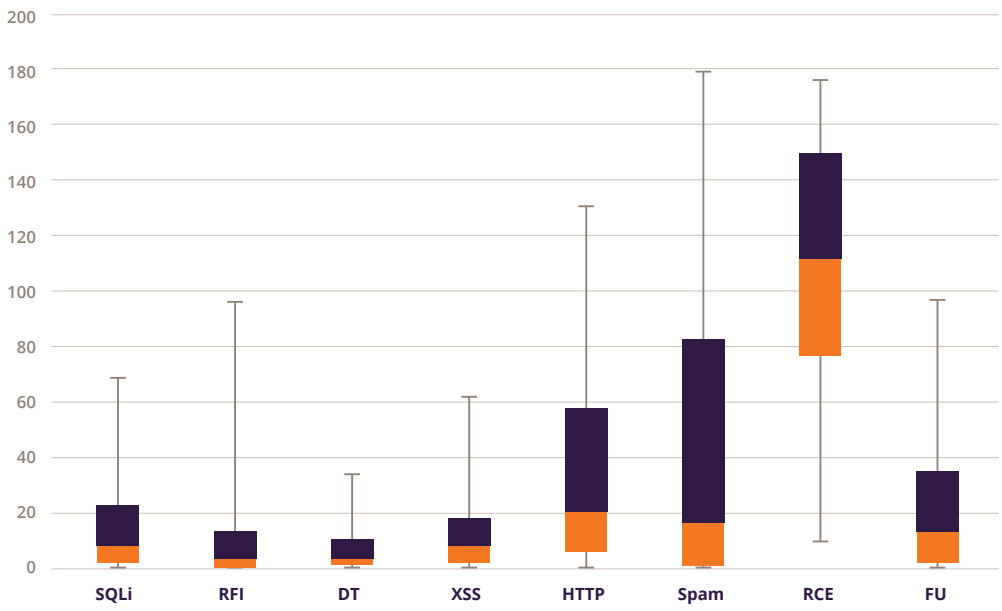


Figure 13: Battle Days

We find a strong correlation with the analysis of quantity of attack incidents from Section 4.2, with RCE attack being the most dominant attack. The results show that a typical application (represented by the median value) will suffer RCE attacks three out of every five days (112 days in half a year). 25% of the applications suffer Spam attacks almost every other day (83 days in half a year).

5. Reputation

Reputation-based security mechanisms are based on classification of events not only according to the content of the requests, but also according to other features such as the originating IP of these requests. We denote these below by detect-by-reputation, as opposed to detect-by-content requests. During the report period, 80,605,285 attack requests were detected-by-reputation on the report applications. We analyze several reputation-based mechanisms, relying on blacklisting of IPs which are essentially divided into two classes. ***Anonymous Browsing*** mechanisms identify Anonymous Proxies and TOR IPs, used by attackers to make tracking them harder. ***Serial Attackers*** mechanisms identify IPs from which significant malicious traffic was detected in recent history, and are often related to automated attackers. In Section 8.3 Scraping Attack from TOR Network below, we describe a case study of such attack.

In Figure 14, we present a week-by-week tracking of the number of IPs included in Anonymous Browsing and Serial Attackers lists (scale located on the right), and the number of requests arriving from these IPs (scale located on the left). Due to the variability in the amount of traffic from bad-reputation IPs, with intensive attack campaigns expressed in hundred times more traffic than normal times, we use logarithmic scale.

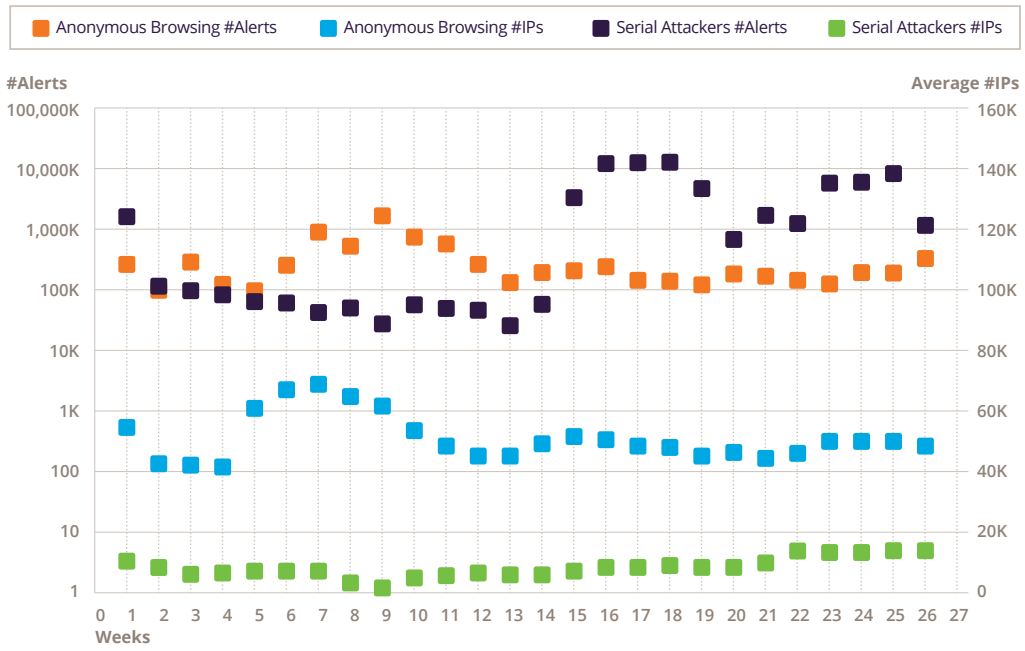


Figure 14: Reputation Over Time

Figure 14 shows that the applications under investigation had a significant increase in the number of Serial Attackers requests at the beginning on week 15, with two waves of 5–6 weeks each. The graph shows only a small increase in the number of Serial Attacker IPs, and thus the attacks just had more malicious traffic per source. The Anonymous Browsing analysis shows a significant wave during weeks 5–10, which is surprisingly accompanied with a decrease in the number of Anonymous Browsing IPs.

The numbers indicate an automated generation of web traffic to some of the applications through Anonymous Browsing mechanisms, which characterize attack campaigns.

The waves in Figure 15 represents weeks with large amount of millions of reputation alerts. This is divided between the reputation classes, with Anonymous Browsing leading with close to a million weekly alerts during weeks 5–14, and Serial Attackers taking the lead in weeks 15–26 with dozens of millions of weekly alerts. This split shows how the reputation mechanisms that detect malicious activity from different angles complete each other.

One would expect that having the attacker's IP in a reputation list, and getting blocked over and over will make the attacker stop the attack or switch IP. In Figure 15, we analyze the attackers that don't stop and keep using the same IP over and over for long periods despite getting blocked. We analyze this by tracking the number of weeks in which an IP remains in reputation lists.

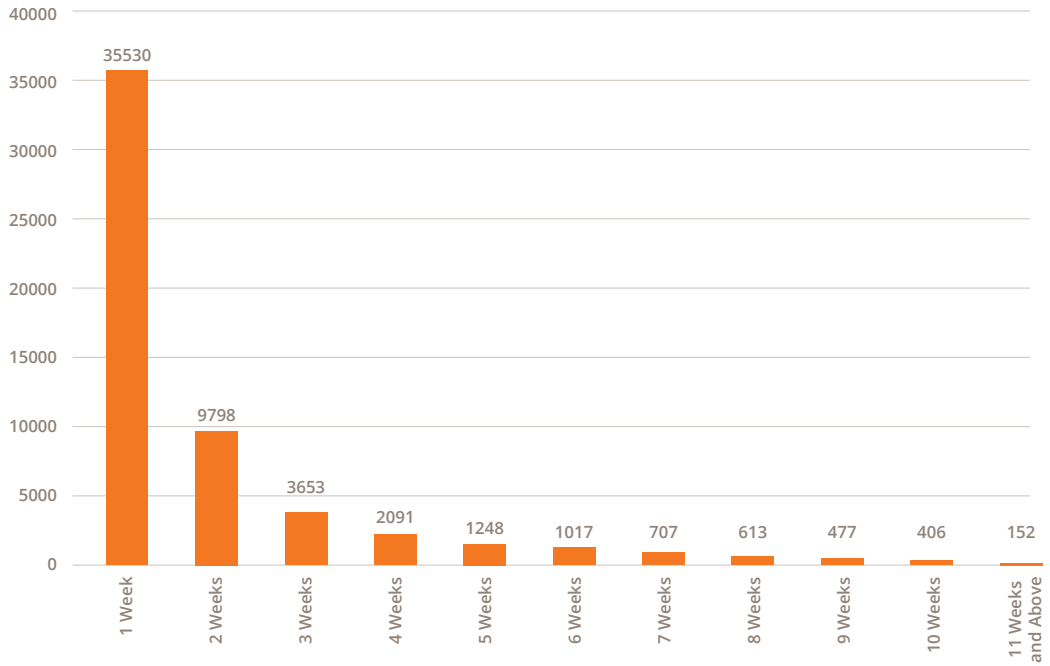


Figure 15: IPs Time Blocked by Reputation

As expected, most IPs were included in the reputation blacklists for a single week, after which their usage for malicious activity had stopped. However, 2,091 IPs continued generating malicious activity and remained in the reputation list for four weeks and 152 IPs were in the lists for 11 weeks and more.

In Figure 16, we quantify the contribution of detect-by-reputation and compare it to the detect-by-content mechanisms. The Figure presents the distribution of the number of request per category.

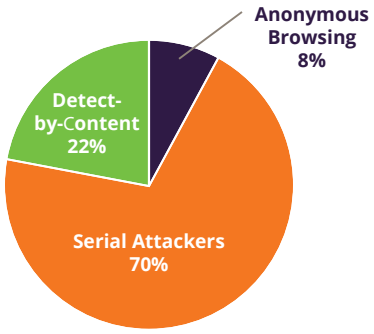


Figure 16: Contents Alerts Versus Reputation Alerts Distribution

We found that 78% of the malicious requests, almost 4 out of 5, were detected-by-reputation—most of them using the Serial Attackers mechanisms. This ratio demonstrates how detection of web scanning efforts obtained from collections of attack information from many applications can be effective in protecting applications. These numbers represent close to 100,000 malicious requests that reputation-based mechanisms detected for an average application on an average month—without the application or WAF investing computational resources in processing them. This statistic goes along with the increased attack intensity we saw in Section 4.3. More intensive attacks means that the IPs from which the attack originates are more likely to get into reputation blacklists and be assigned as Serial Attacker, and more requests from these sources being blocked due to detect-by-reputation mitigation.

6. Vertical Analysis

In this section, we examine the distribution of attacks on applications of various types. We divide the applications according to their Industry (Section 6.1) and Web Technology (Section 6.2).

6.1 Vertical Industries

We divide the applications into the following categories:

- **Economy** – business-related information, such as corporate overviews, business plans, etc., and services that help create such information.
- **Financial services** – financial information, and access to online financial accounts.
- **Computers** – computers related web sites, including SW and HW information, as well as internet and technology details.
- **Health** – health care information and services.
- **Shoppin**g – online shopping sites, with payment and delivery options.
- **Travel** – travel information and equipment.
- **Food** – information on restaurants, recipes, and worldwide dining options.
- **Leisure** – leisure information and products, including sports, entertainment, and recreation.
- **Other** – other web-applications that are not included in one of the above categories.

In Figure 17, to demonstrate the dominance of attack types, we present a 3D graph with the X-axis representing an attack type and the Y-axis representing the different industries. The third dimension is the median of the number of incidents per application, represented by the size of the bubbles. The figure is separated into two different scales because the Spam and RCE activity were significantly higher for all industries.

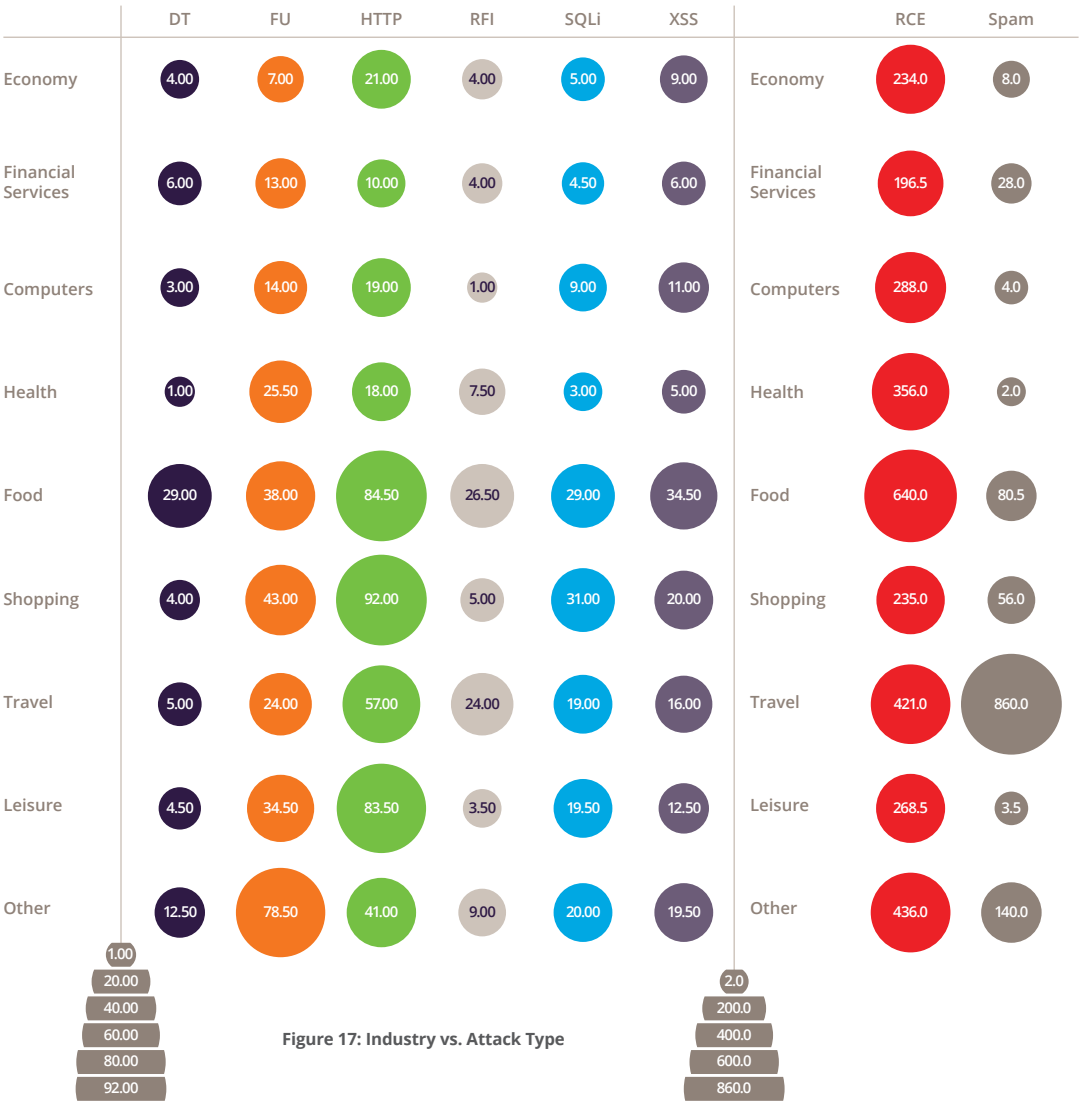


Figure 17: Industry vs. Attack Type

We draw several conclusions from observing how different attacks threaten different industries. The dominance of Spam attacks in Travel applications makes sense, given the nature of travel applications which rely heavily on user recommendations.

RCE is similar across all industries, mostly due to ShellShock scans, which are indifferent to industry type. HTTP attacks are dominant on Shopping, Restaurant, and General web-applications. This could stem, for example, from automatic tools attempting to scrape the items' content.

Figure 18 shows the proportion of incidents per attack type for each of the industries. The numbers within the diagram are the average number of incidents and the percentage of incidents of this attack type for applications associated with this industry.

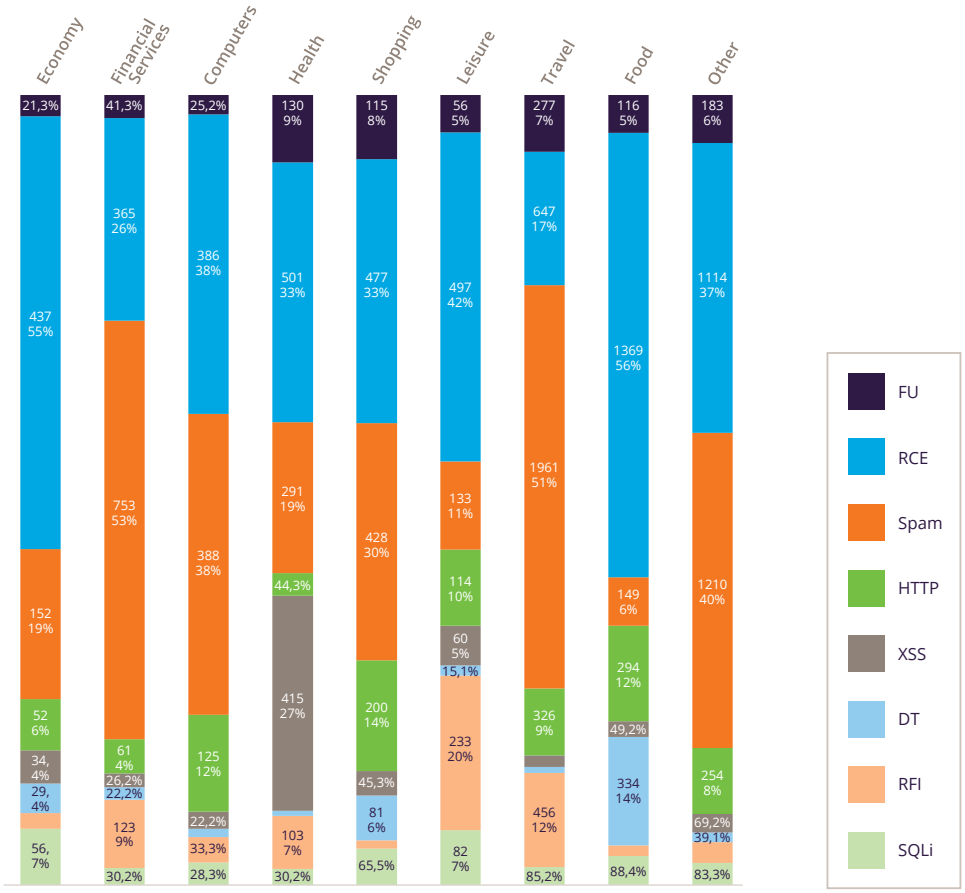


Figure 18: Alerts Proportions by Industry Verticals

Food applications had the highest proportion of RCE attacks: more than half of their incidents are RCE. We could not find a credible explanation for the attackers' preference to take control of food applications. Since there are also general motivations for RCE attacks, like abusing server computational power for harvesting bitcoin or using it as a member of a botnet, we suspect that the reason is based on speculation by the attackers that food applications may be easy targets, and less protected than financial or shopping applications. This conjecture is supported by the nature of RCE attacks we saw, with blind scans of all applications. Financial services and Travel underwent a lot of Spam attacks (more than half of their incidents are Spam), with 1,961 Spam attack on a typical Travel application.

Figure 19 shows similar analysis, without Spam and RCE attacks, emphasizing the trends in other less dominant attacks. The percentages within the diagram are from the total number of attack incidents that exclude Spam and RCE.

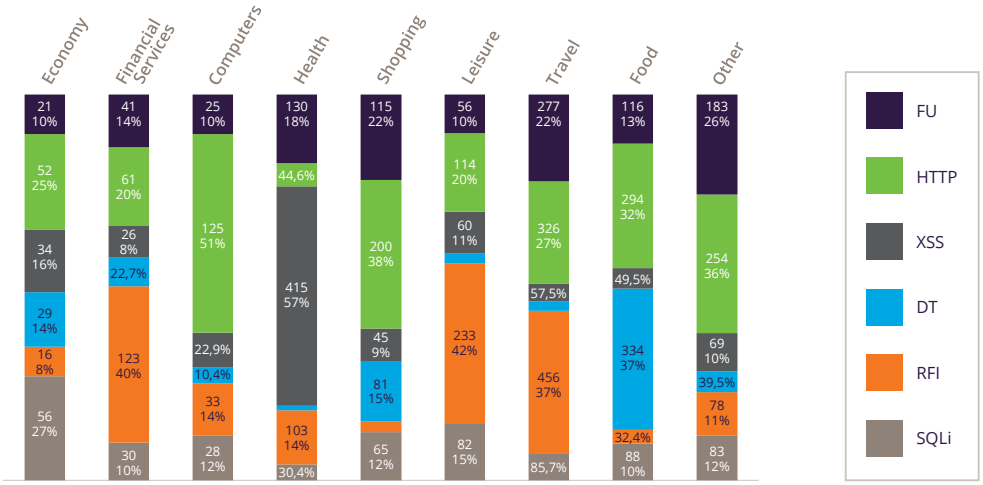


Figure 19: Alerts Proportion by Industry Verticals (Excluding Spam and RCE)

Health web applications suffer massively more XSS attacks—415 incidents which are 57% of the incidents—and significantly more than other industries for which only 5%–16% of the attack incidents were XSS. The Health incidents average is almost 10 times higher than the other industries. It is possible that attackers find Health application attractive for hijacking sessions and stealing Personal Identifiable Information (PII). On the other hand, Health applications rarely suffer from SQL Injection attacks.

Economy industry applications suffers almost twice the relative amount of SQL Injection attacks compared to all the other industries (27%). A potential explanation for this phenomenon is that data harvested from economy applications may be expected, a priori, to be more profitable and easier to monetize for the attacker when compared to other industries.

The Restaurant industry suffers more than twice the relative amount of Directory Traversal attacks (37%), and the highest average incidents per application (334).

Computers and Shopping industries have the highest rate of HTTP attacks (51% and 38% respectively), which might indicate the use of automated tools to browse their web-applications, e.g., for the sake of Content or Price Scraping.

The Travel and Shopping industries suffer a high number of File Upload attacks, with almost a quarter of their alerts being File Upload alerts. Their average per application is also high compared to the other industries (115 and 277 respectively). One possible explanation for this is that many Travel and Shopping applications have file upload functionality, used for their users to upload documents on the relevant subjects (traveling or fashion photos), and are abused by attackers for uploading malicious files.

6.2 WordPress and PHP

Next, we examine attack trends on Web Technology, in particular Content Management Systems (CMS) and Frameworks. We classified 55 applications from our sample of 198 applications as CMS-based, with 20 WordPress applications, 11 Drupal applications, and 24 applications based on 11 other CMS frameworks. Figure 20 shows the five most popular CMS[4], including WordPress and Drupal, which are also found in our applications[5].

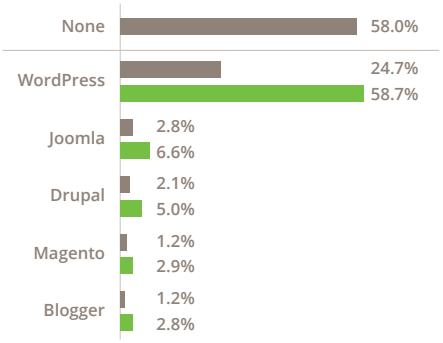


Figure 20: The Usage of CMS for Web-Applications

We then compared the attack incidents and their distribution. In Figure 21, we present the average number of incidents throughout the report period for WordPress, Other CMS (excluding WordPress), All CMS (including WordPress), and non-CMS applications.

Application Type	DT	FU	HTTP	RCE	RFI	SQLi	Spam	XSS	Total
WordPress	39	178	117	768	393	54	1893	55	3497
Other CMS	119	134	280	1031	74	83	833	239	2793
All CMS	90	150	221	935	190	72	1219	172	3049
Non CMS Applications	29	65	111	418	53	48	254	32	1010

Figure 21: Attack Incidents Average per Application for CMS Slices

CMS applications suffer on average three times more attack incidents than non-CMS applications, with 3,049 attack incidents in the report period, compared to only 1,010 incidents for non-CMS applications. This trend holds for essentially all attack types. WordPress applications suffer from even more attacks, with 3,497 attack incidents in six months—250% more than non-CMS applications.

The attraction of attackers to CMS applications and in particular to WordPress is not new. CMS frameworks have an open nature, with open developer communities that generate never-ending sequence of plug-ins and add-ons, with varying level of security. This situation has led to corresponding never-ending flow of CMS vulnerabilities, with WordPress as the leading CMS taking the lead also in the amount of published attacks. Furthermore, the fact that WordPress and other CMS applications resemble each other facilitates automated scanning attacks that work effectively on all applications of this type with only minimal adjustments.

[4] http://w3techs.com/technologies/overview/content_management/all/

[5] The gray percentage refers to the total examined web-applications. The green percentage refers to the CMS market. None means that 58% don't use the CMS monitored. WordPress, for example, is used by 24.7% of all the websites, which is a CMS market share of 58.7%.

We next examine how the attack incidents divide among the different attack types for CMS and Non-CMS applications. In Figure 22, we present the proportion of attack types in CMS and non-CMS applications.

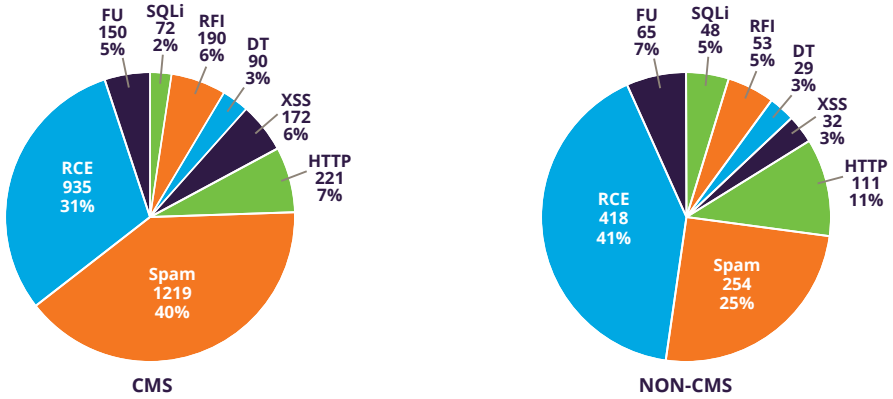


Figure 22: Attack Incidents Proportions CMS/Non CMS

The portion of Spam attacks in CMS applications is 40%–significantly more than the 25% of non-CMS ones. This is expected as CMS application are designed to be user-data driven, and as such are prone to Spam attacks. All the other attack types have smaller portions in CMS applications, but larger absolute numbers.

We compared WordPress applications to non-WordPress CMS applications. In order to focus on the comparison between these populations, we excluded Spam attacks, which are very popular in both populations.

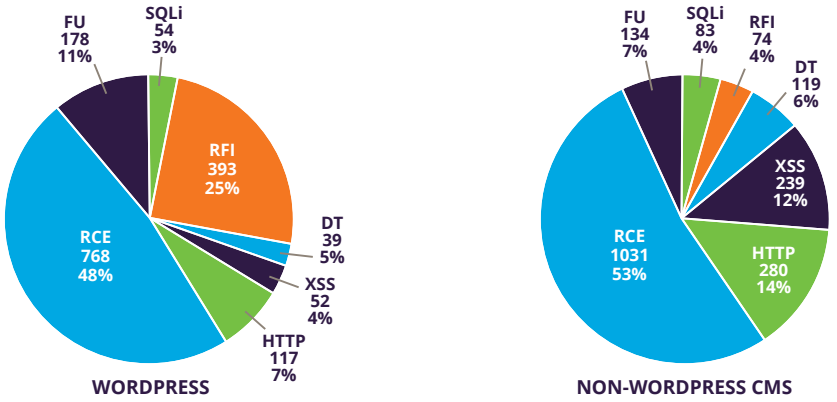


Figure 23: Attack Incidents Proportions for CMS Applications (WordPress/Non-WordPress)

The portion of RFI attacks in WordPress (25%) is significantly higher than the portion in all applications. Given that WordPress is written in PHP, the popularity of RFI attacks on WordPress is natural. However, many other CMS applications are written in PHP, and suffer fewer RFI attacks. A possible explanation to this phenomenon is attackers that don't target a specific application, but start with scanning the Internet for vulnerable applications. A Low Hanging Fruit approach–simple and effective–for detection of potential RFI targets, would be to run a WordPress test and mount an RFI attack in case of success.

We analyzed attack trends on PHP with non-PHP frameworks (Java, Ruby, CFML). Since most of the CMS applications in our data are written in PHP, we omitted them from this analysis, thus focusing on framework trends without being biased with CMS trends.

Figure 24 presents PHP vs non-PHP attack incidents proportions. For each attack type, the average attack incidents per web-application is presented as well.

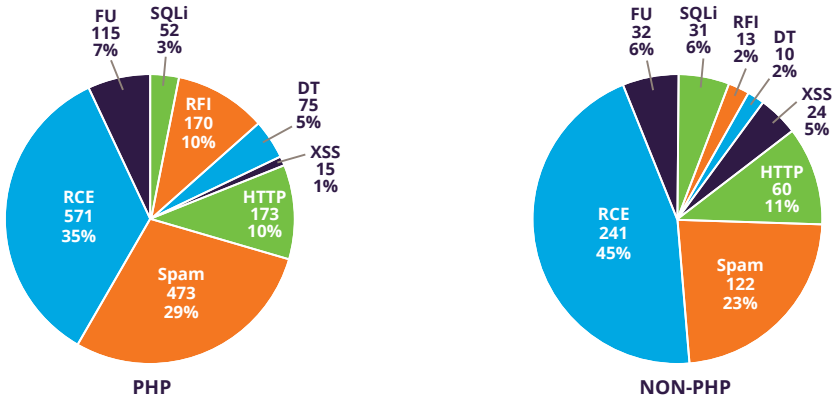


Figure 24: Attack Incidents Proportions PHP/Non-PHP

PHP applications suffered five times more RFI attacks than non-PHP ones. This trend is expected, given that RFI attacks are mainly applicable to PHP applications. While the total number of attacks on a typical PHP application is three times as large as the number for non-PHP applications, the attack types (except RFI and Spam) are distributed fairly similarly between both populations. Since there is a significant intersection between WordPress and PHP applications, WordPress trends of large total number of attacks, in particular Spam attacks, are expressed also in the PHP population.

7. Geographic Attack Trends

We analyzed the geographic distribution of the attack sources for different attack types. We examined both the number of requests (Section 7.1) and the number of attacking hosts (Section 7.2). In both cases, we examined the upper quartile of the number of HTTP requests for countries with populations over a million, and normalized this using an estimate of the per-country number of Internet users, as published by the World Bank[6].

7.1 Traffic Volume

We defined the 'malicious traffic' of a country as the number of malicious HTTP requests per internet user—the higher the ratio, the more malicious is the country's traffic.

In Figure 25, we present the number of malicious requests per internet user (aka, the normalized traffic) for each originating country in a chart that includes the top 10 countries (those having the largest amounts of malicious traffic for at least one attack type). The square size represents the normalized traffic. The bigger the square, the more malicious the traffic is. The number on the square is the absolute number of HTTP requests in thousands. The entire table is ordered by the total number of malicious requests (the right column).

Let's examine one example in order to clarify how to read this table. For SQLi, Moldova has the biggest square and thus has the highest normalized amount of malicious traffic. Yet, it had 129,000 SQLi requests—far less than the United States with 3.6 million SQLi requests. In addition, Moldova is part of the top 10 countries with DT and HTTP malicious traffic. When compared to other countries, it ranked as the second highest country with malicious traffic (see the Grand total column).

[6] Retrieved from http://data.worldbank.org/indicator/IT.NET.USER.P2

	Attack								
Country	DT	FU	HTTP	RCE	RFI	SPAM	SQLi	XSS	Grand Total
Netherlands	125		633	91		27	876	95	1,846
Moldova	18		16				129		163
Ukraine	148	8	382		21	294	442	58	1,353
United States	1,100	49	3,674	1,327	2,045	587	3,604	5,285	17,672
Denmark	28		52	30			75	32	216
United Kingdom		4	424	272				1,525	2,225
Armenia							31		31
Israel	24						92		116
Russian Federation		7				59	1,900		1,967
Macedonia							24		24
Hong Kong		2	37	40					79
Latvia	5	3			5			7	19
China			7,196			1,031			8,227
Singapore		1	1.2	0.4				17	39
Norway			25		4				29
Finland	13							15	28
Switzerland				36					36
Germany		3		274					277
Peru				47					47
Bulgaria				14					14
Czech Republic								23	23
Canada					68				68
Iraq						6			6
Australia					28				28
France		20				49			68
Austria					7				7
New Zealand					4				4
Kuwait					2				2
Philippines						26			26
Romania						4			4

Figure 25: Number of Requests per Country for Each Attack Type

The most dominant countries are the Netherlands, Moldova, Ukraine, and the United States. While countries like Armenia and Israel are the point of origin for a significant number of SQL Injection attacks, the United States, Ukraine, and the Netherlands are leading sources of attacks of all types.

Both the United States and the Netherlands were identified in the previous year as significant sources of malicious activity and still retain this title. Cyprus and Canada were prominent in WAAR 5, being among the top 10 countries for several attack types. This year, Cyprus moved out of the leadership spot, and Canada remained only in the RFI top 10. On the other hand, Ukraine, which was only in the Spam top 10 last year, was ranked high in seven different attack types.

Figure 26 shows the world map of malicious traffic per country for all attack types: DT, HTTP, RFI, Spam, SQLi, and XSS attacks.

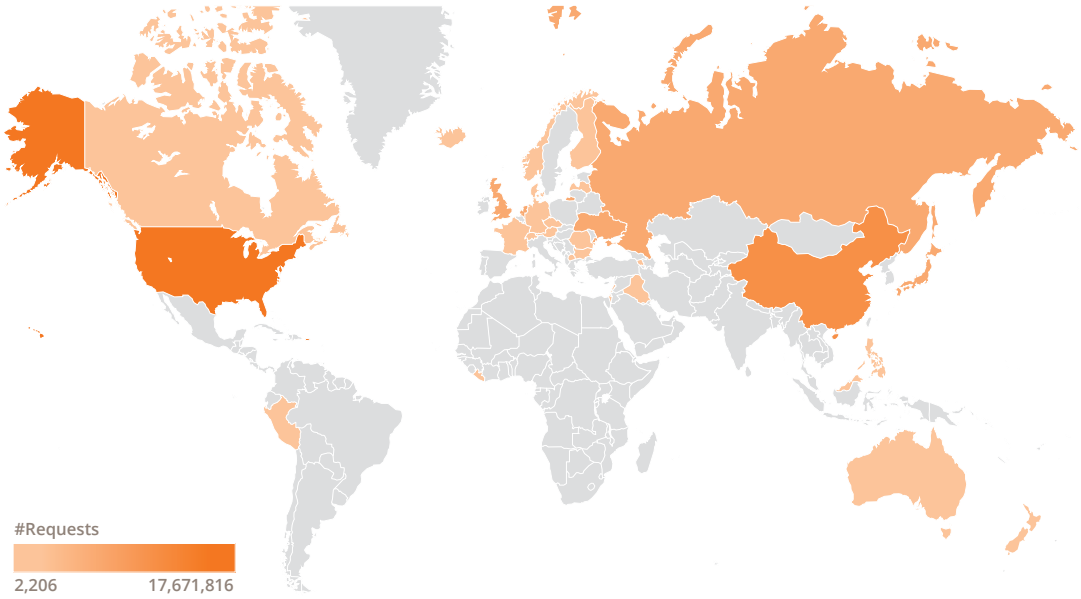


Figure 26: World Map Distribution of Malicious Countries for All Attack Types

The dominating country based on the sheer amount of requests is the United States. Since in this analysis we used absolute numbers of attacks without normalization to the country size, world giants like China (13th in the normalized diagram), the Russian Federation (9th), and United Kingdom (6th) take the lead.

7.2 Attacking Hosts

Next we analyze the geographic dispersion of attacks from a different perspective–according to the number of malicious hosts.

In Figure 27, we present the amount of malicious hosts per originating country in a chart, with normalization and absolute values in thousands, similar to those in Section 7.1.

	Attack								
Country	DT	FU	HTTP	RCE	RFI	SPAM	SQLi	XSS	Grand Total
United Kingdom		0.8	30.8	6.0	2.9		3.6	160.5	206.7
Lativa						0.1		2.2	2.3
United States	8.3	0.8	93.1	12.0	99.1	55.2	40.6	77.3	366.4
China			454.5						454.5
Ireland				0.6	0.4			1.3	2.4
Hong Kong		0.0	1.8	0.0				0.7	2.7
Canada			6.0		4.8	0.1	2.4		14.4
Australia			1.7	0.3	1.5		1.4	2.4	7.2
Colombia							4.7	4.4	9.1
Belarus	1.8								1.8
Lithuania						0.1	0.2	0.4	0.6
Venezuela						0.7		3.0	3.7
Ukraine	3.9	0.0							3.9
Netherlands		0.0	1.2	0.4		0.5	1.0		3.1
Singapore			0.4	0.1	0.3				0.8
Mexico			4.6		3.2				7.9
Kazakhstan	1.2								1.2
Bahrain							0.2		0.2
Norway					0.2		0.3		0.5
Moldova	0.1								0.1
El Salvador				0.2					0.2
Estonia						0.1			0.1
Puerto Rico					0.2				0.2
Romania						0.6			0.6
New Zealand					0.2				0.2
Russian Federation	4.6								4.6
Guatemala							0.2		0.2
Sweden						0.4			0.4
Thailand	0.8	0.0							0.9
Israel	0.2								0.2
Italy	1.2	0.0							1.2
Switzerland				0.1					0.1
Denmark		0.0							0.0
France		0.1							0.1
Germany		0.1							0.1

Figure 27: Number of Hosts per Country for Each Attack Type

The dominant countries are the United Kingdom, Latvia, the United States, and China. Observing the differences from the Request-based analysis, we see that attacks from China and United Kingdom are distributed among more IPs than attacks from other countries. Canada appears as part of the top 10 countries in half of the attack types, less dominant than in the previous year where it appeared in all the attack types examined. The United Kingdom maintains the lead in XSS attacks for both WAARs (first ranking in both reports). Meanwhile, Belarus stays ahead in DT attacks (second and first rankings for the previous and current year, respectively).

7.3 Hosting Services as Attack Sources

Here, we analyze three different Infrastructure as a Service (IaaS) vendors: Amazon Web Services, Google, and Microsoft Azure–which together hold most of the hosting services market.

We found that the portion of malicious traffic originating from these hosting services was 5%, i.e., one out of twenty attacks came from a hosting service.

Table 5 and Figure 28 below show the number of hosts that sent malicious traffic for each vendor (blue circles), and the average number of requests per host (red bars).

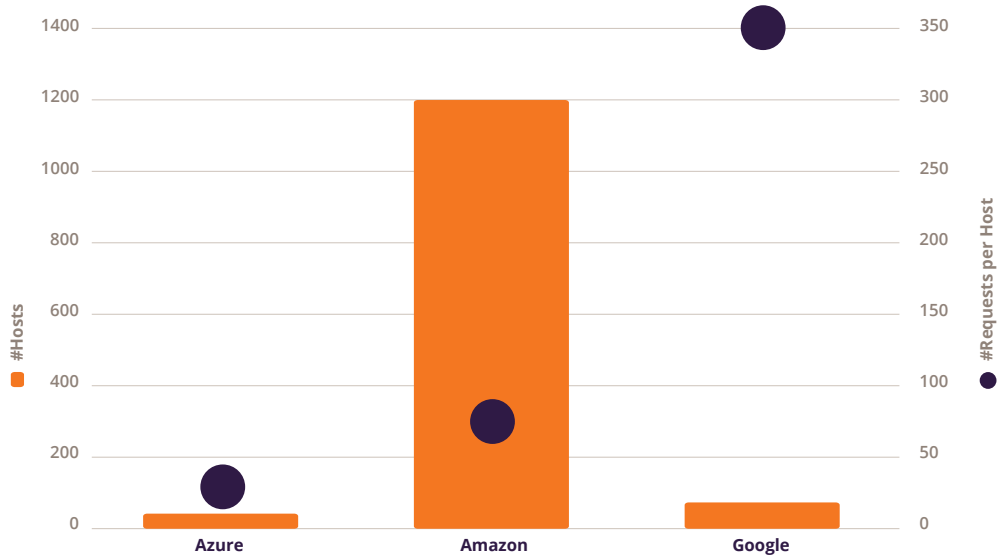


Figure 28: Number of Hosts and Number of Requests per Host

HOSTING SERVICE	#SOURCE IPS	#HTTP REQUESTS	#REQUESTS PER HOST
Azure	49	1,467	30
Amazon	1,210	91,361	76
Google	75	22,688	303

Table 5: Hosting Services Data

The malicious traffic originated in AWS divides between 1,200 AWS IPs, each generating an average number of 75 requests. Thus, 80% of the malicious traffic originated in IaaS vendors comes from AWS IPs. As opposed to that, we detected malicious traffic from only 75 Google hosting services IPs, with each of these being responsible on average for 300 malicious requests, four times the number of AWS IPs.

In general, IPs from IaaS vendors generated an average number of 90 malicious requests, about 10 times the number for non-IaaS IPs. This phenomenon can be explained by attackers investing in the effort to move to the cloud only for large campaigns.

8. Case Studies

In this section, we present several selected case studies representing some of the more intensive attacks we observed during the report period.

8.1 Shellshock Mega-Trend

The discovery of Shellshock[7] was one of the most significant application security events in 2014. The attack, published in September 2014, is a Remote Code Execution attack, allowing an attacker to take over servers that use the Unix Bash shell. In our analysis, we saw multiple attempts to exploit the Shellshock vulnerability. The Shellshock mega-trend expresses in the perfect 100% coverage of RCE attacks in the web application participation analysis (Section 4.1) and the large number of RCE attacks (Section 4.2). Shellshock attacks were detected in all applications in very similar numbers, indicating wide-scale blind scanning of the Internet with Shellshock attacks. Finally, when examining different vertical industries (Section 6.1), we saw that ShellShock scans were aimed at everyone without discrimination.

Figure 29 shows the number of Shellshock incidents per week (the blue bars with the left Y-axis), and the number of web-applications that were targeted per week (the orange markings with the right Y-axis).

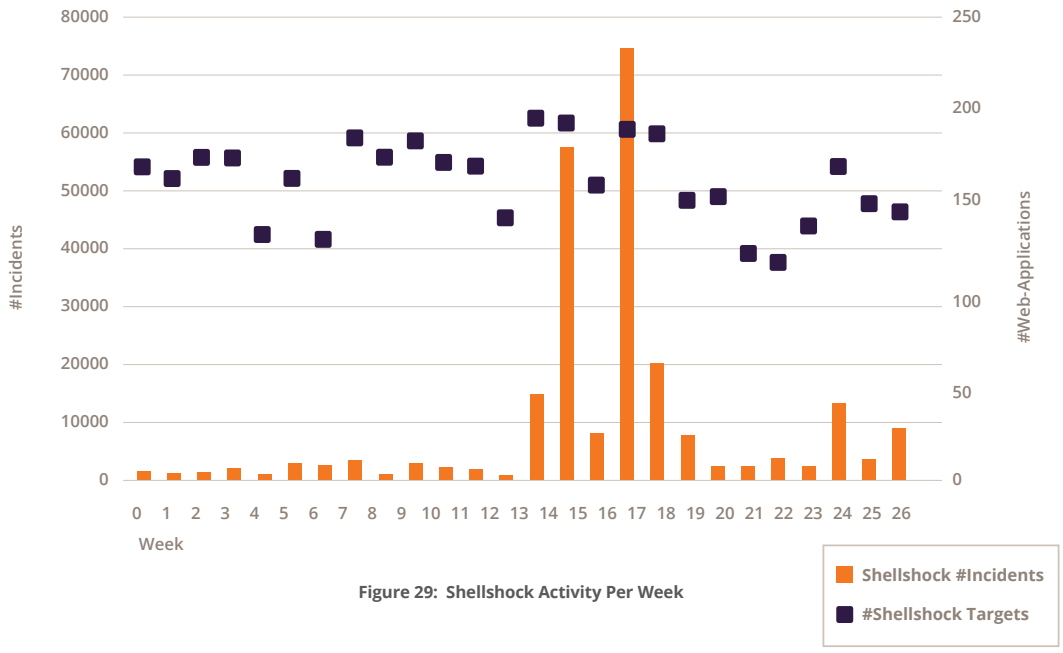


Figure 29: Shellshock Activity Per Week

Exploring Shellshock attacks over time, we observed two dominant waves: the first occurred during September 2014 (this time was not included in the current report) following the discovery of the vulnerability. The second was in April 2015 (weeks 14–18 in Figure 29). The first wave, not included in this report, indicates the quick turnaround of vulnerability-to-attack, with attack vectors integrated into exploit kits and used in the wild by attackers within hours. The second wave, seven months after the publication of Shellshock, showed a wide and intensive campaign persistently attacking most of the applications in our research. During the campaign period, most of the applications were exposed to thousands of Shellshock attempts every single week.

[7] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271

8.2 SQL Injection

We started the analysis in bird's-eye view of the attack traffic on one of the applications participating in the research. This view showed several periods with abnormal attack activity—from which we chose to focus on significantly large SQLi attack during week 23, and significant amount of traffic from anonymous sources in week 9.

Figure 30 shows the distribution of attacks per week, with different attack types represented by different colors.

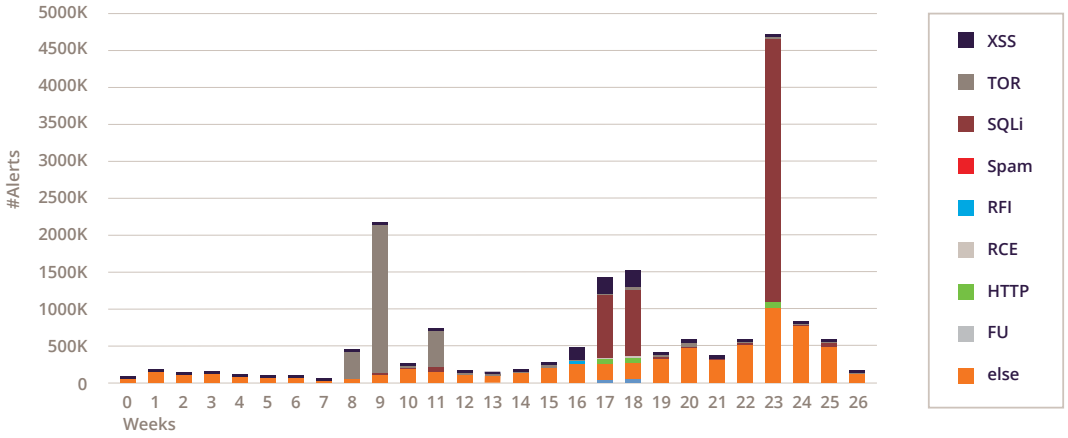


Figure 30: Distribution of Attacks Over Time

In week 23, nearly 3.6 million attacks were registered with high proportion of SQLi attacks (98%).

In Figure 31, we zoom in on a part of this week, and track the amount of SQLi attacks.

Following the detection of malicious activity from IPs participating in this attack campaign, they were assigned as Serial Attacker IPs and further traffic from them was immediately blocked-by-reputation (June 7 at 1 p.m.). Two days later, the attack faded out, possibly due to its ineffectiveness with malicious traffic getting blocked. After a couple of hours, another wave of this attack arrived from a new pool of IPs, with requests very similar to those of the previous wave, and the flow recurred. First, the attack requests were blocked-by-content, and within hours, the IPs were classified as Serial Attackers. The next requests were blocked-by-reputation until the attack fading out the following day.

It is interesting to note that a vast majority of the malicious traffic, around 90%, was blocked-by-reputation, and the remaining 10% was blocked-by-content using signatures, positive security model deviations, and other content analysis mechanisms.

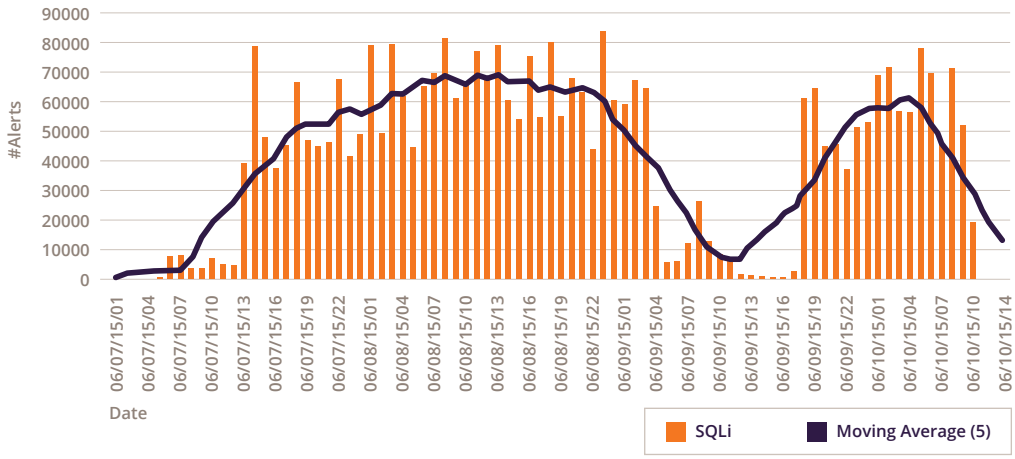


Figure 31: Distribution of SQLi Attacks During Week 23

The attack campaign lasted three days with an average number of 6,800 SQLi attacks per hour. This clearly shows an organized attack campaign using automated tools.

A second reinforcement to our automated tools theory is a relatively uniform distribution of URLs over the entire web-application.

8.3 Scraping Attack from TOR Network

We continued analyzing the attacks on the above application, this time focusing on the activity from TOR network in week 9. Analysis of the URLs to which the attack was targeted showed that 99% of the requests were targeted to only three URLs—one of which was the search page and the other two shopping pages (with parameters such as product id and category id).

We found that 777 TOR IPs were used for the attack, with each producing an average of 2,563 requests. Further investigation of the traffic showed that 1,086 session identifiers (jsessionid) were used during the course of the attack, each being used 1,117 times on average, and that sessions were extensively recycled between different IPs. For example, Figure 32 shows the usage of a specific session ID from as many as 25 different IPs.

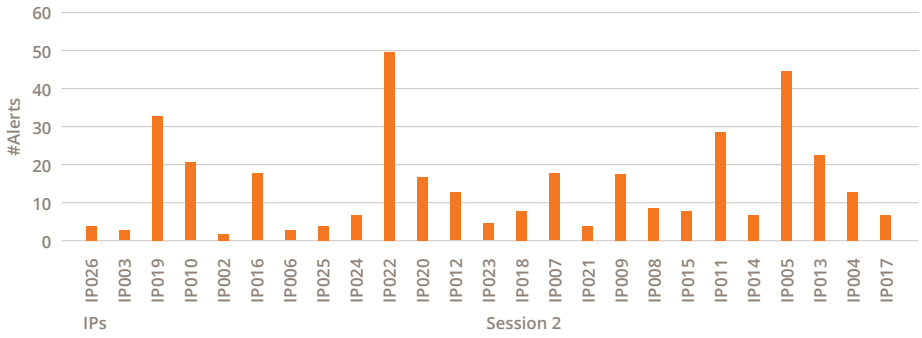


Figure 32: The Distribution of Alerts Among IPs for a Specific Session

Another indicator of the distribution of this attack was the concurrent usage of multiple sessions. For example, on 03/02/2015, 04:10, we could see seven active sessions where each session used over 1-4 different IPs.

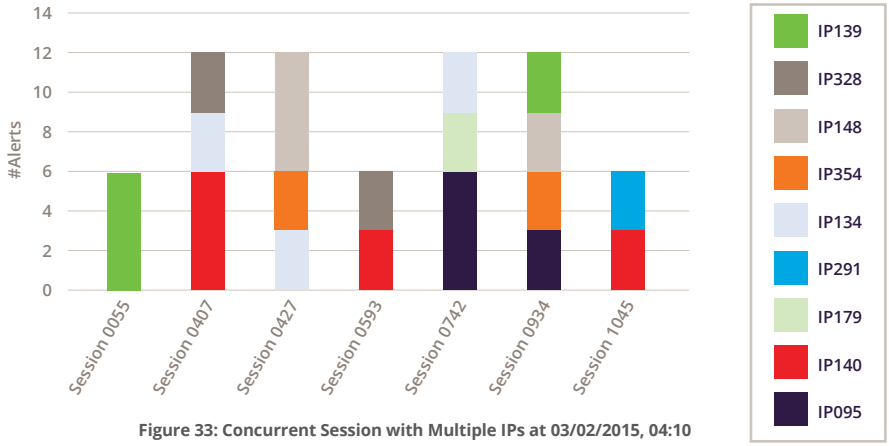


Figure 33: Concurrent Session with Multiple IPs at 03/02/2015, 04:10

The attack spanned over 453,547 unique user agents, indicating random User-Agents. However, we were able to identify three main user agents that were used interchangeably throughout the same session with different permutations or versions:

- MSN-BOT User-Agent: msnbot-Products/1.0 (+http://search.msn.com/msnbot.htm)
- Mozilla User-Agent:: Mozilla/5.0 (Windows; Windows NT 5.1; en-US; +MrTiger) Firefox/10.0
- Opera User-Agent:: Opera/9.80 (Windows NT 6.1; U; es-ES) Presto/2.9.181 Version/12.00 x:0)

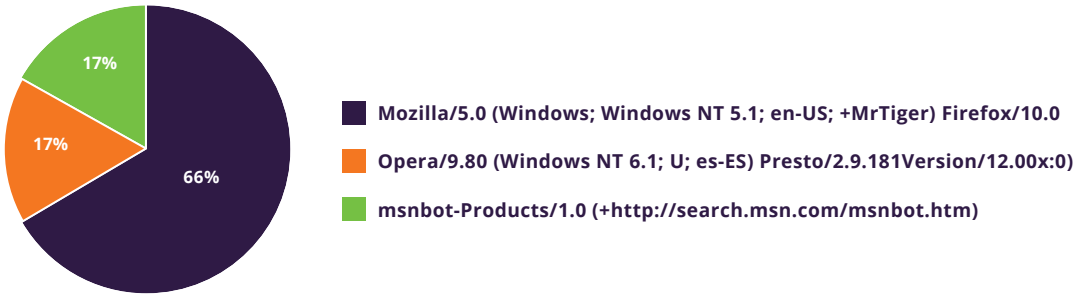


Figure 34: User-Agents Distribution

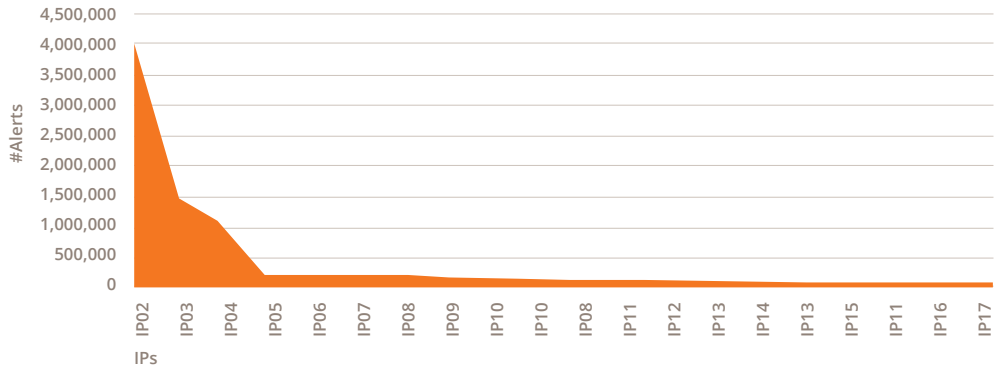


Figure 35: Distribution of Attackers Source IPs Per Attacks

Finally, we analyzed the IP distribution of the attack. Similar to the User-Agent case, we identified three dominant source IPs that were responsible for most of the attacks. All three IPs are registered to hosting services in the United States, which accommodate the TOR exit nodes through which the attack was launched.

Thus, the attack had several characteristics one would expect from an organized Scraping campaign—anonymity, distribution, recycling of session identifiers, multiple User-Agent strings, and random gaps between requests.

8.4 Cross-Site Scripting Attack

Again we started the analysis with a bird's-eye view of the attack traffic on an application, this time finding an unusual surge in Cross-Site Scripting attacks.

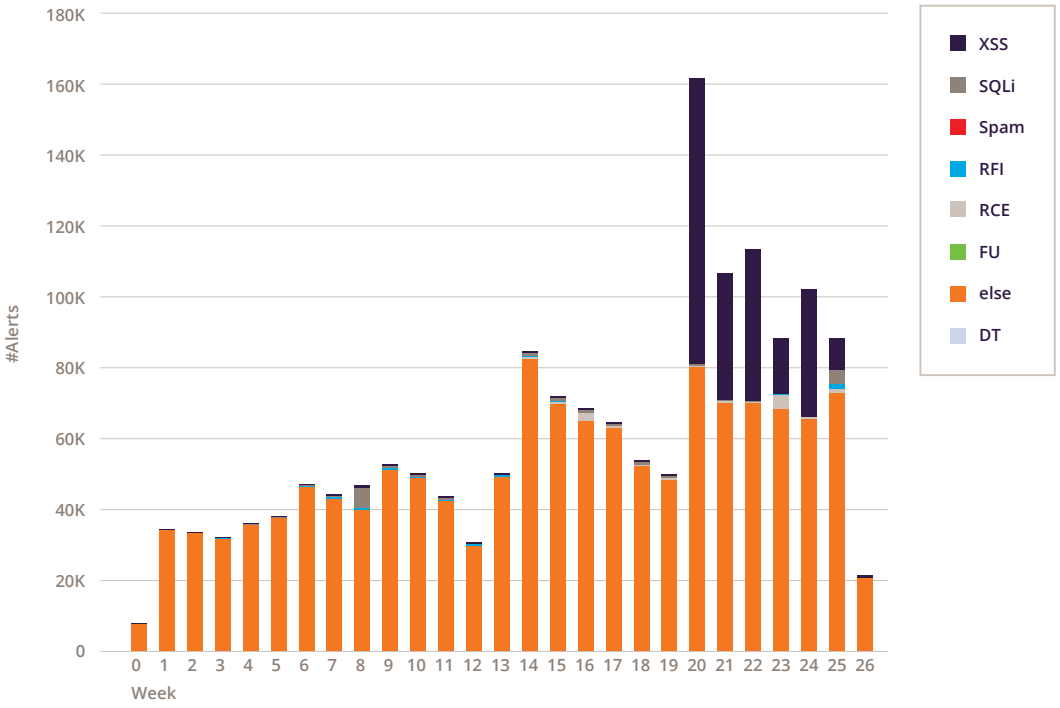


Figure 36: Distribution of Attacks Over Time in Weeks

Figure 36 shows a relatively uniform distribution of attacks with an exception of six weeks (weeks 20–25) with high volume of attacks in week 20 (1.6 million attacks), decreasing in the following four weeks, and then getting back to "normal" in week 26. The main contributor to this surge was a Cross-Site-Scripting campaign with 800,000 requests in week 20 and about half of that in each of the following four weeks. The week-by-week distribution of XSS attacks is shown in Figure 37.

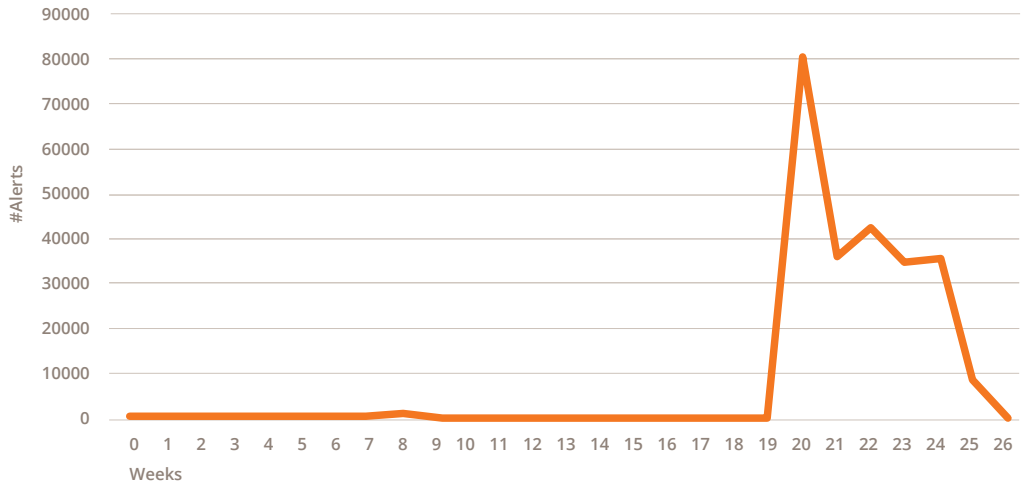


Figure 37: Distribution of XSS Attacks Over Time

We then analyzed the IP distribution of the XSS campaign (Figure 38), and found that two IPs were accountable for 99% of the XSS attacks we discovered. Further examination of these IPs revealed that they are related to a security vendor that offers vulnerability scanning services, which led us to the conclusion that these attacks were initiated from an automatic scanner tool—either by the application owner or by an attacker that chose to out-source his reconnaissance for vulnerabilities in the target application.

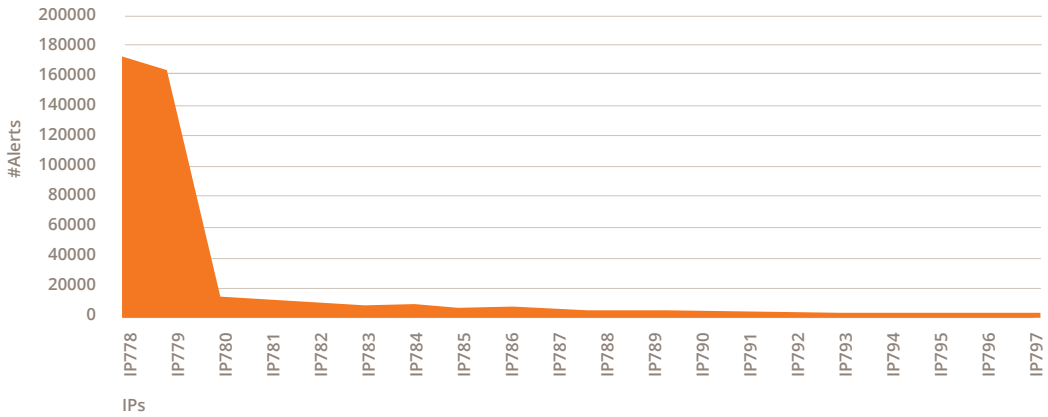


Figure 38: Top 20 XSS Attackers IPs

The contents of the attack requests show they were initiated from the Acunetix scanner. Examination of the attack timing led us to another indication for automated attacks, with the attack following clear time patterns—always on a Wednesday and at specific hours (9 a.m. to 3 p.m.).

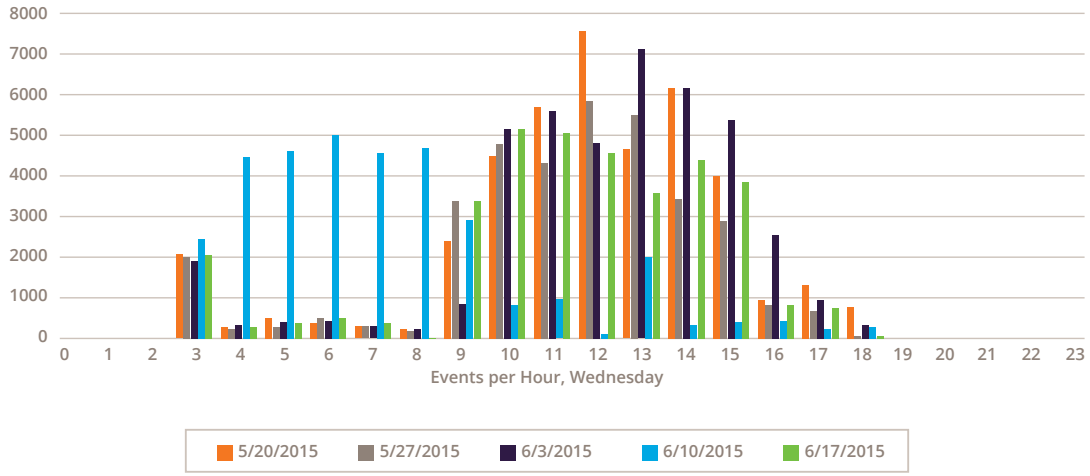


Figure 39: Distribution of XSS Attacks Per Hour on Wednesdays

We ended the analysis of this case study by examining the URLs the attack targeted. We found out that the campaign showed clear preference towards registration pages, with 60% of the attacks directed to registration pages within the application. These pages were only 4.5% from the 6,765 URLs we saw in the attack. We found another indication for the Acunetix scanner with the following URL appearing in some requests as the script source http://testasp.vulnweb.com/t/xss.js.

9. Conclusions and Recommendations

As noted in various sections of this report, attack volumes, sophistication, and the number of applications being exposed are growing. As attackers become more efficient at launching targeted high volume attacks across a wide range of applications, it has become imperative for businesses to educate themselves on the threats at hand—and take steps to thwart/prevent and mitigate them.

To mitigate the attacks and threats mentioned in this report, organizations should consider the following actions:

- **Detect and block attacks that target known vulnerabilities.** The knowledge base of exploitable weaknesses in an application must be frequently updated. There was a high number of automated RCE attacks across all applications after the vulnerability was made public. As patches cannot be applied at the same speed vulnerabilities are being published, organizations must resort to virtual patching for protecting applications in real time.
- **Deploy security solutions that mitigate automated attacks.** To do so, security solutions should recognize known automated sources; differentiate between bots and human clients; as well as detect unusual activity, such as an extremely high rate of web requests from a single user. Automated attacks must be identified as early as possible during an attack incident.
- **Learn from peers.** Applications in similar business verticals may share similar attack characteristics. In this report, we have shown that health applications suffer more than other verticals. Applications with private consumer information seem to suffer more, and there are certain attack vectors such as SQLi & XSS that are growing fast.
- **Consider multiple layers of security to protect data.** It has become evident that end point protection alone cannot mitigate unauthorized data access. Enterprises should deploy controls such as Database Activity Monitoring (DAM) and File Activity Monitoring (FAM) around their business data resources, and identify abnormal and abusive access of data.
- **Participate in a security community and share threat intelligence.** The increased automation and scale of attacks leave a large footprint which can only be seen by analyzing data gathered from a large set of potential victims.
- **Attack distribution is burst-oriented and far from consistently distributed.** Estimations for security measures should be based on the worst case scenario, not on the average case.
- **Security procedures and solutions should be as automated as possible.** Attack volume and turnaround time from vulnerability to attack is too overwhelming for humans to monitor, and in most cases, there will be no advance warning of an attack.

10. Web Attacks

10.1 SQL Injection

SQL Injection (SQLi) is an attack that exploits a security vulnerability occurring in the database layer of an application (such as queries). Using SQL injection, the attacker can extract or manipulate the web application's data. The attack is viable when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed, and thereby unexpectedly executed.

10.2 Remote File Inclusion

Remote File Inclusion (RFI) is an attack that allows an attacker to include a remote file, usually through a script, on the web server. This attack can lead to data theft or manipulation, malicious code execution on the web server, or malicious code execution on the application client side—such as JavaScript execution—which can lead to other attacks. This vulnerability occurs due to the use of user-supplied input without proper validation.

10.3 Directory Traversal

Directory Traversal (DT) is an attack that orders an application to access a file that is not intended to be accessible and expose its content to the attacker. The attack exploits insufficient security validation or insufficient sanitization of user-supplied input file names. Characters representing "traverse to parent directory" are passed through to the file APIs.

10.4 Cross-Site Scripting

Cross-Site Scripting (XSS) is an attack that lets the attacker execute scripts in a victim's browser to hijack user sessions and steal his credentials, deface web sites, insert hostile content, redirect users, hijack the user's browser using malware, etc. XSS flaws occur when an application includes user-supplied data in a page sent to the browser without properly validating or escaping that content.

10.5 Comment Spamming

Comment spamming (Spam) is a way to manipulate the ranking of the spammer's web site within search results returned by popular search engines. A high ranking increases the number of potential visitors and paying customers of this site. The attack targets web applications that allow visitors to submit content that contains hyperlinks. The attacker automatically posts random comments or promotions of commercial services to publicly accessible online forums that contain links to the promoted site.

10.6 Remote Command Execution

Remote Command Execution (RCE) is an attack that allows the attacker to execute operating system commands in a system shell. The attack exploits applications that suffer from insufficient input validation in conjunction with passing this input to a system shell. The attacker's payload is executed with the same privileges of the vulnerable application and can lead to full compromise of the server.

10.7 File Upload

File upload (FU) is an attack that allows the attacker to upload unauthorized file to the server. The attack exploits applications that suffer from insufficient input validation. In cases where the application stores the uploaded file in a public accessed location, an attacker can upload file containing code that will be executed once accessed or use the vulnerable application to store infected files that will be downloaded by victims' machines. In other cases, an attacker can overwrite critical files or waste the server's disk space by uploading extremely big files. This attack may result in defacement, XSS, Phishing attacks, DoS, or full compromise of the server.

The Imperva Application Defense Center (ADC)

The Imperva Application Defense Center (ADC) is a premier research organization for security analysis, vulnerability discovery, and compliance expertise. ADC research combines extensive lab work with hands-on testing in real world environments to ensure that Imperva products, through advanced data security technology, deliver up-to-date threat protection and unparalleled compliance automation.