Hacker Intelligence Initiative, Monthly Trend Report #9

Automation of Attacks

1. Overview

Cyber-criminals are increasingly using automation to carry out their attacks on web applications. This phenomenon has several reasons:

› Automatic tools enable an attacker to attack more applications and exploit more vulnerabilities than any manual method possibly could.

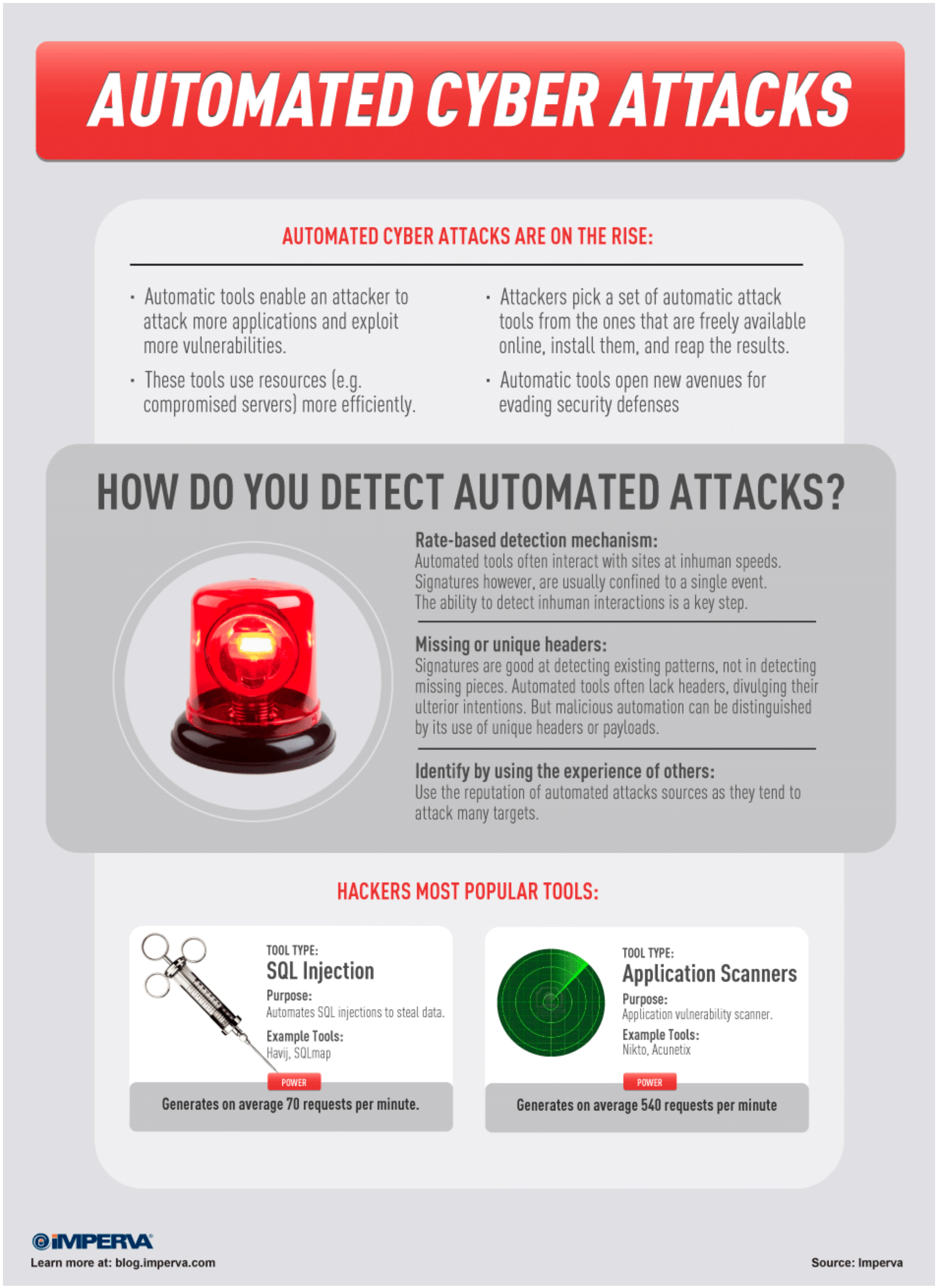
› The automatic tools that are available online save the attacker the trouble of studying attack methods and coming up with exploits to applications' vulnerabilities. An attacker can just pick a set of automatic attack tools from the ones that are freely available online, install them, point them at lucrative targets, and reap the results.

› These tools use resources (like compromised servers that are employed as attack platforms) more efficiently.

› Automatic tools open new avenues for evading security defenses. For example, such a tool can periodically change the HTTP User Agent header that is usually sent in each request to an application and that may be used to identify and block malicious clients. As another example, sophisticated automatic tools can split the attack between several controlled hosts, thus evading being blacklisted.

This report analyzes recently observed trends in the automation of attacks on web applications. The focus is on SQL Injection (SQLi[1]) and Remote File Inclusion (RFI[2]) attacks. Our analysis points out some warning signs that an automatic attack is in progress and suggests countermeasures for mitigating it. We conclude this report with an analysis of three automated attacks and demonstrate how each can be identified using the suggested identification methods.

[1] See Imperva's HII report: http://www.imperva.com/download.asp?id=352
[2] See Imperva's HII report: http://www.imperva.com/download.asp?id=60



AUTOMATED CYBER ATTACKS

AUTOMATED CYBER ATTACKS ARE ON THE RISE:

- Automatic tools enable an attacker to attack more applications and exploit more vulnerabilities.
- These tools use resources (e.g. compromised servers) more efficiently.
- Attackers pick a set of automatic attack tools from the ones that are freely available online, install them, and reap the results.
- Automatic tools open new avenues for evading security defenses

HOW DO YOU DETECT AUTOMATED ATTACKS?

Rate-based detection mechanism:
Automated tools often interact with sites at inhuman speeds. Signatures however, are usually confined to a single event. The ability to detect inhuman interactions is a key step.

Missing or unique headers:
Signatures are good at detecting existing patterns, not in detecting missing pieces. Automated tools often lack headers, divulging their ulterior intentions. But malicious automation can be distinguished by its use of unique headers or payloads.

Identify by using the experience of others:
Use the reputation of automated attacks sources as they tend to attack many targets.

HACKERS MOST POPULAR TOOLS:

TOOL TYPE:
SQL Injection
Purpose:
Automates SQL injections to steal data.
Example Tools:
Havij, SQLmap
POWER
Generates on average 70 requests per minute.

TOOL TYPE:
Application Scanners
Purpose:
Application vulnerability scanner.
Example Tools:
Nikto, Acunetix
POWER
Generates on average 540 requests per minute

Learn more at: blog.imperva.com

Source: Imperva

2. Detailed Analysis

Hackers considering how to attack Web applications weigh the benefits of a successful attack (money, prestige within their peers, etc.) against the effort, time, and money needed to accomplish the attack. Using software to attack other software shifts the balance in this tradeoff, especially if the used software is already available: an unskilled, inexperienced attacker can easily attack hundreds of applications with a press of a key within minutes and collect his reward relatively safe of the risks of detection and accountability.

Publicly available software tools assist the attacker in all the phases of an attack: finding potentially vulnerable applications (for example, using Google Dorks[3]); scanning these applications to find the unplugged security holes they actually contain; preparing exploits tailored to the found vulnerabilities; and finally exploiting the found vulnerabilities. Attack tools can work "vertically," applying various and more and more sophisticated attacks against a chosen victim, or "horizontally," doing a wide but shallow scan for easy pickings within the multitude of applications on the Web. Regardless of the way the hacker employs the tools in his arsenal, from defensive perspective identifying that incoming traffic is generated by a software tool is a warning signal that an attack may be in progress. Such an attack may be especially harmful, due to its speed and thoroughness at finding and exploiting vulnerabilities. On the other hand, if automatic attack can be reliably identified, this also presents an opportunity to block most of the malicious traffic directed at the protected application.

2.1 Automation identification methods

2.1.1 Traffic shape

Traffic characteristics like attack rate, attack rate change and attack volume are prime candidates to identify attacks that were initiated using automatic tools. Obviously, if a single host issues HTTP requests to an application at a high rate (e.g., 12 requests per minute) this traffic is software-generated. As Table 1 shows, attack rate alone is enough to flag a significant number of attacks (especially for RFI) as being software-generated.

Attack type	Attacks	Consecutive attacks from same host with minimal rate of 12 per minute	Ratio of Identified automated attacks (based on attack rate)
RFI	60,656	18,257	30.1%
SQLi	239,993	177,872	74.1%

Table 1: High-rate automated attacks, January-March

Attackers are aware that traffic shape can be used to identify and mitigate attacks that are directed at a specific site. Advanced automation tools can introduce random timeouts between attacks on a site to imitate the behavior of an innocent human user. An attack tool can also use the time between attacks on a specific application to attack multiple other applications on other sites. Thus, each application sees just the low rate of attack directed at it. Another option for attackers to avoid rate-based detection of their activity is to issue the attack from multiple hosts. This kind of distributed attack is still coordinated by a single human attacker, but the victim application logs only low traffic rate from each sender of HTTP requests.

An automatic tool attacking an unknown application must run through a long phase of exploring combinations of the application's URLs and parameters to find vulnerable ones. If the tool tries to avoid rate-based defense mechanisms by slowing down its interaction with the application, it will generate an unusually long communication pattern. A human user, on the other hand, will rarely interact with an application continuously for hours at a time.

A high rate of attacks from a host should not be used as an exclusive indicator that this specific host initiates automated attacks. For example, large parts of the traffic directed at an application may pass through Content Delivery Network servers or other types of Web proxies. Such Web elements cannot be "blamed" for the traffic that passes through them. To get a complete and truthful identification that an application is under automated attack, generic traffic shape thresholds should be combined with other attack indicators or application-specific profiling.

[3] See Imperva's HII report: http://www.imperva.com/download.asp?id=171

2.1.2 HTTP headers

Attack automation tools that are freely available online are often published as legitimate penetration testing tools. Therefore, they mark the traffic they produce with an application-specific User Agent string. For example, the popular SQL injection tools *sqlmap*[4] and *Havij*[5] send requests with a User Agent containing "*sqlmap*" and "*Havij*," respectively. These tools are often used by malicious attackers, so a black list of values of the User Agent header is sometimes useful for identifying automatic attacks, even when its traffic volume is low. However, the tools can masquerade as prevalent browsers by using the same header's value they generate. During January-March, 3.8% of the SQLi attacks we observed contained a "*havij*" User Agent, and 4.6% contained a "*sqlmap*" User Agent. Another 4.7% of the SQLi attacks had "*NetSparker*" User Agent, but they all composed a single attack. Similarly, "*libwww-perl*" appeared in the User Agent of 2.7% of the RFI attacks we observed during this period. This indicates that these attacks were initiated from scripts written in Perl – one of the favorite programming languages for hackers (though it has many legitimate uses as well).

Generally, HTTP requests generated by automation tools often do not contain headers[6] that web browsers usually send, like *Accept-Language* or *Accept-Charset* that shows the languages and character sets that the client can handle. Again, these headers can be faked, and on the other hand, a benevolent client may also drop these headers, however the absence hints that the traffic should be closely monitored.

As the figure shows, 98% of RFI attacks and 88% of SQLi attacks are automated, as indicated by the high rate of their initiation from the attacking host or the missing HTTP Accept headers in the attack messages (or both).

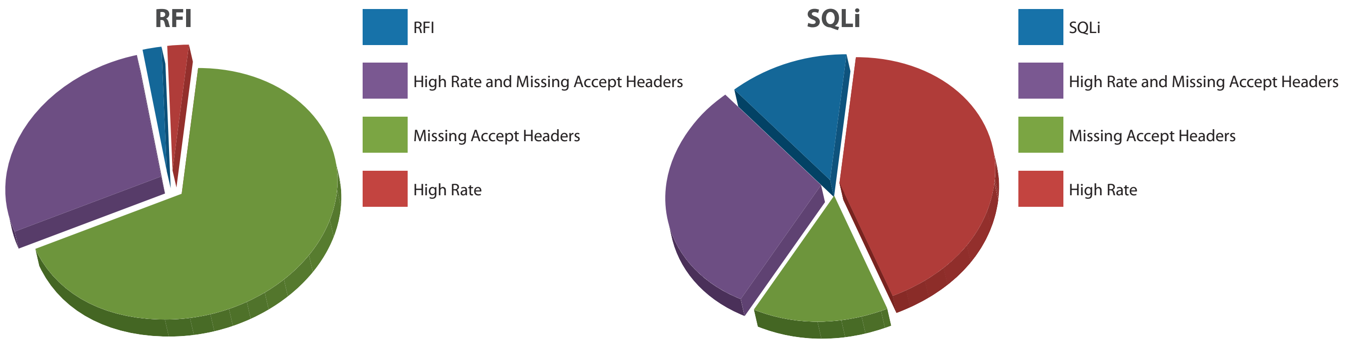
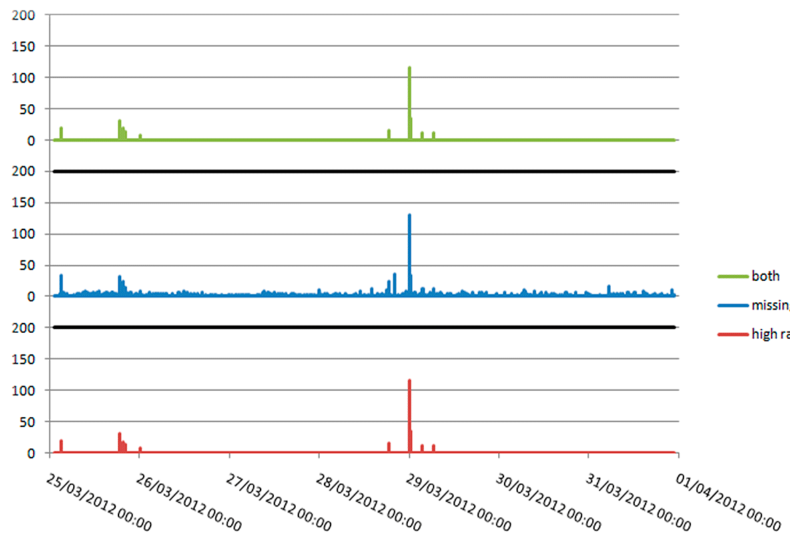


Figure 1: Distribution of RFI and SQLi attacks: Having high initiation rate and/or missing Accept headers

Looking at the pattern of occurrences of automated attacks as revealed by these indicators (Figure 2), attacks without Accept headers are sometimes sent at a high rate. However, tools also send them very frequently at low rates, as can be seen for RFI in Figure 2. This can be explained as either a deliberate ploy to avoid detection or as a side effect of the tool being used to attack multiple applications concurrently and not for a concentrated attack on a single application. Furthermore, the graph reveals different patterns between the two attack types: While in RFI, there is a lot of relatively low-rate traffic without Accept headers; in SQLi, the most dominant attacks have high rate and still contain the Accept headers. This distinction further emphasizes the importance of using multiple indicators to identify automated malicious attacks, as each of them yields somewhat different portions of the traffic.

[4] See: http://sqlmap.sourceforge.net/
[5] See: http://itsecteam.com/en/projects/project1.htm
[6] See: http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html



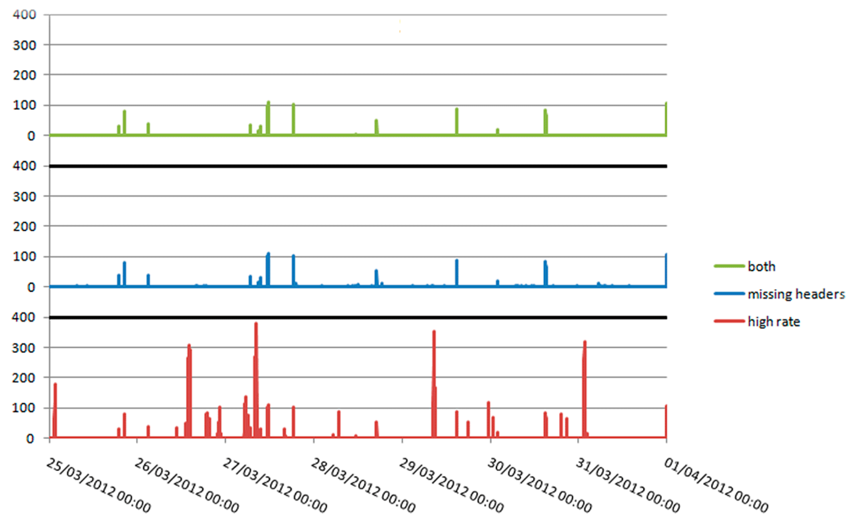


Figure 2: Occurrence of automated attacks having high initiation rate and/or missing Accept headers, during the last week of March 2012.

2.1.3 Tool fingerprints

Automatic attack tools implement some specific algorithm for finding and exploiting application vulnerabilities. Though each tool may be able to adapt to various attack targets and scenarios, the traffic it produces is eventually limited by the software itself. Analyzing the history of attacks can sometimes show patterns that match the tool that generated them, like specific strings in generated SQL fragments used in SQL injection. Such patterns can also be extracted directly from the source code of the tool, if it is available for investigation.

Fingerprints of tools can be used to identify automated attacks with high certainty. However, each one is specialized to some versions of a specific tool, so an ongoing research effort is needed to keep a list of such fingerprints relevant for newly published attack tools.

2.1.4 Host's black lists

When an automated attack is identified from a host, it is very likely that the malicious application that ran on this host will continue to attack the same or other applications in the near future. Therefore, the IP addresses of such suspicious hosts may be collected in a black list, and any traffic from them should be closely examined. The IPs in the black list contents should be expired or refreshed periodically according to new indicators of the traffic content from the respective hosts.

As an example of the potential usefulness of a blacklist of IPs that initiated automated attacks, we took a closer look at the hosts that issued attacks without Accept headers during January-March. 4,990 hosts issued such automated SQLi attacks, and 2,730 hosts issued such automated RFI attacks. We observed that for these hosts, the median of the count of attacks and attacks' activity days is low. Also, the distribution of these values is wildly divergent, as seen from the ratio of the standard deviation to the mean (see Table 2).

However, as Figure shows, a large group of the hosts involved in SQLi attacks were very actively attacking for about three weeks, and a group of hosts continued to attack for much longer, up to 80 days. Figure shows that a large portion of RFI attackers was very active during even longer periods. Attackers that were active for at least three days accounted for 30% of the automated SQLi attacks and 61% of the automated RFI attacks.

We conclude that with appropriate thresholds on attack activity period of identified automatic attackers, blacklists of these attackers can be extracted from observed attack traffic. These lists can be used to block significant portions of following attacks. As can be seen in Figures 3 and 4, in both attack types some attacks last for more than two months, thus making blacklists a useful tool for blocking attackers. The value of each list expires within a few days or weeks (depending on the attack type), but with periodic refreshment this simple mechanism is a very useful security tool.

Its importance is enhanced when blacklists are shared by communities of applications, since an attacker identified by one application can be blacklisted and blocked by all other applications before any harm can be done.

		Median	Mean	Standard Deviation	Max
SQLi	Attacks activity days	1	3.28	8.4	81
	Count of attacks	3	21.7	334.4	19,119
RFI	Attacks activity days	1	5.4	12.7	90
	Count of attacks	4	21.3	188.1	8,956

Table 2: Attacking hosts' activity characteristics

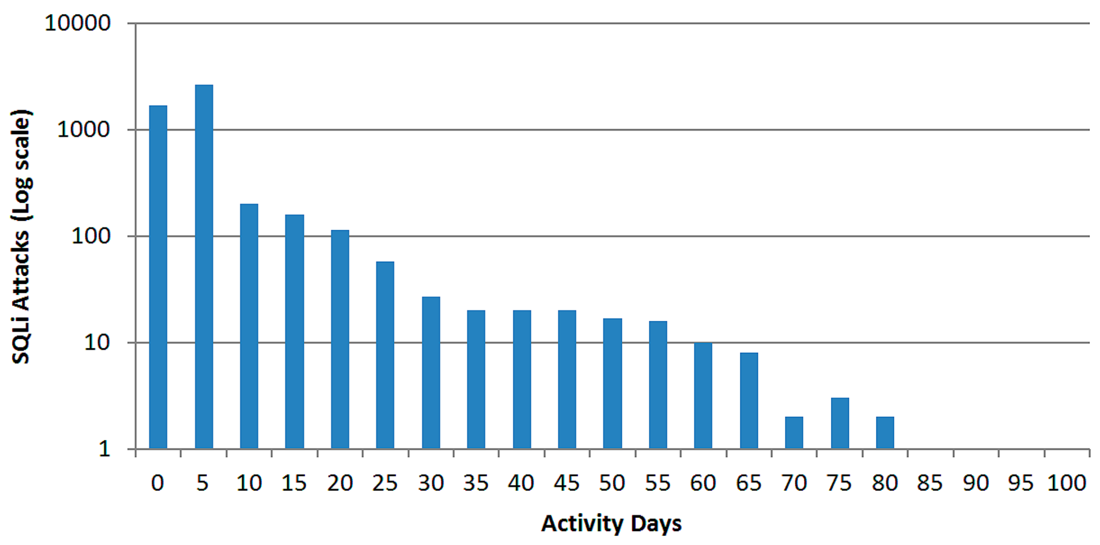


Figure 3: Activity of hosts issuing automated SQLi attacks (without Accept headers). The number of attacks is presented on logarithmic scale.

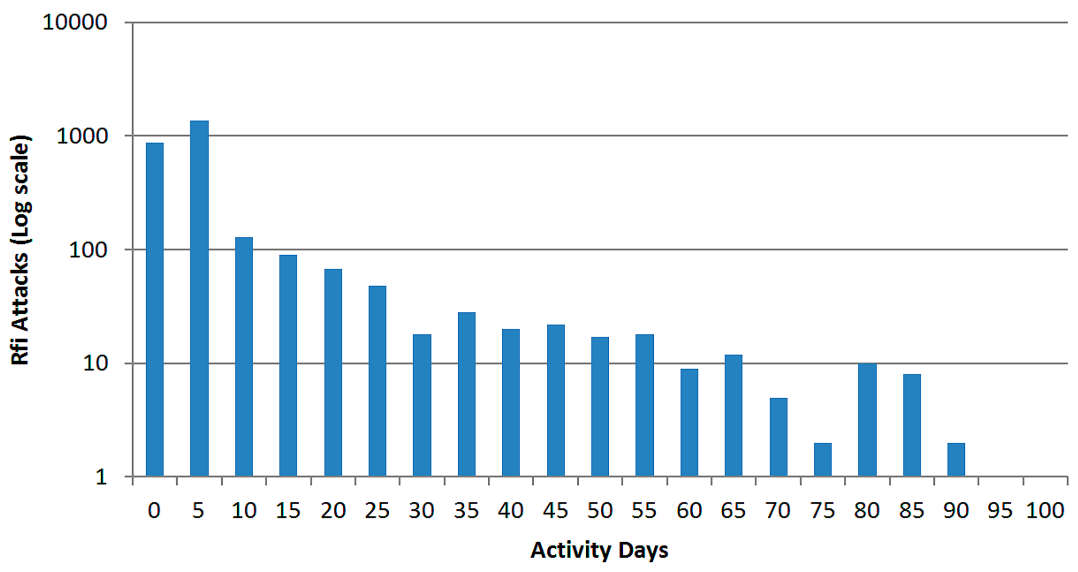


Figure 4: Activity of hosts issuing automated RFI attacks (without Accept headers). The number of attacks is presented on logarithmic scale.

2.2 Automatic attacks trends

2.2.1 Geographic origin of attacks

The observed SQLi attacks without Accept headers during January-March originated from 4,990 hosts located at 74 countries. For RFI, the attacks came from 2,730 hosts and 69 countries. As Table 3 shows 80% of the automated SQLi attackers were hosts in the United States. While 49% of RFI attackers were located at the United States, the rest were more evenly distributed. This is also illustrated in Figures 5 and 6.

SQLi			RFI		
Country	**Hosts**	**% of Hosts**	**Country**	**Hosts**	**% of Hosts**
USA	3994	80	USA	1337	49
China	355	7	Germany	170	6
United Kingdom	75	2	France	146	5
Russian Federation	49	1	United Kingdom	124	5
Canada	40	1	Canada	105	4
Republic of Korea	33	1	Republic of Korea	80	3
Germany	31	1	Netherlands	68	2
Brazil	29	1	Brazil	54	2
India	28	1	Russian Federation	51	2
France	24	1	Italy	41	2

Table 3: Geographic distribution[7] of hosts sending automated (without Accept headers) RFI and SQLi attacks

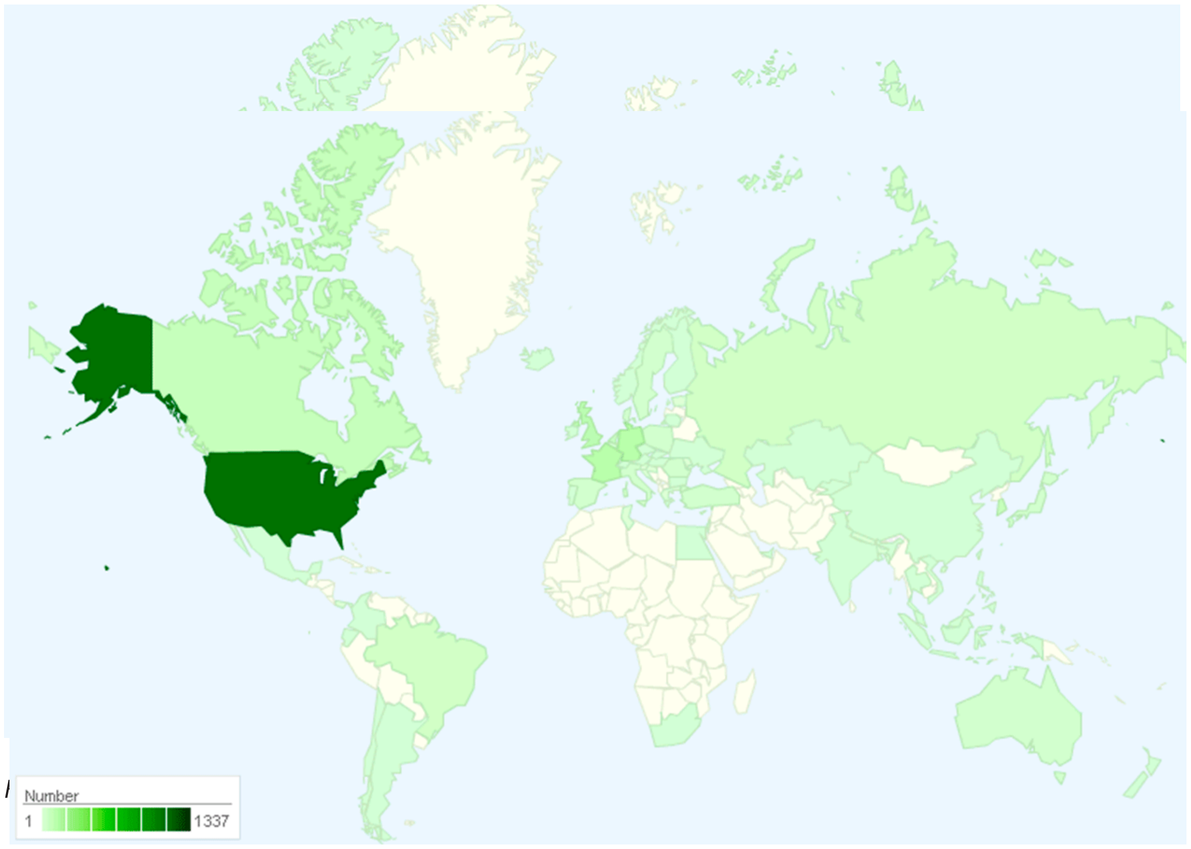


Figure 6: Hosts initiating automated (without Accept headers) RFI attacks per country

[7] The number of hosts per country is not normalized by the total number of hosts on each country

A further analysis of the attacking hosts that issued high-rate attacks (that is, higher than 12 requests per minute), revealed a somewhat different pattern.

As shown in Table 4, 30% of SQLi attacking hosts originate from China, exceeding even the USA. These two countries account together for more than half the high-rate SQLi attacks, while the other attackers are distributed among some quite unusual countries, like Egypt, Morocco, and Luxemburg. This might be consistent with the spreading use of automated, user-friendly attacking tools that enables inexperienced users to execute attacks rather easily.

SQLi			RFI		
Country	**Hosts**	**% of Hosts**	**Country**	**Hosts**	**% of Hosts**
China	98	30	USA	58	39
USA	78	24	Germany	16	11
Netherlands	9	3	France	11	7
Morocco	8	2	Republic of Korea	5	3
Egypt	7	2	Italy	5	3
Luxemburg	7	2	Netherlands	5	3
Brazil	7	2	Spain	4	3
France	7	2	Turkey	4	3
Indonesia	6	2	Canada	4	3
Russian Federation	6	2	Japan	2	1

Table 4: Geographic distribution[8] of hosts sending high-rate RFI and SQLi attacks

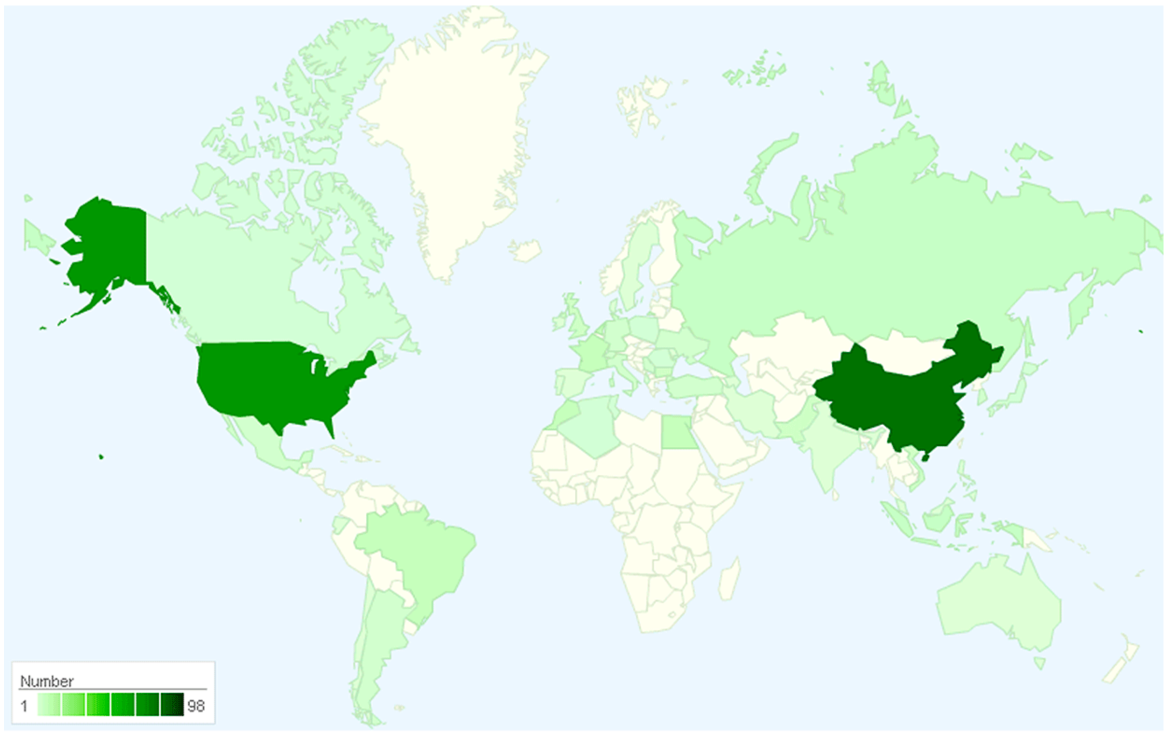


Figure 7: Hosts initiating high-rate SQLi attacks per country

[8] The number of hosts per country is not normalized by the total number of hosts on each country

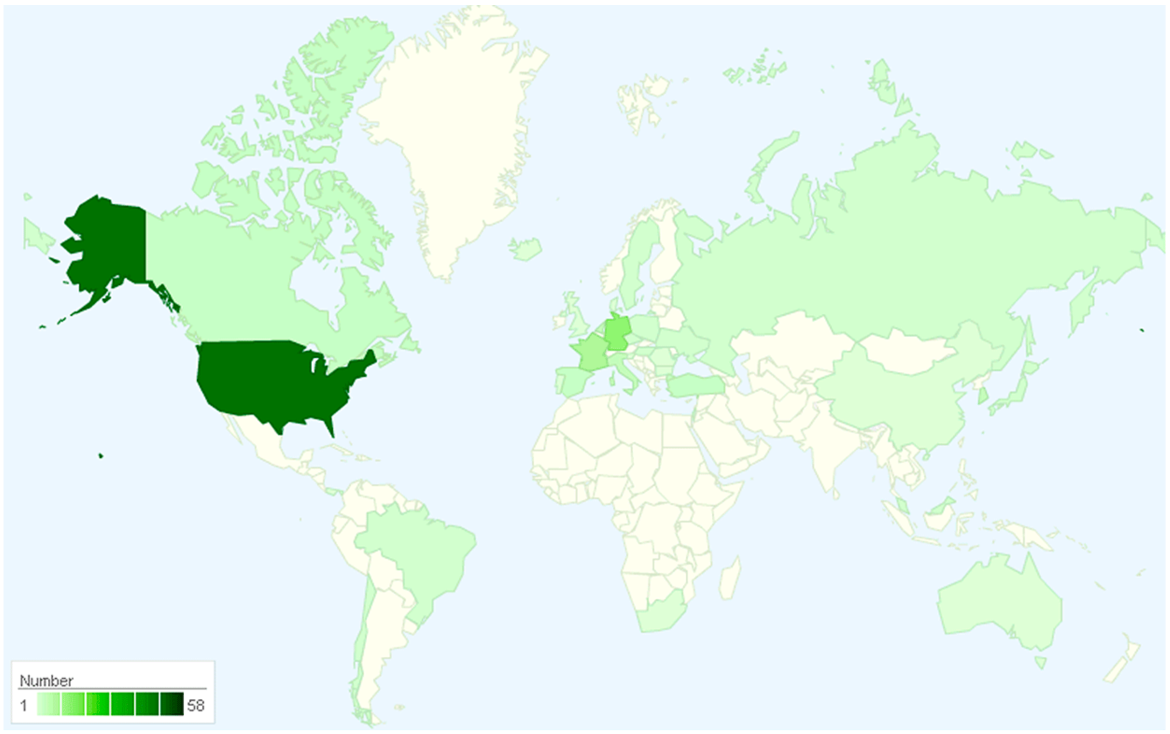


Figure 8: Hosts initiating high-rate RFI attacks per country

2.3 Compromised servers as attack platforms

We have confirmed[9] that at least 57 of the 4,990 hosts that issued automated SQLi attacks were compromised HTTP servers. Similarly, we have confirmed that 462 of the 2,730 hosts that issued automated RFI attacks were compromised HTTP servers.

The high ratio of compromised servers in RFI attacks (17%) is interesting, since this kind of attack is intended to take control of a web application. The exploits used in RFI attacks often let the attacker maintain a hidden control channel to the server on which the compromised application is deployed. Through this channel, the attacker combines the infected server into his distributed platform of computing resources, and uses it to attack yet other applications on the Internet. This growing network of maliciously controlled servers can be compared to spreading of a disease within a population through carriers.

2.4 Automation Examples

In this section we demonstrate how the above-mentioned methods can be used to identify automated attacks, as were observed in our data. Although these attacks use different attack methods, different tools and have different aims, using the outlined methods in collaboration enables us to identify all of them.

2.4.1 Havij

As mentioned earlier, Havij is a popular freely available SQL injection tool, known for its efficiency.

2.4.1.1 Rate

The average rate was 70 requests per minute, with the slowest attack having 38 requests per minute obviously much faster than any human attacker could reach manually.

[9] Other hosts that participated in the attacks may be compromised servers, but we were unable to confirm this

2.4.1.2 HTTP Headers

None of the requests had Accept-Language or Accept-Charset headers, suggesting they were not generated by a normal browser.

Most of the requests had Havij typical user agent: **Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727) Havij**. These requests also had other headers, like "Host" and "Accept." The rest of the requests had no headers at all, and, therefore, were identified by other measures.

2.4.1.3 Tool Fingerprint

The injected SQL strings usually contain a constant number that is repeated in changing amount of times. (For example: **999999.9 union all select 0x31303235343830303536,0x31303235343830303536,0x31303235343830303536—**)

This number can be used as a fingerprint unique for the identification of Havij-generated attacks.

2.4.1.4 2.4.2.4 Blacklists/ Reputation

A query in an IP database revealed that the attacking IP is recognized as a mail server and dictionary attacker.

2.4.2 Nikto Scanner

Nikto is an open-source scanner that tests for various weaknesses and vulnerabilities, among which RFI, Directory Traversal, Cross-Site Scripting, and SQL injection. It also tests for outdated versions and potentially dangerous files.

2.4.2.1 Rate

The observed attack vectors had a rate of 540 requests per minute, with the slowest vector having 264 requests per minute. Such high-rate activity should immediately set an alarm.

2.4.2.2 HTTP Headers

As stated on its website: "Nikto is not designed as an overly stealthy tool[10]…".

Thus, all the requests contain the typical user agent that contains, apart from the tool's name and version, also a "Test ID" that identifies the specific vulnerability scanned.

In the observed attack, TestIDs go from 1 to 6486. e.g. **Mozilla/4.75 (Nikto/2.1.4) (Evasions:None) (Test:000108)**

Each Test ID seeks a certain vulnerability in a predefined set of locations. For example, Test:000003 requests look for the file **cart32.exe** in 21 different locations on the attacked machine. The same goes for Test:000004, that searches the file **classified.cgi**. Each numeric TestID fits a certain file. Other Test IDs contain the type of the test, for example:

Mozilla/4.75 (Nikto/2.1.4) (Evasions:None) (Test:map_codes)

Mozilla/4.75 (Nikto/2.1.4) (Evasions:None) (Test:Port Check)

Mozilla/4.75 (Nikto/2.1.4) (Evasions:None) (Test:embedded detection)

Other than the user agent, headers like "Content-Type" and "Connection" appear occasionally in Nikto requests, but neither of the requests had Accept headers.

2.4.2.3 Tool Fingerprint

As the User Agent appears in all the requests, it serves as a good fingerprint to identify the tool.

Another unique identifier can be found in the URL Nikto uses in its RFI tests; in each request, the URL http://cirt.net/rfiinc.txt? is inserted in a different HTTP parameter. The file contains the php code: `<?php phpinfo(); ?>` that simply checks whether this parameter is vulnerable for RFI attacks.

2.4.2.4 Reputation

In this case, there was nothing to suggest the source of the attack had malicious reputation. The source IP was allocated to LightEdge.com, a Cloud computing provider.

To conclude, attacks generated by Nikto scanner have a typical user agent, no Accept headers, show high frequency of requests and have a typical URL used for RFI.

[10] See: http://cirt.net/nikto2

2.4.3 A Bruteforce Login Attack

In this attack, the attacker tried to login to the site with the username "root," using a predefined set of passwords. The passwords were tried in alphabetical order (abdol, abraham, adult, admin, ahmed, ahmet, and so on…) and no password was repeated more than once.

2.4.3.1 Rate

The attacker sent 285 requests per minute, amounting to a total of 6,526 requests.

2.4.3.2 HTTP Headers

The user agent of the requests was **User-Agent: Mozilla/4.0 (compatible; MSIE 5.00; Windows 98)**. This is weird, since the value contains the string itself "User-Agent:" at the beginning, which is not supposed to happen. We can speculate that the tool generating these requests allows the user to configure manually a user agent, and, presumably, the attacker inserted "User-Agent" at the beginning by mistake. Although this is probably a one-time incident, it demonstrates the importance of being aware of allowed and "normal appearing" headers, user agent in particular.

As in the previous attacks, this attack as well had no Accept headers but only occasional "Cache-Control," "Connection," etc.

2.4.3.3 Tool Fingerprints

Half of the requests were login attempts, in which the username and password were delivered in HTTP Parameters. The other half contained a single Parameter, "token," whose value was always set to: "**5a94a47d909ce4888776555f0f795b60**". This string may also be used in the future to identify the same tool.

2.4.3.4 Reputation

The same attacking IP was seen scanning other targets several days prior to this attack. The other scans were conducted using Dfind Scanner, another freely available scanning tool. The same IP using different tools to attack on different occasions illustrates the importance of keeping and updating blacklists of suspicious IPs.

3. Summary and Conclusions

With automation, the odds of cyber attack are close to 100%. How can security teams prepare and stop malicious, automated site traffic in order to:

› Block attacks early and efficiently.

› Defend against 0 days.

› To save analysis resources by clustering all attack vectors related to the same attack to a single group.

Detecting automation require abilities greater than plain signatures. Moreover, detecting bad automation must also allow non-malicious automation. How can you do this?

Contending with automated attacks requires:

› Rate-based detection mechanism: Automated tools often interact with sites at inhuman speeds. Signatures, however, are usually confined to single event. The ability to detect inhuman interactions is a key step.

› Missing or unique headers: Signatures are good at detecting existing pattern not in detecting missing pieces. Automated tools often lack headers, divulging their ulterior intentions. But malicious automation can be distinguished by its use of unique headers or payloads.

› Identify by using the experience of others (reputation): Automated attacks sources tend to attack many targets.

Imperva
3400 Bridge Parkway, Suite 200
Redwood City, CA 94065

Tel: +1-650-345-9000
Fax: +1-650-345-9004
www.imperva.com