The Role of Web Hosting Providers in Detecting Compromised Websites

Davide Canali
EURECOM, France
canali@eurecom.fr

Davide Balzarotti
EURECOM, France
balzarotti@eurecom.fr

Aurélien Francillon
EURECOM, France
aurelien.francillon@eurecom.fr

ABSTRACT

Compromised websites are often used by attackers to deliver malicious content or to host phishing pages designed to steal private information from their victims. Unfortunately, most of the targeted websites are managed by users with little security background - often unable to detect this kind of threats or to afford an external professional security service.

In this paper we test the ability of web hosting providers to detect compromised websites and react to user complaints. We also test six specialized services that provide security monitoring of web pages for a small fee.

During a period of 30 days, we hosted our own vulnerable websites on 22 shared hosting providers, including 12 of the most popular ones. We repeatedly ran five different attacks against each of them. Our tests included a bot-like infection, a drive-by download, the upload of malicious files, an SQL injection stealing credit card numbers, and a phishing kit for a famous American bank. In addition, we also generated traffic from seemingly valid victims of phishing and drive-by download sites. We show that most of these attacks could have been detected by free network or file analysis tools. After 25 days, if no malicious activity was detected, we started to file abuse complaints to the providers. This allowed us to study the reaction of the web hosting providers to both real and bogus complaints.

The general picture we drew from our study is quite alarming. The vast majority of the providers, or "add-on" security monitoring services, are unable to detect the most simple signs of malicious activity on hosted websites.

Categories and Subject Descriptors

K.6.5 [**Security and Protection**]: Invasive Software (e.g., viruses, worms, Trojan horses), Unauthorized access (e.g., hacking, phreaking); C.4 [**Performance of Systems**]: Measurement techniques

Keywords

Shared web hosting; web security

1. INTRODUCTION

Owning and operating a website has become a quite common activity in many parts of the world, and millions of websites are operated, every day, for both personal and professional use. People do not need anymore to be computer "gurus" in order to be able to install and run a website: a web browser, a credit card with a few dollars' balance, and some basic computer skills are usually enough to start such an activity.

Of all the possible ways to host a website, shared hosting is usually the most economical option. It consists in having a website hosted on a web server where other websites may reside and share the machine's resources. Thanks to its low price, shared hosting has become the solution of choice for hosting the majority of personal and small business websites all over the world.

Being so common, however, shared hosting websites have also high chances of being targets of web attacks, and become means for criminals to spread malware or host phishing scams. In addition, such websites are often operated by users with little or no security background, who are unlikely to be able to detect attacks or to afford professional security monitoring services.

Our work focuses on shared web hosting services, and presents a study on what shared hosting providers do in order to help their customers in detecting when their websites have been compromised. We believe this is an important commitment, given the fact that shared hosting customers are the most vulnerable to web attacks [9]. Furthermore, even a security-aware shared hosting customer would never be able to fully protect and monitor his or her account without the provider's cooperation. In fact, in a shared hosting configuration, the user has few privileges on the machine and she is not allowed to to run or install any monitoring or IDS application, nor to customize the machine's web server, its firewall, or security settings. Thus, in order to protect his or her website, a user has to fully rely on the security measures employed by the hosting provider.

In our study, we also tested the providers' reactions to abuse complaints, and the attack detection capabilities of six specialized services providing security monitoring of websites for a small fee.

In a recent survey [4], Commtouch and the StopBadware organization reported the results of a questionnaire in which 600 owners of compromised websites have been asked some questions about the attacks that targeted their websites. From this study, it emerged that, among the surveyed users, 49% of them were made aware of the compromise by a browser warning, while in fewer cases they were notified by their hosting provider (7%) or by a security organization (10%). Also, 14% of the users who took the survey said their hosting provider removed the malicious content from their website after the infection. At the end, only 12% of the customers were satisfied from the way their hosting provider handled the situation, while 28% of users who took the survey were considering to move to a new provider because of this experience.

Inspired by the StopBadware report, we decided to systematically analyze, on a wider scale and in an automated way, how web hosting companies behave with regard to the detection of compromised websites, what their reactions are in case of abuse com-

WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

plaints, and how they proceed to inform a customer about his website being compromised.

To our knowledge, this is the first work studying, on a worldwide scale, the quality and reliability of security monitoring activities performed by web hosting providers to detect compromised customer websites. Unfortunately, the general picture we drew from our results is quite alarming: the vast majority of providers and "add-on" security monitoring services are unable to detect the most simple signs of malicious activity on hosted websites. It is important to note that we do not want to blame such providers for not protecting their customers, since this service is often not part of the contract for which users are paying for. However, we believe it would be in the interest of the providers and of the general public to implement simple detection mechanisms to promptly identify when a website has been compromised and it is used to perform malicious activities.

Section 2 of the paper describes the setup and deployment of the test cases we employed to carry out our study; Section 3 reports the results of our experiments, as well as some insights on how hosting providers act with regard to preventing abusive uses of their services and web attacks against their customers' websites. Section 4 explores the related work in this field. Section 5, finally, summarizes the main findings of our work, and concludes the study providing ideas for future improvements in this area of research.

2. SETUP AND DEPLOYMENT

For our study, we selected a total of 22 hosting providers, chosen among the world's top providers in 2011 and 2012 (we will refer to these as *global-1* to *global-12*), and among other regional providers operating in different countries (referred to as *regional-1* to *regional-10*). We selected the *global* providers by picking the ones appearing most frequently on lists of top shared hosting providers published on web hosting-related websites, e.g., `tophosts.com`, `webhosting.info`, and `webhostingreviews.com`. The *regional* providers were instead chosen from the "Country-wise Top hosts" list published by the `webhosting.info` website [19], with the aim of having an approximately uniform geographical distribution over every area of the world. Our final list included providers in the US, Europe, India, Russia, Algeria, Hong Kong, Argentina and Indonesia.

For our study, we limited our choice to providers that allowed international registrations, as our hosting accounts were registered using real personal data of people belonging to our research group. In fact, we noticed that some providers, probably because of regulations in their country, limit the possibility of registering a web hosting service only to national customers. This is especially true for countries such as China, Brazil, and Vietnam, whose providers often require a national ID card number upon registration.

Also, our choice was limited to providers offering shared hosting services as part of their products, allowing to host at least one domain name per account, supporting the PHP programming language, and the FTP transfer protocol.

2.1 Test Cases

We conducted our study by registering five shared hosting accounts for each of the 22 web hosting providers. Each one of the five accounts was targeting a particular class of threat, chosen among the most common types of web attacks that could be easily detected by hosting providers.

Four out of the five test cases we deployed are based on a static snapshot of a website running OsCommerce v.2.2. The application was modified so that the PHP pages always returned a static version of the site, without the need of installing a backend database. Each snapshot was modified by hand in order to include the ad-hoc code required for our experiments, and to diversify the content, the appearance, and the images shown in each page.

Our test files were deployed in the `/osco` subdirectory of every hosting account we registered, while the home page of each domain showed only an empty page with the message "Coming soon...". We did not create any link to the `/osco` subdirectory, and we excluded the possibility for web spiders to visit our test case websites by denying any robot access using the `robots.txt` file. This was done in order to avoid external visits to our test case websites, which could have interfered with our tests.

Intentionally installing and exploiting vulnerable web applications on shared hosting accounts may raise some ethical and legal concerns. For this reason, we carefully designed our tests to resemble real compromised websites - being at the same time completely harmless for both the provider and other Internet users. For example, we modified the application code to mimic an existing vulnerability but, compared to their real counterparts, our code was executed only when an additional POST parameter contained a password that we hardcoded in the application, thus allowing only us to exploit the bug.

2.1.1 SQL Injection and Data Exfiltration (SQLi)

The first test case aimed at detecting whether web hosting providers detect or block SQL injection and data exfiltration attacks against their customers' websites. The test consisted in deploying the static snapshot of OsCommerce including a page that mimics the SQL injection vulnerabilty presented in CVE-2005-4677.

Setup - The `product_info.php` page was modified to recognize our SQL injection attempts and respond by returning a list of randomly generated credit card numbers along with personal details of fictious people (name, address, email, and MD5 password hash). In order to pass the Luhn test, fake credit card numbers were generated using an online credit card test number generator [6].

Attack - The attack for this test case was run every hour, and consisted of a script mimicking a real SQL injection attack: first, the fake vulnerable page (`product_info.php`) was visited, then a sequence of GET requests were sent to the same page adding different payloads to the `products_id` GET parameter. The first request simulated somebody testing for the presence of SQL injection vulnerabilities by setting `products_id=99'`; then, five attack requests were issued to the same page by setting the following payloads for the vulnerable parameter:

```
1 : 99' UNION SELECT null,CONCAT(first_name, ...
      customers_password),1,CONCAT(cc_type, ...
      cc_expiration) FROM customers LIMIT 1,1/*
2 : 99' UNION ALL SELECT null,CONCAT(first_name, ...
      customers_password),1,CONCAT(cc_type, ...
      cc_expiration) FROM customers LIMIT 2,1/*
3 : 99' UNION S/**/ELECT null,CONCAT(first_name, ...
      customers_password),1,CONCAT(cc_type, ...
      cc_expiration) FROM customers LIMIT 3,1/*
4 : 99' UNION S/**/ELECT null,CONCAT(first_name, ...
      customers_password),1,CONCAT(cc_type, ...
      cc_expiration) FR/**/OM customers LIMIT 4,1/*
5 : 99' UNION S/**/ELECT null,CO/**/NCAT(first_name, ...
      customers_password),1,CO/**/NCAT(cc_type, ...
      cc_expiration) FR/**/OM customers LI/**/MIT 5,1/*
```

Listing 1: Payloads of fake SQL Injection requests

The purpose of these payloads was to detect whether hosting providers employ any blacklist-based approach to detect SQL injection attempts on their customers' websites. Requests in lines 1 and 2 would fail in case the providers employ simple blacklisting rules (blocking any UNION SELECT and UNION ALL SELECT) in URLs. The last three requests would fail only if provid-

ers deploy more complex rules that are able to blacklist typical SQL words even in case they are stuffed with comments, or if words like FROM, CONCAT and LIMIT are blacklisted as well.

2.1.2 Remote File Upload (Web Shell) and Code Injection Using Web Shell (SH)

The goal of this test is to understand whether providers detect the upload and usage of a standard PHP shell, automatic file modifications on the customer's account, or the presence of malicious code on the home page of the website. In the test, a fake web shell is uploaded to the hosting account, and fake commands are issued to it, resulting in some drive-by-download code being added to the home page of the e-commerce web application.

Setup - This test uses the base static snapshot of the OsCommerce v.2.2 web application, and simulates a Remote File Upload vulnerability in the file `admin/categories.php/login.php`, as the one described in [13]. Our fake attack was designed to upload a modified version of the popular c99 PHP shell (one of the most common web shells on the web), that has no harmful effects other than the ability to inject custom code in the home page of the e-commerce web application. Also in this case, the custom code injection is enabled only when certain hidden parameters are specified along with the request of the c99 shell, thus allowing only us to trigger the injection. The content to be injected in OsCommerce's index page is a snippet of a real malicious code launching a drive-by download attack, that has been disabled by wrapping it into an `if` statement with a complex condition that is always False. We submitted the index page with the injected content to the VirusTotal online virus scanning service [1], and it was detected as malicious by 13 anti virus engines.

Attack - The test case for this attack was run every hour, and consisted in a script performing the upload of the web shell, followed by a number of commands issued on the shell. The shell file, called *c99.php* as the original shell, in order to be easily identifiable from the web server logs, was uploaded to the vulnerable URL by specifying the secret parameter enabling the upload. If the upload was successful, five commands were issued to the *c99.php*, picked randomly from a list of GET and POST requests containing both Unix commands and file names, so to make the requests seem like the result of someone trying to manually explore the contents of the server. The requests simulated actions such as trying to read files (e.g., `/etc/passwd`) and execute unix commands (`who`, `uptime`, `uname`, `ls`, `ps`). Our intuition was that hosting providers would probably be alerted by requests containing some of these filenames or commands. Finally, the test used the PHP shell to inject a plaintext version of the malicious code into the home page of OsCommerce.

2.1.3 Remote File Upload of a Phishing Kit (Phish)

Similarly to the previous test, this test uses a file upload vulnerability in the OsCommerce application to upload a phishing kit to the web server. The phishing kit consists of an archive containing a static snapshot of a real Bank of America scam. The test aims at detecting whether hosting providers are able to detect the presence of a phishing kit on the customer's account. The phishing kit was installed inside a directory named `/bankofamerica.com`, thus allowing to detect any visit to the scam pages by simply looking at the requested URLs.

Setup - This copy of the application is configured with the same Remote File Upload vulnerability explained for the previous test. However, the vulnerable path for this test is `admin/banner_manager.php/login.php`. Whenever this script is issued an upload request for a file with `tar` extension, it uploads the archive and automatically unpacks its contents to the upload directory, thus allowing for an automatic installation of the phishing kit. The phishing kit we deployed is an exact copy of a real Bank of America phishing kit found in the wild, modified to remove the back end code (thus making it unable to store and send any user information).

Attack - This attack was split in two phases, which we refer to as *attacker* and *victim*. The *attacker* phase, run every 6 hours, consisted in triggering the remote file upload vulnerability and uploading the phishing kit. The *victim* phase of the attack was run four times per hour, and consisted in a script that simulated a victim falling prey of the scam. In order to look realistic, the victim requests were disguised as coming from a range of different valid User-Agent strings used by web browsers on Windows operating systems. Every simulated victim visit comprised a sequence of GET and POST requests containing the form parameters required by the phishing pages. At each victim visit, the data sent in the requests was randomly picked among a set of fake personal details we created by hand, containing names, addresses, passwords and credit card numbers of fictitious people.

2.1.4 Suspicious Network Activity: IRC Bot (Bot)

This test aims at understanding whether providers employ any network rules to detect suspicious connection attempts to possibly malicious services. For this study, we opted to deploy to our accounts a script simulating an IRC bot. The reason for this choice is that IRC bots are probably one of the most common and easily detectable bots, because IRC connections are very often made to the standard IRC port (6667) using cleartext communication.

Setup - This test uses our basic OsCommerce installation with no modifications. The executable bot client was deployed to the hosting account via FTP, thus simulating an attack in which the attacker has stolen the customer's web hosting credentials. The files to be uploaded are two IRC client binaries written in C (one compiled for 32-bit architectures, and one for 64-bit ones), and a PHP script that executes the right binary depending on the underlying OS type, and outputs its results. The IRC client, once launched, disguises itself as "syslogd" and tries to connect to a machine hosted on our premises that runs a fake IRC server on the standard IRC port. If the connection succeeds, the client and server exchange a few messages resembling real IRC commands (such as `NICK xxx, USER xxx, JOIN #channel`) and the client reports some information about the infected machine (host name, OS type, kernel version); at last, the client closes the connection.

Attack - The test case for Bot was run every hour, and started with opening a FTP connection and uploading the two binaries and the PHP file in a new directory created in the web site's root folder. If the upload succeeded, an HTTP request was issued to the PHP file launching the IRC client. The output of this request allowed us to determine whether the hosting provider was blocking the use of possibly dangerous PHP functions (IRC client execution denied - *system()* function disabled), blocking outgoing connections to certain ports (binary executed, but connection attempt failed), or allowing everything (successful connection to the server). In order to make the upload of the IRC botnet files appear even more suspicious, the FTP upload was executed using IP addresses from several different countries.

2.1.5 Known Malicious Files (AV)

This test aimed at understanding whether providers perform any scans of their disks with off-the-shelf anti virus software. The test simply consisted in deploying, via FTP, two common known malicious files to the customer's hosting account.

Test #	SQLi	SH	Phish	Bot	AV
Blocked by ModSecurity base rule set	○	○	○	○	-
Blocked by ModSecurity OWASP rule set	●	◐	○	○	-
High severity IDS alerts	5	2	2	0	0
Detectable by antiviruses	no	yes	no	no	yes

Table 1: Attacks detection using freely available state-of-the-art security scanning tools. Legend:
○ no; ◐ in part; ● yes (full); - not applicable

Setup - Websites hosting this test used a simpler structure than the previous tests, and consisted in a single static HTML page containing random sentences in English and a few images. As in test Bot, we chose to use FTP to upload the malicious files to the account, to simulate a case in which the attacker has knowledge of the customer's account credentials. The two malicious files were *c99.php*, a real c99 PHP web shell, detected on VirusTotal with a score of 25/43 (25 antivirus engines detecting it, out of 43 it was tested against), and *sb.exe*, a copy of the 2011 Ramnit worm, detected by 36 out of 42 antivirus products according to VirusTotal. In order to make sure the malicious files were not reachable by any web visitor, but only available to people having internal access to the server, they were uploaded to a directory protected by means of *.htaccess* (denying the listing of its files) and *.htpasswd* (requiring a password to access its files from the web).

Attack - The attack itself consisted simply in connecting to the hosting account's web space via FTP and uploading every time (deleting and re-uploading if already present) the protected directory and the two malicious files. Also in this case, FTP connections were issued from IP addresses in different countries.

2.2 Attack Detection Using State-of-the-Art Tools

Before deploying the tests to the shared hosting accounts, we made sure they could be detected using common state-of-the art tools, that can be easily employed by any hosting provider. In order to do so, we executed our tests against an installation of the SecurityOnion Linux distribution, which includes a preconfigured set of open source tools for monitoring suspicious network and system activity (Bro IDS, Snort, Sguil). The installation of this distribution was then equipped with the Apache2 web server and the ModSecurity plugin, along with its base recommended rule set.

We also installed the OWASP ModSecurity "Core Rule Set", a set of common security rules for Apache ModSecurity that is maintained by the OWASP foundation [14]. These are free certified rule sets providing generic protection from unknown vulnerabilities often found in web applications. We installed version 2.2.5 of the rule set on our test machine, and disabled some rule sets (base rules number 21, 23, 30) for being too generic and generating too many false alarms. We finally ran each of the five test cases toggling on and off the OWASP ModSecurity rules.

Table 1 summarizes what we were able to detect or block using this setup, during the execution of each test. Four out of the five attacks would have been blocked or detected by employing free network and host monitoring solutions like the ones mentioned above, and the remaining attack could have been easily detected by setting up a simple connection filtering rule in the firewall.

2.2.1 SQLi

The attacks of test SQLi, when run using the basic installation of ModSecurity, succeed, but generate a series of five different high severity alerts about possible web server SQL injection attempts. When the OWASP rule set is enabled, however, all the five SQL injection attempts on which the attack is built fail.

2.2.2 SH

The SH test, executed against a webserver with the basic ModSecurity rules, successfully uploads the c99 shell and injects the drive-by code in *index.php*. However, two high severity events are raised by the IDS, one of which notifying a remote code execution on OsCommerce v.2.2 (triggered by our attack to upload of the web shell). If the OWASP rules are enabled, the remote file upload succeeds but most of the commands issued to the web shell fail and raise critical alert messages, notifying the possibility of a web file injection attack. The index file modification, finally, fails and raises a message notifying the detection of multiple URL encodings in the request, as a possible sign of protocol evasion. Finally, it has to be noted that, although we removed all the existing functionalities from the original web shell, our *c99.php* contains some original PHP code to display images and UI elements, plus our custom drive-by injection code. As such, it would still be detected during a virus scan by approximately 17% of the antivirus engines on the market (its VirusTotal score is 7/42). The *index.php* containing the injected content would instead be detected by almost 30% of the antiviruses, having a VirusTotal detection score of 13/44.

2.2.3 Phish

This attack succeeded but raised two high severity events: potential remote code execution in OsCommerce v.2.2, and presence of PHP tags in the HTTP post (detected on the tar file containing the phishing kit). On the victim's side, no HTTP request is blocked when uploading personal information to the scam pages. A possible solution to stop, or at least raise alerts on the victim's requests, however, could be deploying a simple IDS/IPS rule that detects the submission of parameters containing cleartext personal details, such as credit card numbers and cvv2 codes.

2.2.4 Bot

The Bot test case was undetected by the basic and OWASP ModSecurity rule sets, as it was run via FTP. In our tests, the connection succeeded and the bot and fake IRC server completed their message exchange. A normal firewall rule blocking outgoing connections to port 6667 (IRC) would have, however, blocked the attack.

2.2.5 AV

The malware upload test (AV) was undetected by our test deployment, because no HTTP traffic was generated and no network antivirus was used. However, as explained in Section 2.1.5, we recall that the uploaded *c99.php* and *sb.exe* are common malicious files detected by VirusTotal with a detection scores of 25/42 and 36/42, respectively. Therefore, the vast majority of off-the-shelf antiviruses would have detected them during a scan of the website's root directory.

2.3 Test Scheduling and Provider Solicitation

All attacks were run without interruption on every hosting account for the first 25 days of testing. As explained in the previous section, each attack was repeated multiple times per day in order to generate more alerts and increase the probability of being detected.

If the hosting provider did not detect any suspicious activity during this time frame, the tests entered a second phase, during

which we solicited the provider to detect our attacks and take action against them. This solicitation took place as an abuse notification email for the Phish and AV tests, in which we reported the presence of malicious files on the web application. We also generated "fake" abuse notifications to study the reaction of the providers to bogus complains.

This allowed us to understand: 1) how quickly providers respond to abuse notifications, if they ever do, 2) if they actually verify the presence of malicious content or activity on the account before taking any action, and 3) what kind of actions they take in order to stop the abuse. Abuse notifications were sent to providers by email, using real (authenticated) email addresses registered on 3rd party domains, to make them look as realistic notifications from random web users.

2.3.1 Real Abuse Notifications

Starting the 25th day of testing, we started sending one abuse complaint per day to each provider on which tests Phish and AV had not been previously detected. We stopped the notification process and the real attacks on the account either when the 30 day testing period elapsed, or after the provider responded to the notification. The notification email explained that an email had been received, with a link pointing to content hosted on the provider's premises. The link pointed to the phishing kit's index page for Phish, and to the *sb.exe* file for AV test. In addition, the email mentioned that the user's antivirus raised an alert when trying to visit the URL, and suggested the web provider to check the contents of the account.

2.3.2 Fake Abuse Notifications

Apart from real abuse notifications, we also sent emails in which we complained for perfectly clean websites. To perform this test, we cleaned and re-used the account used for the SQLi and Bot tests. The website contents were replaced by a single static HTML page containing one JPG picture and a long list of news extracted from the RSS feeds of popular international news websites. Starting on the 25th day, we sent to every provider an email per day, where the user complained about the presence of offending or malicious content on these accounts. Since at the time these emails were sent the websites were absolutely clean, these fake notifications allowed us to understand whether providers actually check the veracity of the complaints they receive before taking any action. The first complaint email was from a user pretending that the website's content was offending his religious views, and kindly asking to stop the website owner from spreading such disrespectful messages. In the second scenario, the notification email was from a user claiming to have received an email with a link to the website in question. The user explained that his browser denied access to the URL, and that at a closer look the website looked like hosting a phishing scam. Also in this case, the account hosting the reported webpage was absolutely clean, hosting only the benign static HTML home page.

One may argue that, in case of these fake notifications, the provider could react by suspending or shutting down the user account by having a look at the logs of the machine on which the account was setup, and noticing past malicious activity, even though, at notification time, the website was clean. We did our best in order to avoid this from happening, by deploying our tests for fake notifications on accounts that hosted the SQLi and Bot tests. These tests could not be considered malicious (no malware nor phishing files were ever uploaded) but the mere evidence that the website was under attack. Moreover, attacks for these tests could only have been detected at a network level, since no trace was left on the disk.

3. EVALUATION

During our experiments, we evaluated the security measures put in place by web hosting providers to detect malicious activities, compromised websites, and prevent abuse of their services. We group our findings in three categories: account verification upon signup (3.1), compromise prevention and detection capabilities on live websites (3.2.2), and responses to abuse notifications (3.3).

3.1 Sign-up Restrictions and Security Measures

Even though our work was not meant to test the anti-abuse signup policies of web hosting providers, we report here some results that may contribute in understanding how much effort providers put in preventing services subscription by malicious users.

Several providers try to discourage abusers by asking to verify the information entered during the signup phase, either by calling the customers on the phone, or by requiring a scanned copy of their documents (such as government issued ID, credit card used for the purchase). Some providers also use 3rd party fraud protection services, that block purchases based on a set of heuristics. For example, we observed several cases in which the providers correlated the geographic location of the customer, the billing information, and the IP address used for the purchase.

The shared hosting accounts we used for our study were all registered using real personal information of people working in our group, and the billing information of our research institute. The sign-up process was carried out from several IP addresses, using either credit card or PayPal payments.

Anti-abuse signup policies vary widely between hosting providers. Top *global* hosting providers are more cautious with regard to signup, often blocking attempts - e.g., blocking multiple registrations from the same billing address and credit card number, verifying the customer's personal information by verification phone calls or ID and credit card checks. *Regional* providers seem to be more permissive, probably because they have less incentives in making their signup process more difficult, which could make them lose potential customers.

Among the twelve *global* providers, seven of them required us to verify our account information for at least one of the accounts we registered with them. In order to verify our account information, all these companies required a scanned version or photocopy of a government issued photo identification card (such as passport or driver's license) and the front and back of the credit card used at signup (without showing the first 12 numbers and the cvv2 code). Only one out of these seven companies claimed, on its website, to manually verify every customer's signup before allowing the purchase of its web hosting services. Indeed, this was the only provider that verified every account we registered with them.

Regional providers, instead, do not seem to be as cautious during the account signup phase. Only one out of ten blocked an account creation because of a mismatch between our billing address and the geolocation of the IP address used for registration.

Finally, three of the *regional* providers we tested had a very simple signup process, where users could register an account in one click, by filling all the required personal and payment information in one page. These providers never asked us to verify our information upon registration, and thus could possibly be a good choice for criminals wanting to perform abusive subscriptions.

Signup verification requests are either sent during registration or after a successful account registration and activation. While requiring an account verification upon signup can be effective in preventing malicious registrations, it can also make the hosting provider lose potential good customers that may not have time or patience

Provider	Verification time		
	Before payment	Before activation	After activation
global-2	25%	-	50%
global-3	25%	-	25%
global-4	33%	-	-
global-5	40%	-	-
global-6	-	33%	-
global-7	100%	-	-
global-8	50%	25%	-
regional-2	33%	-	-

Table 2: Account verification times. Values represent the percentage of verification requests on the number of accounts we registered for each provider. "Before payment" means during the registration process. "Before activation" means once the client's billing account is created, but the hosting service is not yet active. "After activation" indicates when the hosting account is active and a website has possibly already been installed.

to provide all the required information. On the other hand, requiring an account verification once the service has been purchased and set up has the drawback of temporarily suspending an account on which a website has already possibly been deployed, thus causing a service outage for a benign customer. During our experiments we encountered both situations. Table 2 shows the percentage of verification requests on the number of accounts we registered for each provider, grouped by the time at which the request was issued. Only providers that requested at least one account verification are listed.

The table shows that, in general, most of the anti-abuse systems send alerts and block a registration attempt during the customer's signup phase. This typically happens when the user enters his or her credit card details and tries to complete the hosting purchase. Others, instead, let the client sign up for the service and receive its management panel credentials, but lock the web hosting service activation until a copy of the customer's document is received by the support department. Two web hosting providers (*global-2,3*) sent verification requests when the web hosting account was already active and the customer's website deployed. This caused a temporary service disruption for the affected accounts, making their websites unavailable for several hours. Certain providers, finally, issued verification requests at different times, probably depending on the kind of alert they received from their abuse prevention system (*global-2,3,8*).

3.2 Attack and Compromise Detection

During the first phase of our experiments, we deployed our five test suites on every hosting provider and recorded whether the hosting provider took some action or contacted us to notify that malicious activity was observed on our account. As explained in Section 2.3, if no malicious activity was detected on the account during the first 25 days, we started sending abuse notifications to the hosting provider, in order to stimulate a response. The results of this second phase are summarized in Section 3.3.

To make our fake attacks look realistic, our test cases were run automatically at certain time intervals (as explained in Section 2.1), and the attacks were executed from different IP addresses belonging to several different countries. Also, in order to avoid having only "artificial" malicious requests in the web server logs of our accounts, we generated some background traffic simulating real visits to our websites. This was accomplished by developing a simple traffic generator tool, that visited every account we deployed every 10 minutes, and randomly followed links on every website up to a depth of 30. In the general case, this meant following an average of 13 links on every website, thus generating a bit less than two thousand hits per day on every active account. The machine used for traffic generation was not used for other experiments and used a different set of public IP addresses than the ones we used to run the attacks.

3.2.1 Attack Prevention

Even though our study focuses on the ability of the providers to detect compromised websites, during our experiments some of our attacks were blocked and were therefore ineffective. In some of these cases, we proceeded by manually compromising the account. For example, whenever a provider denied the possibility of running test SH, we manually uploaded the drive-by download code to the account to continue the experiment. This allowed us to test whether the provider was able to detect the menace by scanning the customer's account. For the phishing attack (Phish), since it had to be detected on a network level, we did not take such measure and thus no manual upload was performed on accounts of providers blocking the remote file upload.

Table 3 reports, for each test and provider, whether the web hosting company took any measure to prevent the attack. Such measures depend on the test case, and ranged from employing URL blacklists to blocking outgoing connections or process executions.

URL blacklisting.

Some providers employ URL blacklists in order to prevent SQL injection attempts (test SQLi) and remote file uploads (SH, Phish).

However, as shown in column *SQLi* of Table 3, none of the providers we tested were able to fully prevent our SQL injection attacks. This is probably due to the adoption of simple keyword-based blacklisting rules, that can be easily bypassed by introducing SQL comments in the middle of blacklisted keywords (such as using "SE/**/LECT" instead of "SELECT", as explained in Section 2.1.1). Two providers (*global-1, regional-2*) blocked the first four requests of our attacks, and other five providers were able to block only the first two. The remaining did not adopt any SQL-injection protection mechanism at all.

Regarding tests SH and Phish, some providers were able to prevent the attack by employing URL blacklists probably containing specific rules for the detection of common vulnerabilities on web applications, such as the ones we employed for the tests presented in Section 2.2, provided by the OWASP foundation. Regarding SH, Table 3 shows that some providers were able to only partially prevent the attack. These providers did not block the file upload itself, but employed blacklisting rules to block some requests to the web shell (these requests contained common file names, e.g., */etc/-passwd*, or common parameter sequences such as *.php?act=cmd*).

Connection and OS-level filtering.

The attack files for test Bot were first uploaded to the customer's account via FTP, then the fake IRC client was executed issuing a HTTP request to a PHP file launching an executable file using the *system()* PHP function. A total of 18 providers were able to fully stop the attack: of these, 50% did so by disabling the *system()* function in PHP, while the remaining half firewalled outgoing connections to the IRC port.

When the attack was prevented, we were expecting some form of notification regarding the suspicious activity. After all, it is not

normal that a shared hosting user has a disguised process that tries to connect to an IRC server every hour for one month.

Two hosting providers allowed the attack only at certain periods in time (*global-2* and *global-6*). This may due to temporary misconfigurations on their networks or to automatic account migrations over different machines with different configurations (for example, the account running test Bot on provider *global-6* connected to our fake IRC server from eight different hosts during the 25 days testing period).

No prevention results are shown for test AV, as this test did not run any attack and no filtering was done on the upload of malicious files via FTP.

As a final remark, we noticed that, for some tests, some providers had exactly the same behavior. This is the case, for example, of *global-1* and *regional-2* and *global-8* and *regional-3*. We thus believe that these providers employ the same protection mechanisms and web server security configurations for their shared web hosting solutions. These services are probably provided by third party companies as part of common off-the-shelf security solutions.

3.2.2 Compromise Detection

Sadly, all but one of the providers we tested did not notify their clients when their websites were compromised and were used to perpetrate obvious malicious activities.

The only hosting provider that reacted to one of the attacks was *global-4*, but that reaction happened 17 days after the beginning of test AV. The provider properly notified the presence of a malicious file (the `c99` shell) on the user's web hosting account. In addition, the provider warned the user that a service suspension would occur if no reply to the alert was received by the customer support service within 24 hours. However, the message was not mentioning the presence of the other malicious file on the account, namely, `sb.exe`. This suggests that the alert was an automated message resulting from a virus scan of the account, and that no human operator actually checked the contents of the directory in which the two malicious files were stored.

We were quite surprised by our findings, as we were expecting to have at least a few of our scenarios detected by the vast majority of web hosting providers. It emerges that, on shared hosting servers, even the most basic virus scan is not as common as one could expect. From our measurements, we are not able to tell if the hosting providers run antivirus systems on their shared hosting servers. However, if they do, they are either using outdated signature definitions, or the frequency at which they perform the scans is less than once a month.

3.3 Solicitation Reactions

As explained in detail in Section 2.3, whenever one of our test suites was not detected by the hosting provider for 25 consecutive days, we started sending daily abuse notification emails to the provider's abuse contact. The purpose of sending these messages was to understand whether web hosting providers respond and react to abuse notifications (e.g., by suspending a compromised account or notifying the customer of his or her website being compromised). To complete our test, we also sent fake abuse notifications for perfectly clean webpages, with the aim of understanding whether any providers take action without first verifying the claims. This would pose a serious menace, as it would be a very easy and effective way to conduct a Denial of Service attack against websites of other users. The following paragraphs are meant to give some insights and details on what is presented in the "Solicitation Reaction" section of Table 3.

3.3.1 Abuse Notifications

Unfortunately, 50% of both the *global* and *regional* web hosting providers never replied to any of the real abuse notifications we sent. This percentage is quite alarming, and means that if a website is hosting malicious content (such as phishing or malware), no action will be taken to stop it from spreading and reaching its victims. Moreover, phishing attacks and malware files used in dropzones usually have a short lifetime, and, as such, even a late response to a malware or phishing abuse notification would have little or no effect on the general outcome of the attack.

Seven out of the eleven providers that replied to our complaints replied either the same day or the day after the notification was sent. This is a good indicator, meaning that these companies probably care about web abuses and are able to handle these issues in a timely manner. The only provider that replied later than 5 days after the notification was *regional-5*, with an average response time of 16 days. After such a long delay any action would be basically useless, as the website may have completely changed in the meantime.

There were a variety of reactions to our abuse complaints. The most common approach was to temporary suspend the customer's account, with five companies performing at least one suspension as result of a malware or phishing abuse complaint. We consider this action a reasonable response to the abuse, causing a temporary disruption of the services the client is paying for, but blocking the immediate threat. Other providers responded to the notifications by cleaning up the account, removing the suspicious files (4 providers - note that this action seems to be more common among *regional* providers), or by forwarding the abuse notification to the customer (1 case). We considered such responses, in general, to be appropriate to stop the menaces from spreading, and at the same time avoiding to impact too much the user's services.

Provider *global-12* reacted without notifying the website's owner: in the case of AV, the account was terminated, while in the case of phishing (Phish), the directory containing the fake phishing kit was removed. Also in the case of provider *regional-6*, actions were taken without notifying the user, with the exception that, in this case, the reactions to the abuse notifications consisted in deleting all the files (including the clean ones) of the user's websites!

Controversial responses to our abuse notifications were those from providers that sent ultimatums to the user (marked with U, in the table), warning him that offending content had been found on his website, and that if no cleanup was performed within a few hours, the account would have been suspended. This was controversial because, as in the case of provider *global-6*, even though we did not take any action to respond to the provider ultimatum, the fake phishing pages were still present on our account after several days. This means that the provider did not keep to its commitment.

Finally, a few responses were partially or fully unsatisfying. The *regional-3* provider replied to the malware abuse complaint probably after scanning the customer's account using an antivirus. The reply stated that a c99 PHP shell had been found on the account, and asked the notifier if he wanted them to remove it. The malicious executable was not mentioned at all and no further action was taken, thus leaving both malicious files on the account. The case of providers *global-2, global-3* and *global-5* is quite particular. While experiments were in progress on most of the providers, and once our tests Phish and AV reached their 25th day on *global-2* and *global-5*, notifications were sent to the two providers. First, provider *global-5* replied by terminating the account (disabling both the billing and the hosting account) and giving the customer 15 days to reply and to recover his files. We replied, asking to reenable the account for recovering our files, but in the meanwhile another abuse response was received from provider *global-2*, ter-

Provider	Account verification	Attack Prevention/Detection (days)					Solicitation Reaction		
		SQLi	SH	Phish	Bot	AV	Abuse complaint	Fake abuse complaint	Avg. reply delay (days)
global-1	○	◐/ ○	●/ ○	●/ -	●/ ○	- / ○	○ N	● N	-
global-2	◐	○/ ○	○/ ○	○/ ○	◐/ ○	- / ○	○ T	- -	1
global-3	◐	- / -	○/ ○	○/ ○	●/ ○	- / ○	○ N/T	- -	-
global-4	◐	○/ ○	○/ ○	○/ ○	●/ ○	- / ●(17)	● S	◐ U	0
global-5	◐	- / -	○/ ○	○/ ○	●/ ○	- / ○	○ T	- -	0
global-6	◐	○/ ○	○/ ○	○/ ○	◐/ ○	- / ○	○ U	● O	2
global-7	●	◐/ ○	○/ ○	○/ ○	●/ ○	- / ○	○ N	● N	-
global-8	◐	◐/ ○	○/ ○	●/ -	●/ ○	- / ○	○ N	● N	-
global-9	○	○/ ○	●/ ○	●/ -	●/ ○	- / ○	○ N	● N	-
global-10	○	○/ ○	●/ ○	●/ -	●/ ○	- / ○	◐ S	● N	4
global-11	○	○/ ○	○/ ○	○/ ○	●/ ○	- / ○	○ N	● N	-
global-12	○	○/ ○	○/ ○	○/ ○	○/ ○	- / ○	◐ T,C	● O	0
regional-1	○	◐/ ○	◐/ ○	○/ ○	●/ ○	- / ○	● S,C	○ S	0
regional-2	◐	◐/ ○	●/ ○	●/ -	●/ ○	- / ○	○ N	● N	-
regional-3	○	◐/ ○	○/ ○	●/ -	●/ ○	- / ○	◐ O,C	● O	0
regional-4	○	○/ ○	○/ ○	○/ ○	○/ ○	- / ○	○ N	● N	-
regional-5	○	○/ ○	○/ ○	○/ ○	●/ ○	- / ○	◐ S	● O	16
regional-6	○	◐/ ○	◐/ ○	○/ ○	●/ ○	- / ○	◐ C	○ C	1
regional-7	○	○/ ○	○/ ○	○/ ○	●/ ○	- / ○	○ N	◐ U	5
regional-8	○	○/ ○	○/ ○	○/ ○	●/ ○	- / ○	● S,F	● O	1
regional-9	○	○/ ○	○/ ○	○/ ○	●/ ○	- / ○	○ N	● N	-
regional-10	○	○/ ○	○/ ○	○/ ○	●/ ○	- / ○	○ N	○ P	0

Table 3: The results of our study. Legend:

-	**not applicable**	**N**	**no reply**	**P**	**forced password reset**
○	**no / not satisfying**	**S**	**account suspension**	**C**	**cleanup or file removal**
◐	**in part / partly satisfying**	**T**	**account termination**	**U**	**ultimatum to the user**
●	**yes (full) / satisfying**	**F**	**complaint email forwarded**	**O**	**reply but no action**

minating our account. Starting that moment, within a few hours, all the accounts we had registered on providers *global-2, global-3* and *global-5* were terminated without any explanation, even when we tried to contact the companies to ask details about the reasons of our accounts' termination. The only response we were able to get was: *"Due to certain items contained in the account information, this account was flagged for fraud. For security reasons, this flag caused the system to delete your account. At this time we ask you to seek out a new hosting company."*

Either the three companies used the same support service, provided by a third party, or they shared information between them. Indeed, the termination notifications for all the accounts on the three providers were sent by the same support representatives, and contained exactly the same text (only the email signature changed, containing the email and postal address of the appropriate company). For this reason, we expect the support center for these companies was able to link our accounts' personal information and understand they were all registered by the same group of individuals. Thus, having received complaints for two of the accounts, all the other accounts that could have been reasonably linked to them were terminated as well.

When this happened, some test cases had not been deployed yet on these providers (SQLi on *global-3, global-5*) and others had not yet reached their 25th day of execution (Phish on all, and SQLi on *global-2*), thus no fake abuse notifications were sent for them. This explains why Table 3 has missing data for such providers in columns "SQLi" and "Fake abuse complaint". This is also why in the "Abuse complaint" cell for provider *global-3*, we listed N/T: no abuse notification response was received (N), but a termination occurred anyway (T) for other reasons.

Finally, for provider *global-9*, we were not able to properly contact its abuse department: out of the four different abuse notifications we sent to its abuse email address, only the last two received an automated reply, saying that in order to report an abuse, it is necessary to click on the help link on the web hosting provider's home page and follow a series of steps (at the time we received these responses, the five-days testing period was already expired). We flagged this case as "no reply" because, although we tried to submit the complaints following the company's advice, the user interface adopted by the provider makes it very difficult, even for an experienced user, to find the right way to report a website abuse. Moreover, once a visitor is able to reach the right page for submitting a website abuse notification, he or she is required to register an account before being able to file a complaint.

3.3.2 Fake Abuse Notifications

We expected most web hosting providers to ignore our abuse notifications regarding "offending content" (see 2.3) and to check the website's contents but take no action in case of the fake phishing complaints. In Table 3, we thus marked as "satisfying" also the providers that never replied to our complaints. However, this is not always a good sign, especially when the same provided never responded to the real complaints.

Sadly, some of the reactions we observed were clearly in contrast with our expectations. Both providers marked with "U" believed either our religious complaint (*global-4*) or our phishing

one (*regional-7*), warning the website owner about the possibility for his account to be suspended if the offending content was not removed within a few days. However, contrarily to what was promised, the content of the websites was left untouched and none of these providers took any action to block the user's account after the ultimatum expired.

One provider, *regional-1*, suspended one of our clean accounts on the same day it was notified as hosting a phishing website. *regional-6*, instead, acted as in the case of real abuse complaints: all the pages on the account's web hosting directory were deleted, and the website's home page was replaced by an "under construction" page. This was already bad when associated to a real malicious content, but in case of a bogus complaint it is really an unacceptable behavior. One last provider, then, responded to the fake phishing abuse notification by sending the website owner an email stating that his website has been attacked, and as such a password reset had been forced on the account. Furthermore, the malicious files were disabled (by means of changing their access permissions) and their list was sent to the user: the list contained the benign website home page and the `jpeg` picture included in it. We were not able to figure out how the web hosting provider assumed the static HTML home page and the picture could contain malicious code.

Only four web hosting providers replied to our fake abuse notifications with messages that completely satisfied our expectations. In these cases, marked with "O" in the table, the support representative informed the notifier that upon manual inspection, the website seemed to be clean, and, in case some content seems to be offending somebody's cultural views, the issue has to be resolved in person by contacting the owner of the website. From this analysis it seems that *regional* providers are slightly more likely to perform a manual content inspection on the websites they host (at least 30% of the ones we tested), compared to *global* providers (only two out of twelve).

3.4 Re-Activation Policies

Whenever an hosting account was suspended, providers often provide the customer with the steps to follow in order to have the account re-activated. These steps usually imply changing every password of the account (billing, FTP, database passwords, etc.), writing a letter or an email stating the agreement to the provider's Terms of Service, and removing the malicious files or re-installing a clean copy of the website. Among the companies that suspended our accounts, *global* hosting providers seem to stick to strict legal requirements before allowing customers to have their accounts re-activated after a violation of the terms of service. The two hosting providers that suspended at least one of our accounts required us to send an email (*global-4*) or a scanned letter or fax (*global-10*) to their support department, stating that we have followed all the necessary steps to clean up our account and reset our login credentials, and that in future we will abide by the terms of service of the company. Regional providers appear to be more "informal" with regard to this, as often a simple email replying to the incident notification, explaining that we were running a vulnerable web application or using a weak FTP password, was sufficient to have our account re-activated. Also *regional* providers, however, in their incident notifications, advised the user to follow basic steps to secure his account (password change, website cleanup) before requesting a service re-activation. During our tests on *regional-1*, in one case, a scanned version of the customer's identification card was required in order to re-activate a suspended account.

Finally, in the case of service terminations, the providers just wanted the user to leave their company, replying to service re-activation requests with emails stating in that, given the kind of activity encountered on the account, the company was not willing anymore to provide their service to such customers.

3.5 Security Add-on Services

In our study, we also evaluated the ability of third party "add-on security providers" to detect attacks or abuses on a website. These services can be purchased separately from web hosting accounts*, and associated with a domain or website to monitor. In some cases, the subscriber has even the option to give his FTP/SFTP access credentials to the security service, to allow an in-depth scan of all the files on his or her account (also those that may not be reachable from the web). For our study, we selected four companies offering such security services, chosen among the most common and advertised on the web. We limited our choice to services that are affordable for a personal or small business use ($30/month max subscription price). We did so in order to test services that are in line with the level of web hosting we were testing. Indeed, it would not be reasonable to pay hundreds of dollars per month, or more, to protect a $10/month hosting plan.

Some of the add-on companies we evaluated are proposing several level of service, at different pricing. We thus registered every protection level available, up to the $30/month threshold we had fixed, ending up registering a total of six security add-on services (two each from the companies offering multiple levels of protection). Six additional hosting accounts were purchased, from different companies, in order to accommodate our tests for these security services. In the following, we refer to them as *sec-1* through *sec-4*. The two variants for companies offering different levels of protection are labeled with a *-basic* or *-pro* suffix, to distinguish, respectively, the cheapest version of the service from the more expensive one. Services in the *-pro* version, for both providers *sec-1* and *sec-2*, allow to scan, daily, all the files on the customer's FTP hosting account, if they are provided with his or her access credentials. We configured both services to enable this kind of scans. The other four security services, contrarily, perform only scans on publicly accessible pages of the websites they are configured to monitor. Such scans include, in most of the cases, checking for malware, malicious links, blacklisted pages, and performing reputation checks on both the website and the provider hosting its contents.

3.5.1 Evaluation of the Security Services

The security services' evaluation schedule was tighter than the normal test evaluation schedule, as we expected security add-on services to react faster to attacks and suspicious account activities, being specially designed for detecting security issues. Thus, the tests on accounts hosting the security add-on services were run for a total duration of 50 days, 10 days for each test, from SH to AV. The SQL injection test was not run on such web hosting accounts, because its attack does not generate any side effect on the hosting account and thus could not be detected by third party external security services.

We noticed that two of the companies providing the add-on security services are listed among the partners of known URL blacklisting services. We therefore used the last 10 days of testing to study reactions to the notification of suspicious URLs to such blacklists. For this, we scheduled a last test consisting in a new deployment of SH, along with the submission of its drive-by download page to a few malicious URL reporting and blacklisting services. The

*Although these services can be purchased separately, several web hosting providers offer security services from third party companies at a discounted price, if purchased in conjunction with a web hosting plan.

Provider	Attack Detection				
	SH	Phish	Bot	AV	SH-BL
sec-1-basic	○	○	○	○	○
sec-1-pro	○	○	○	◐	●
sec-2-basic	○	○	○	○	○
sec-2-pro	○	○	○	○	○
sec-3	○	○	○	○	○
sec-4	○	○	○	○	○

Table 4: Results of our evaluation of third party security services. Symbols and their meanings are the same as in Table 3.

URL blacklisting requests were sent on the same day the tests were deployed. We refer to this test as "SH-BL".

Results are shown in Table 4. One can see that detection capabilities for add-on services are comparable to those of providers. However, in this case, customers pay for a service whose only commitment should be monitoring a website in search of potential vulnerabilities or malicious content. Almost all the services we tested in this part of our study seem to completely fail this objective.

All the services were configured to send notifications to the user whenever a security issue was detected on the monitored website. None of the add-on security services detected anything anomalous during our tests SH, Phish, Bot (attacks were all successful and never blocked by the hosting provider). Test AV was not detected either, but the *sec-1-pro* service raised a warning for having detected the c99 web shell on our hosting account. However, this alert was visible only when logged on the security service's web management panel, where the c99.php file was listed as suspicious. No critical alerts were issued, nor any email was sent to the user as notification for this event. Finally, the only successful detection was performed by the *sec-1-pro* service, detecting our drive-by download page the day following our blacklisting request for its URL. As the *sec-1* security company was listed as one of the partners of the blacklisting service, we expect that our URL blacklisting request was forwarded to the security service right after our submission, thus allowing a timely detection.

4. RELATED WORK

Several works have studied the threats that affect websites all around the world as well as users visiting infected pages [15–17]. Research has been focusing also on the ways in which criminals exploit search engines in order to reach their victims, by poisoning search results for popular queries [7]. Other papers have explored how similar techniques are used in order to find vulnerable websites [12] and web servers [8]. Researchers have also studied how all these activities are combined by criminals in order to be able to conduct attack campaigns in which tens of thousands of hosts are infected [18]. Canali et al. [3] studied the behavior of actual attackers on the web, by installing vulnerable web applications in a controlled environment.

Bau et al. [2] evaluated current commercial tools for detecting vulnerabilities in web applications. Such tools mainly rely on black-box approaches, and are not able to find all possible vulnerabilities.

Recently, a web hosting provider [5] anounced an improvement of his hosting offer by adding free automated website vulnerability scanning, fixing and recovery. Such service is premsumably running as white-box approach on the network and server side. This service is related to what, in our work, we refer to as "add-on" security services. Unfortunately, this service was announced when our experiments were already completed, and it was therefore not possible to integrate it into our results.

Commtouch [4] surveyed 600 compromised websites owners and, among other things, reported on the process by which the websites owners became aware of the compromise. However, this was done with a publicly advertised pool on *detected* compromised websites and may therfore be biased.

Finally, some past work bas been focusing on studying the take-down process employed in the case of phishing websites [10, 11]. This is related to some of the findings we reported in Section 3.3, but is aimed at studying the phenomenon at a ISP and hosting provider level, rather than analyzing the providers' responses one by one and provide details on how they react to abuse notifications.

To our knowledge, this paper is the first attempt to systematically study, on a worldwide scale, how web hosting providers act with regard to the security of their customers and of their own infrastructure - focusing in particular on the detection of compromised accounts, rather than the presence of vulnerabilities.

5. LESSONS LEARNED, CONCLUSIONS

We can summarize the main findings of our experiments around the following five points:

Registration - Top providers invest a considerable effort to collect information about the users who register with them. This procedure can be an effective technique to prevent criminals from hosting their malicious pages on those providers.

Prevention - About 40% of the providers deployed some kind of security mechanism to block simple attacks, ranging from SQL injections to exploitation of common web application vulnerabilities.

Detection - Once the customer is registered, most of the providers do nothing to detect malicious activities or compromised websites - therefore providing very little help to their customers. We were surprised to discover that 21 out of the 22 tested providers did not even run an antivirus once per month (or they run them with old or insufficient signature sets) on the hosted websites. Moreover, none of them considered suspicious having multiple outgoing connection attempts towards an IRC server.

Abuse Notification - Only 36% of the providers reacted to our abuse notifications. When they promptly replied, most of the time their reaction was inappropriate or excessive. None of the *global* providers and only one of the *regional* ones were able to properly manage both the real and the fake complaints in a timely manner.

Security Services - The use of inexpensive security add-on services did not provide any additional layer of security in our experiments. Also the services that were configured to scan the content of our sites via FTP failed to discover the malicious files.

The main differences between *global* and *regional* providers appeared to be in terms of registration verification (in favor of *global* providers) and reaction to real complaints (in favor of *regional* ones).

As we already mentioned in the introduction of this paper, web hosting providers are in the position to play a key role in the security of the Web. In fact, they host millions of websites that are often poorly managed by unexperienced users, and that are likely to be compromised to spread malware and host phishing kits. Unfortunately, all the shared web hosting providers we tested in our study missed this opportunity.

6. ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n°257007.

7. REFERENCES

[1] VirusTotal - Free Online Virus, Malware and URL Scanner. https://www.virustotal.com/.

[2] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell. State of the art: Automated black-box web application vulnerability testing. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 332–345. IEEE, 2010.

[3] D. Canali and D. Balzarotti. Behind the scenes of online attacks: an analysis of exploitation behaviors on the web. In *Proceedings of the 20th Annual Network and Distributed System Security Symposium*, NDSS '13, Feb. 2013.

[4] Commtouch and StopBadware. Compromised Websites - An Owner's Perspective. http://stopbadware.org/pdfs/compromised-websites-an-owners-perspective.pdf, February 2012.

[5] W. de Vries. Hosting provider antagonist automatically fixes vulnerabilities in customers' websites. https://www.antagonist.nl/blog/2012/11/hosting-provider-antagonist-automatically-fixes-vulnerabilities-in-customers-websites, November 2012.

[6] fyicenter.com. Credit card number generator - test data generation. http://sqa.fyicenter.com/Online_Test_Tools/Test_Credit_Card_Number_Generator.php, 2010.

[7] J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi. deseo: combating search-result poisoning. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, pages 20–20, Berkeley, CA, USA, 2011. USENIX Association.

[8] J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi. Heat-seeking honeypots: design and experience. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 207–216, New York, NY, USA, 2011. ACM.

[9] Larry Ullman. Understand your hosting, five critical e-commerce security tips in five days. Peachpit Blog, 2011. http://www.peachpit.com/blogs/blog.aspx?uk=Understand-Your-Hosting-Five-Critical-E-Commerce-Security-Tips-in-Five-Days.

[10] T. Moore and R. Clayton. Examining the impact of website take-down on phishing. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, eCrime '07, pages 1–13, New York, NY, USA, 2007. ACM.

[11] T. Moore and R. Clayton. The consequence of non-cooperation in the fight against phishing. In *eCrime Researchers Summit, 2008*, pages 1 –14, oct. 2008.

[12] T. Moore and R. Clayton. Financial cryptography and data security. chapter Evil Searching: Compromise and Recompromise of Internet Hosts for Phishing, pages 256–272. Springer-Verlag, Berlin, Heidelberg, 2009.

[13] Number 7. osCommerce 'categories.php' Arbitrary File Upload Vulnerability, November 2010. http://www.securityfocus.com/bid/44995/info.

[14] OWASP foundation and TrustWave SpiderLabs. Owasp modsecurity core rule set project. https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project, 2012.

[15] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All your iframes point to us. In *Proceedings of the 17th conference on Security symposium*, SS'08, pages 1–15, Berkeley, CA, USA, 2008. USENIX Association.

[16] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. The ghost in the browser analysis of web-based malware. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, HotBots'07, pages 4–4, Berkeley, CA, USA, 2007. USENIX Association.

[17] N. Provos, M. A. Rajab, and P. Mavrommatis. Cybercrime 2.0: When the cloud turns dark. *Queue*, 7(2):46–47, Feb. 2009.

[18] B. Stone-Gross, M. Cova, C. Kruegel, and G. Vigna. Peering through the iframe. In *INFOCOM, 2011 Proceedings IEEE*, pages 411 –415, april 2011.

[19] webhosting.info. Country-wise top hosts. http://www.webhosting.info/webhosts/tophosts/Country/, 2012.