Prevention of Outbound Unsolicited Electronic Mail

Martynas Bendorius[1], Ingrida Lagzdinyte-Budnike[1], Kestutis Paulikas[1], Aurelijus Budnikas[2]
[1]*Department of Applied Informatics, Kaunas University of Technology,*
Studentu St. 50–402a, LT-51368 Kaunas, Lithuania
[2]*Department of Telecommunications, Kaunas University of Technology,*
Studentu St. 50–424, LT-51368 Kaunas, Lithuania
ingrida.lagzdinyte@ktu.lt

***Abstract*—A novel outbound unsolicited electronic mail prevention solution is presented, which can be used in shared web hosting environments. It combines several new ideas, including the rate of non-existent recipient email addresses, the content of the file which called the mail function and automatic actions after detection of suspicious activity to isolate and block it. A complete solution was designed and implemented, all the algorithms were described and presented in the article. The evaluation was done by testing the efficiency of outbound unsolicited electronic mail detection and the number of incidents blocked.**

***Index Terms*—Electronic mail; message systems; unsolicited electronic mail.**

I. INTRODUCTION

Even in 2015, unsolicited electronic mail, also known as SPAM, not just continues to plague businesses and users, but also accounts for a noticeable amount of traffic in global networks. According to an international software security group Kaspersky Lab [1], in Q2 2015, the proportion of spam in email traffic was 53.4 %.

A variety of different methods exist to send unsolicited electronic mail, however, one of the most popular ways is sending SPAM through existing vulnerable websites on web hosting servers. The reason behind this trend is the fact that 86 % of all websites have at least one serious vulnerability [2]. The report by WhiteHat Security also notes that most of the time the sites contain more than just a single vulnerability. A provider of cyber and data security products Imperva confirms WhiteHat indications in "Web Apllication Attack Report #5" [3] and additionally notes the increase of 24 % in Remote File Inclusion (RFI) attacks and WordPress as the most common target for Content Management System (CMS) attacks.

Today's world is mostly concentrated on incoming spam filtering solutions like the most popular open source solution SpamAssassin [4], greylisting, blacklists, SMTP-time filtering and research on spam filters. Even though they help lowering the amount of received spam in customer's mailboxes, not all of the messages are successfully identified as spam. Moreover, it does not change the amount of spam in global networks.

Manuscript received 21 December, 2015; accepted 30 May, 2016.

To filter spam, major email service providers check SPF [5] and DKIM [6] records [7], but that does not help if spam is sent from a PHP script in a web hosting account, configured with correct SPF and DKIM records. That is why attackers send SPAM from infected websites.

To lower the amount of spam in global networks and consequences caused, outbound spam prevention solutions must be used. In addition to, using outbound SPAM prevention provides other benefits for shared web hosting providers, including non-blacklisted IP addresses, good server reputation, less emails in mail queue, resulting in better quality of services for the end customer. Data centers usually have strict Terms of Service (ToS) for spam and take serious actions when they detect outbound spam from their network to save the reputation of their entire network. Actions may include null routing of the client's IP addresses [8] if outbound spam from the servers is detected, resulting in no availability of other services like HTTP, FTP or DNS, or a block for SMTP port 25 [9], resulting in no availability of SMTP service, which prevents delivery of normal email messages from shared web hosting servers.

II. TECHNIQUES TO PREVENT OUTBOUND SPAM

One of the most popular technique to prevent outbound spam on shared web hosting servers is rate limiting, offered by two of the most popular web hosting control panels cPanel [10] and Plesk [11]. The technique limits the number of email messages that could be sent in a particular amount of time for a specific email address or domain. It is effective with strict limits set, but has several disadvantages. Firstly, if a customer reaches the email rate limit set without sending any spam messages, all the additional outgoing messages will fail and the customer will be blocked from sending more mails for a particular amount of time. Secondly, when the period of time expires for the limit, the customer is able to send the same amount of messages again, including the spam messages. For example, if the rate limit is set to 50 messages per hour, when the hour limit expires, the customer can send another 50 messages. Thirdly, even if administrators are notified about customers who reached the limit, they do not indicate if that was a spam incident or not, and requires system administrators to review the incident manually.

Another approach offered by cPanel is outbound mail

filtering using Apache SpamAssassin [12], however due to high resource usage, many false positives and insufficient protection they do not suggest using it on production servers.

III. PROPOSED SOLUTION

Considering the experience in shared web hosting sphere and knowing most common ways used to send unsolicited electronic mail, we are offering a complete solution which automatically detects and blocks the attack. Our goal is to block email delivery from infected websites. The solution consists of several separate ideas: A) limiting the number of message submits to nonexistent email addresses per hour, B) analyzing the PHP scripts which were used to send mails. Both of the solutions ensure blocking further attempts to send mail from affected accounts or scripts directly in MTA.

A. Rate Limiting of Attempts to Send Emails to Nonexistent Mail Addresses

Attackers usually use a large list of email addresses having a number of email addresses which never existed or do not exist anymore. The idea of rate limiting of attempts to send emails to nonexistent recipient email addresses is not new, it has been discussed on the Internet in 2010 by Andrew Hearn, CEO of Andrews & Arnold Ltd, providing email services for the masses, and some other places. However, they focus on authenticated SMTP users, and no complete solution was found for emails sent from PHP scripts on shared web hosting environments.

Our proposed solution allows to set custom rate limiting period and the number of nonexistent recipients, that way shared web hosting companies can tune the settings up for them to be stricter or more tolerant. In our further experiments the default limit was set to delivery of 100 emails to nonexistent email addresses in 1 hour until the delivery from a specific path gets blocked. The default limit was set to 100, because from various tests we have got some false-positives with a lower number, mainly from electronic shops having fake registrations for newsletters.

When the limit is reached, the full script execution path is added to the list of denied paths file automatically, the end-user (legal shared web hosting account owner) and system administrators get notified about the incident. Without a manual action from administrators or the end customers it is not possible to unlock the path, so attackers cannot send more emails even when the rate limiting period (1 hour) ends.

Recipient verification is based on SMTP callback verification offered by Mail Transfer Agents (MTA) [13]. In our case Exim was used as an MTA. To verify a sender address it connects to a remote SMTP server (set in DNS MX records), executes a standard HELO command required by the protocol, submits an empty MAIL FROM field and fills the RCPT TO field with the address to be tested, then it quits with a QUIT command. The remote server response to the RCPT command is a 2xx code meaning that verification succeeded, or a 5xx code, which means that verification failed. For any other condition the next host set in DNS MX records is tried (if there are set any). A successful callback does not guarantee a successful delivery, but a failing callout guarantees that delivery would fail. To save the resource usage by address verification, Exim caches the result of callbacks. The solution is represented in Fig. 1.

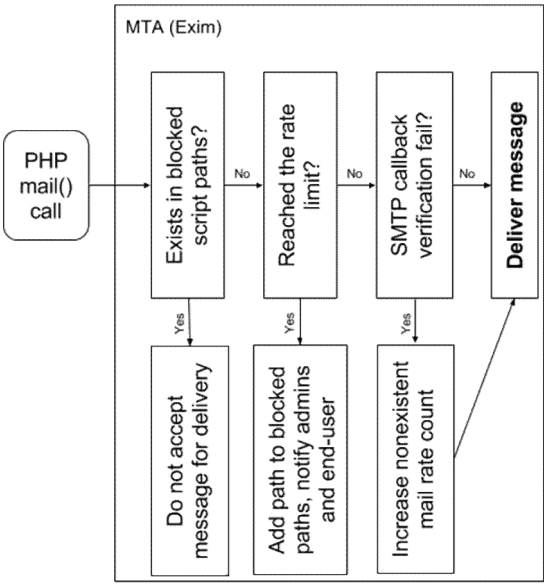

Fig. 1. Rate limiting solution for non-existent mails scheme.

The proposed solution uses a time limit of 10 seconds for a timeout and a defer_ok setting [14], which makes the check to treat a failure of contacting any host or any other kind of temporary problem as success by the ACL. The structure of the callback (also known as callout) is represented in Fig. 2.

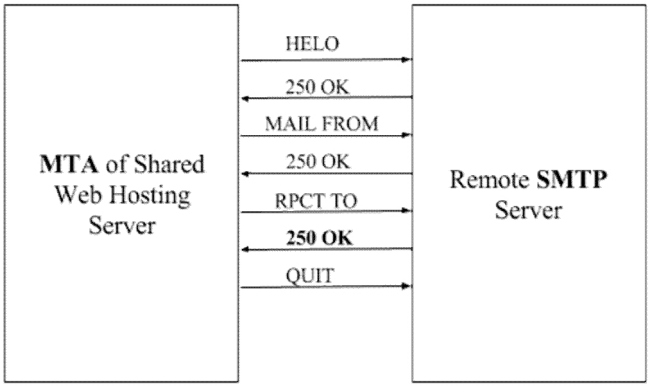

Fig. 2. SMTP callback scheme with a correct recipient.

As seen in Fig. 2., the response in bold after RPCT TO command returned 2xx code, meaning a verification success. The proposed solution only counts the email addresses that return code 5xx, meaning a verification failure.

B. Identification of Malicious PHP Scripts

The previously described solution reflects only a part of the complete solution. Even though the previous solution stops many of the spam attempts, it still can be complemented. Attackers still have a way to send up to 100 unsolicited electronic mails from a directory on the system.

To reduce the number of mails sent a PHP code analysis of scripts which used the *mail()* function can be done. By analyzing spam scripts on a shared web hosting server, it was noted that most of the infected scripts use an encoded PHP code inside. In such case a *base64_decode()* PHP function is called to decode the code and *eval()* PHP language construct is used to execute the code. 5264 virtual

hosts on different dedicated servers were scanned to see if there are any occurrences of *mail()*, *base64_decode()* and *eval()* usage in the same PHP script and it was found that none of normal PHP scripts had this combination used in a single PHP file.

Despite the fact PHP allows simply disabling PHP functions in PHP configuration file, *base64_decode()* and *eval()* are used in normal scripts too, so they could not be simply disabled, because that could break a number PHP applications on shared web hosting servers. In addition to, *eval()* is a language construct, so it is not possible to disable it in disable_functions PHP configuration setting.

To effectively detect scripts having *eval()* and *base64_decode()* in the same file, which was used to send mails using *mail()*, log scanning must be used. The reason behind that – the files are encoded using *base64_encode()* and that allows attackers to hide a *mail()* function call in their PHP scripts. However, PHP offers live logging setting for the usage *mail()* function [15], and this feature allows us to know which PHP script called the *mail()* function, then it can be simply scanned for the usage of *base64_encode()* function and *eval()* language construct. We cannot simply say that a PHP script, which has it's source code encoded and *mail()* function hidden, is a malicious script, because it is possible to encode PHP source code using ionCube or Zend encoders, and they help commercial solutions to legally hide their PHP source code.

Our proposed solution uses an application named Swatchdog (Simple Log Watcher) [16] to scan the logs in real time and parse the scripts having the *mail()* calls. If the file is detected to have *base64_decode()* and *eval()* inside, it's added to the list of denied paths and administrators together with the end-user are notified as shown in Fig. 3.

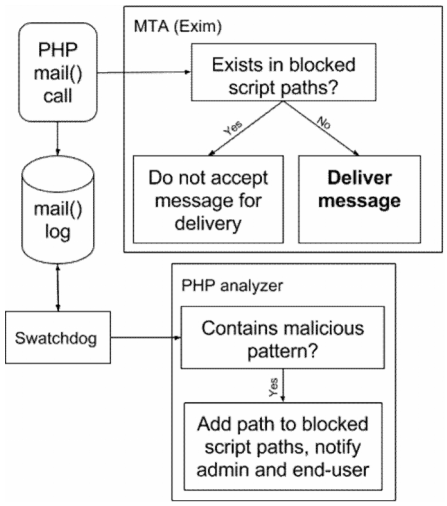

Fig. 3. PHP code analysis scheme.

As seen in Fig. 3, when PHP mail() function call happens, it instantly logs the usage of the function and submits the mail to MTA at the same time. This is the main reason why we cannot technically stop the first mail from being sent, because MTA and Swatchdog work independently from each other. However, Swatchdog checks the log in real time and passes the full path to the script, which analyses PHP code. If the script detects *eval()* and *base64_decode()* function in the script, which was used to sent the mail, it adds the path to the list of paths that are not allowed to send any mails from the system. The second try to send a mail using PHP will fail, because MTA will detect that the path is not allowed to send the mail and will refuses to deliver it.

When the automatic detection and blocking of spam delivery was started, a few more patterns used to hide PHP files were found and added to automatic blocking script:

– More than 10 occurrences of variable GLOBALS, *eval()* function called in the same file, *mail()* function hidden;

– GLOBALS, substr, return used on the same line, *mail()* function hidden;

– More than 10 occurrences of *chr()* function, *eval()* function called in the same file, *mail()* function hidden.

Even though the regular expression list for detection of malicious PHP scripts detected all the files that were used to send unsolicited electronic mail, we are sure that more patterns exist and even more ways to hide PHP code will be created in the future. The main drawback of this method is the need to constantly update the static list of regular expressions for the detection of malicious PHP scripts that are used to send spam.

Most of the time content management systems (CMS) have folders that are used for user-level file uploads. These folders are a common target for unrestricted file upload attacks [17]. To partially solve the main drawback of the method mentioned above, a global blocked script paths file can be used. That way MTA would check if the path from which the mails are sent has no matches in the regular expression list of paths. If a match is detected, delivery would be blocked. For example, it would be safe to block email sends with the following regular expressions for WordPress CMS:

– ^.*/wp-content/cache.*

– ^.*/wp-content/uploads.*

IV. EVALUATION

To evaluate the solution, both of the spam prevention solutions were tested separately.

Our qualitative characteristics for evaluation criteria were:

– detection of malicious attempts to send spam;

– number of false-positive detections.

The rate limiting of sends to non-existent email addresses solution was installed to a random production server having 688 shared web hosting accounts and 1801 domains setup on it. The solution was installed and left for a week to run. Then a file having the list of blocked paths was checked to see when and what was blocked.

In a period of 7 days, 10 attempts to send spam were blocked from 3 different web hosting accounts. That accounts for 0.44 % of all the customers. The results are shown in Table I. From the directory names seen in directory column we can see that malicious files are usually placed deeply in subdirectories, so that it would be harder to detect them for the end-customer. In addition to, the name of the infected files is usually set to confuse the end-customers too, because file names like css.php or blog.php do not look strange, and it could be believed that the files are provided by the content management system or it's plugins. The date

shown in the table indicates when the spam attempt was detected and blocked, to evaluate the quantity of scripts blocked in a period of time. Real domain names were changed in the table for confidentiality purposes.

TABLE I. LIST OF BLOCKED PATHS.

Directory	Date	Infected file
/home/gertex/domains/secretdomain.pl/public_html/wp-content/uploads/wow-slider-plugin/9/tooltips	02 Nov 2015 12:08	css.php
/home/gertex/domains/secretdomain.pl/public_html//produkty	03 Nov 2015 03:46	blog.php
/home/technogrf/domains/secretdomain2.pl//public_html/wp-includes/pomo	03 Nov 2015 04:30	plugin74.php
/home/technogrf/domains/secretdomain2.pl//public_html/wp-includes/js/swfupload	03 Nov 2015 23:29	ini.php
/home/technogrf/domains/secretdomain2.pl//public_html/wp-admin/images	04 Nov 2015 20:34	global76.php
/home/gertex/domains/secretdomain.pl//public_html/wp-includes/js/mediaelement	05 Nov 2015 01:15	page26.php
/home/gertex/domains/secretdomain.pl/public_html//wp-content/uploads/2015/04	06 Nov 2015 08:09	test.php
/home/champion/domains/secretdomain3.pl/public_html/zarzadzanie	06 Nov 2015 14:42	error65.php
/home/gertex/domains/secretdomain.pl//public_html/cgi-bin	07 Nov 2015 03:31	error94.php
/home/technogrf/domains/secretdomain2.pl//public_html/wp-content/plugins/ubermenu/standard/styles	07 Nov 2015 06:19	help15.php

All of the blocked paths had malicious content inside and there were no false positives. That indicates that the protection works and that the threshold of 100 sends to non-existent emails per-hour could be lowered to catch more spam attempts. 1 attempt to spam of 11 (9.1 %) used no *base64_decode()* and *eval()* in their PHP code. Even though it is a low amount, it is usually enough for the server to get blacklisted and for the actions to be taken from the data center (null route the IP address or block SMTP port). The fact confirms that there is a need of a complete combined solution, which detects both the usage of malicious PHP scripts and rate limit of sends to non-existent email addresses.

The identification of malicious PHP scripts method was tested on a separate server to get the efficiency results without any influence from the previous detection method. The server had 359 active shared webhosting accounts owning 1469 domains. Test results were also taken from a time period of a week. It was found that the spam affected 3 different customers and the spam attack was blocked. That accounts for 0.84 % of all the customers on the server. The results from blocked paths file are shown in Table II.

TABLE II. LIST OF BLOCKED PATHS.

Directory	Date	Infected file
/home/tagmaler/domains/secretdomain2.dk//public_html/wp-content/plugins/codestyling-localization	02 Nov 2015 15:19	ini.php
/home/vokalind/domains/secretdomain.dk//public_html/wp-content/themes/enfold/config-layerslider/LayerSlider/img	03 Nov 2015 08:37	help.php
/home/vokalind/domains/secretdomain.dk//public_html/wp-content/themes/twentythirteen/inc	03 Nov 2015 18:24	code25.php
/home/bedsteci/domains/secretdomain.info//public_html/libraries/joomla/session	04 Nov 2015 22:13	default.php

The identification of malicious PHP scripts method, which complements the first method of rate limiting sends to non-existent email accounts, also had no false positives and malicious scripts were found in all the paths reported.

Administrators and end-customers were successfully informed about the attempts to send unsolicited electronic mails from affected websites. Instructions for unblocking the path were also provided. One of the notifications is shown in Fig. 4. It is beneficial to inform the end-customer about the incident and allow them to unblock themselves, because they can clean their websites up by themselves, upgrade their content management system and plugins used with it, without any actions done from system administrators. In very rare cases it could be the customers themselves responsible for the spam, however, such cases can be quickly identified, because administrators would repeatedly receive messages from the system about the spam originating from the customer.

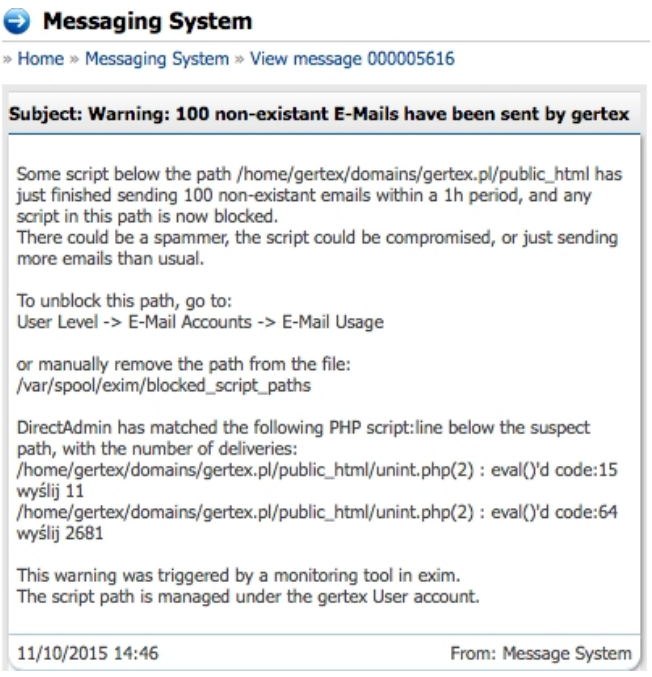

Fig. 4. Notification about suspicious activity under account.

The efficiency of our proposed solution was also compared with rate limiting – the most widely used outbound unsolicited mail prevention method. Comparison was done by setting the rate limit to 10 emails/minute and checking the efficiency of every method by sending different amount of emails every minute. We randomly added 1/3 non-existent email addresses to the list of recipients. The results are seen in Fig. 5.

As seen in Fig. 5 above, rate limit method has the advantage of preventing the outbound spam when it is sent from unrecognized PHP script and the amount of spam

messages is low. That way the threshold of non-existent email addresses per hour in our proposed solution is not reached. In all the other cases our proposed solution behaved equally or better than the rate limit method.

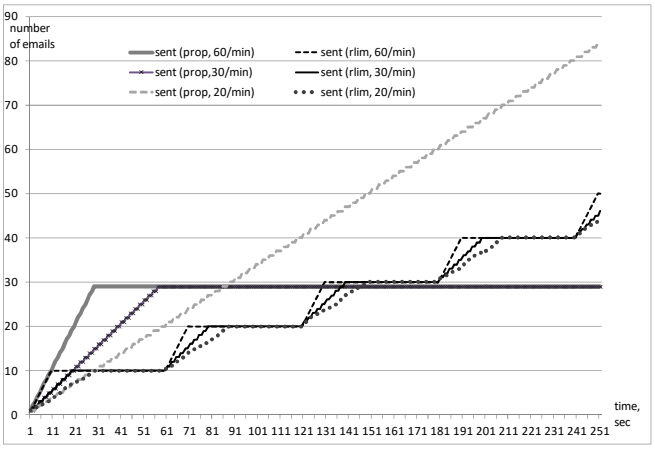

Fig. 5. Comparison of proposed solution and rate limit method.

V. CONCLUSIONS

Unsolicited electronic mail is still an actual world-wide problem, having a proportion of 53.4 % in global email traffic. With a growth of the number of unsolicited electronic mails from infected websites, there is an urgent need to replace a widely used rate limiting solution, which simply describes how many emails can be sent in a particular amount of time, because of the problems faced by shared web hosting end-customers.

In this work a novel solution to identify and prevent outbound spam was offered, which ensures more modern protection. A complete solution was created and used for the evaluation of the product on production servers with a success.

Experimental results have shown no false-positive reports and an accurate detection of malicious scripts used to send unsolicited electronic mails in shared web hosting servers. Even though the solution successfully blocked spam attempts and prevented the servers of getting blacklisted, that indicates that stricter rate limits for email sends to non-existent email addresses could be chosen, according to the environment. In addition to, it might be worth combining conservative rate-limit method with the proposed solution to prevent even more outbound unsolicited mail in situations with unrecognized PHP scripts and high-quality spam lists with minority of non-existent email addresses.

REFERENCES

[1] Kaspersky Lab, "Spam and phishing in Q2 2015". [Online]. Available: https://securelist.com/analysis/quarterly-spam-reports/71759/ spam-and-phishing-in-q2-of-2015/

[2] WhiteHat, "Website security statistics report". [Online]. Available: https://www.whitehatsec.com/statistics-report/featured/2015/05/21/statsreport.html

[3] Imperva, "Web application attack report #5". [Online]. Available: http://www.imperva.com/docs/hii_web_application_attack_report_ed5.pdf

[4] Apache Software Foundation, "What Apache SpamAssassin is". [Online]. Available: http://svn.apache.org/repos/asf/spamassassin/branches/3.4/README

[5] SPF Council, "What is SPF". [Online]. Available: http://www.openspf.org/FAQ/What_is_SPF

[6] D. Crocker, "DKIM frequently asked questions". [Online]. Available: http://www.dkim.org/info/dkim-faq.html

[7] Google, "Authenticate email with DKIM". [Online]. Available: https://support.google.com/a/answer/174124?hl=en

[8] Hetzner, "Terms and conditions". [Online]. Available: https://www.hetzner.de/es/hosting/legal/agb

[9] OVH, "Specific Terms and Conditions on the rental of a dedicated server". [Online]. Available: https://www.ovh.com/us/support/termsofservice/Special_conditions_for_dedicated_server.pdf

[10] cPanel Inc., "How to prevent email abuse". [Online]. Available: https://documentation.cpanel.net/display/CKB/How+to+Prevent+Email+Abuse

[11] Parallels Inc., "Protection from outbound spam". [Online]. Available: http://download1.parallels.com/Plesk/Doc/en-US/online/ plesk-administrator-guide/index.htm?fileName=71349.htm

[12] cPanel Inc., "Scan outgoing mail". [Online]. Available: https://documentation.cpanel.net/display/ALD/Scan+Outgoing+Mail

[13] University of Cambridge, "Exim callout verification". [Online]. Available: http://www.exim.org/exim-html-current/doc/html/spec_html/ch-access_control_lists.html#SECTcallver

[14] University of Cambridge, "Exim additional parameters for callouts". [Online]. Available: http://www.exim.org/exim-html-current/doc/html/spec_html/ch-access_control_lists.html#CALLaddparcall

[15] The PHP Group, "Runtime configuration". [Online]. Available: http://php.net/manual/en/mail.configuration.php

[16] Todd Atkins, "Simple Log Watcher". [Online]. Available: http://sourceforge.net/projects/swatch/

[17] The Sans Institute, "Web application file upload vulnerabilities". [Online]. Available: https://www.sans.org/reading-room/whitepapers/testing/web-application-file-upload-vulnerabilities-36487