Active Intrusion Management for Web Server Software: Case WordPress

Patrik Paarnio
Department of Business Management and Analytics
Arcada University of Applied Sciences
Helsinki, Finland
e-mail: patrik.paarnio@gmail.com

Sam Stenvall
Department of Business Management and Analytics
Arcada University of Applied Sciences
Helsinki, Finland
e-mail: sam.stenvall@nordsoftware.com

Magnus Westerlund
Department of Business Management and Analytics
Arcada University of Applied Sciences
Helsinki, Finland
e-mail: magnus.westerlund@arcada.fi

Göran Pulkkis
Department of Business Management and Analytics
Arcada University of Applied Sciences
Helsinki, Finland
e-mail: goran.pulkkis@arcada.fi

***Abstract*—Methods for active management of intrusion attacks against WordPress web sites are proposed for improved real-time web security. Intrusion management is defined to be active when both intrusion responses and forensic investigations are proactive and/or automatically triggered by intrusion attacks. Booby traps as active defense against intrusion attacks using return-oriented programming and other related research is briefly surveyed. Active intrusion management techniques such as booby trapped patches to publicly known vulnerabilities in WordPress plug-ins and redirection scripts for WordPress plug-ins are proposed. Experimentation results with proposed booby trapped patches and proposed redirection scripts are presented and evaluated.**

Keywords – active intrusion prevention; active intrusion detection; web site vulnerability; WordPress vulnerability; booby trap.

I. INTRODUCTION

A fully secure network must be able to resist any type of intrusion attack and all vulnerabilities in the network must be eliminated, while it is sufficient for an attacker to find only one network weakness. Current defense methods, such as firewalls, antivirus software, and intrusion detection systems (IDS) cannot prevent all types of intrusion attacks. Most current defense methods react rather passively on intrusion attacks with intrusion alert messages to a human network administrator or to a computer doing network administration. Thus, a current IDS can often only detect occurred intrusion and related network damage. Forensic information of an intrusion attack can also usually only be traced afterwards from log files and from other system state changes caused by the attack.

Securing or hardening Content Management Systems (CMS) has become a struggle for web site administrators. The exploitation of CMS systems such as Joomla and WordPress is extraordinarily easy due to rapid development of plug-ins for the systems, bad software engineering practices (e.g., lack of quality assurance for plug-ins), and the ease of use (in-depth technical skills are not required for installing and using the software). This allows a potential attacker to scan for public installations and their corresponding vulnerabilities with minimal risk to be detected as a threat. We consider current passive intrusion management methods often too limited in ability to secure installations.

Intrusion management is defined to be active when both intrusion responses and forensic investigations are proactive and/or automatically triggered by intrusion attacks. This paper presents active intrusion management of web server software based on WordPress. We present using an exploratory case study methodology for developing an understanding of the underlying system deficiencies. This method allows us to gain deeper insights into chains of cause and effect, in the specific software, to answer the research question of how to improve security in WordPress through active intrusion management.

The paper is organized as following; we start with related research by introducing return-oriented programming as a method for both performing attacks and as a defense mechanism for implementing “booby traps”. The third section then develops an analogy for open source web software based on a case study for WordPress. In the fourth section we report preliminary research results, before concluding in the final section.

II. RELATED RESEARCH

Prevention, detection, and responding to intrusion attacks has for many years been an important research topic. Intrusion responses are created through notification, manually, and automatically. Four desirable features of an ideal intrusion response system have been proposed: automated responses, proactivity, adaptability, and cost efficiency [1].

Active intrusion defense is based on automated intrusion responses. In [2], it is proposed an intrusion management system based on intelligent decision making agents invoking response executables and scripts for different intrusion attack types. Active defense based on distributing new access control policies to firewall nodes in a network once intrusion is detected is presented in [3].

 ISBN: 978-1-61208-432-9

Active defense called honey-patching against attempt to exploit network software vulnerabilities is proposed in [4]. A honey-patch redirects attacks to an unpatched decoy, which collects relevant attack information and also allows attacks to succeed in order to deceive attackers.

A Linux distribution based on Ubuntu LTS, the Active Harbinger Distribution, includes many preinstalled and configured tools for active defense against malicious activity such as network scanning and connecting to restricted services. The functions of these tools "range from interfering with the attackers' reconnaissance to compromising the attackers' systems. [5]

In a guide to intrusion detection and prevention systems, automated intrusion attack responses are characterized as a technique, which "can respond to a detected threat by attempting to prevent it from succeeding". Such intrusion attack responses can

- stop the intrusion attack by terminating the network connection or user session being used by the attack or by blocking all access to the target of the attack,
- change the security environment of the target of the intrusion attack, for example by reconfiguring a firewall or a router or by applying a patch to a vulnerability exploited by the attack, or
- change the intrusion attack process from malicious to benign, for example by removing malicious file attachment from e-mail messages before they reach their recipients [6].

Active defense called "booby trapping" against code-reuse intrusion attacks based on return-oriented programming (ROP) [7] is presented in [8].

A. *Return-Oriented Programming (ROP)*

In a ROP attack the attacker takes over program flow control in a network connected computer without injection of malicious program code. ROP gadgets i.e., short instruction sequences terminating with a RETN assembly instruction (return from procedure) are linked together from the control stack. RETN fetches the return address from the control stack, which is manipulated in a ROP attack.

A ROP attack requires some buffer overflow vulnerability. The attack starts with injection and execution of program code, which overwrites a return address of a RETN instruction on the control stack. The resulting execution of RETN is a jump to another gadget selected by the attacker. The ROP attack is implemented by execution of a gadget chain. [7][9]

A ROP attack can succeed only if two preconditions are fulfilled: the flow control of a program must be acquired and in the program there must be gadgets, which can be linked together in an attack. A ROP attack is prevented when at least one precondition is absent. Elimination of all buffer overflow possibilities in a program prevents a ROP attack to start. Replacement of all RETN instructions in a program with other subprogram return instructions prevents a ROP attack to proceed. An indirect defense against ROP attacks is Address Space Layout Randomization (ALSR), in which instruction addresses in the program are changed. Then, a ROP attack cannot find needed gadgets at expected addresses, but can still search needed gadgets with brute force methods [10]. To detect an ongoing ROP attack the RETN execution frequency can be monitored to issue an alert if a preset threshold value is exceeded [11] [12].

B. *Booby Trapping*

Booby trapping software is active defense based on ROP functionality against intrusion attacks using ROP. A booby trap is program code inserted into a computer program as binary gadget changes at compile time or at load time of binaries in such a way that the functionality of the program remains unchanged. A booby trap can thus be executed only by ROP attacks against the changed program. A booby trap is program code and cannot therefore be deactivated by an intrusion attack. The program code of a booby trap can send an alert to a network administrator, register the IP address of the intrusion attack source, redirect the attack to a honeypot, even launch a counterattack, etc. [8]

Insertion of booby traps at compile time requires access to the original source code of a program. Insertion of booby traps at load time is possible using suitable machine code jump instructions without access to the original source code. Booby traps can be inserted at binary addresses of exploitable gadgets or randomly to possibly trap intrusion attacks, which scan programs to find exploitable gadgets. Figure 1 illustrates binary code changes with inserted booby traps in a program [8].

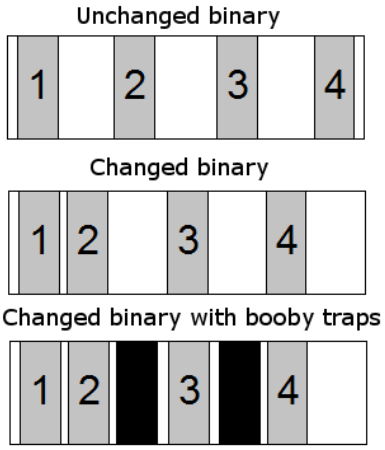

Figure 1. Binary changes with insertion of two booby traps in a program with four exploitable gadgets.

III. ACTIVE INTRUSION MANAGEMENT FOR WEB SERVERS – CASE WORDPRESS

The basic concept of ROP attacks does generally not apply to web applications. While ROP attacks against the Apache web server itself have been implemented and evaluated [13], protection against such attacks will not prevent attacks against web server code. Web server application code is easily booby trapped by modifying the source code, since the code is generally not pre-compiled.

A WordPress [14] installation on Apache or on some other web server platform has several known vulnerabilities, which have been patched. It can also be considered highly likely that there are several still publicly unknown zero-day vulnerabilities prone to intrusion attacks. The reasons of the high vulnerability of WordPress is the modular software architecture with a multitude of

 ISBN: 978-1-61208-432-9

possible, potentially vulnerable plug-ins and the open source code, which anyone can examine to find exploitable vulnerabilities. The large global WordPress user base is also a stimulating feature for intrusion attacks against WordPress installations.

Program vulnerability patches can be booby trapped to trigger collection of forensic information about intrusion attack attempts based on available exploits. Collected forensic information can be used to create proactive responses to possible future intrusion attacks, for example by blacklisting source IP addresses related to detected use of exploits. Attempts to exploit patched or even publicly unknown vulnerabilities in non-existing (i.e., not installed) plug-ins can be redirected to honeypots or to sandbox environments where responses and/or forensic investigations are automatically triggered. An Internet connected WordPress installation with patched vulnerable plug-ins on an Apache web server has been used in booby trapping experiments described in this chapter

A. Booby Trapped WordPress Vulnerabilities

Vulnerabilities are patched before being booby trapped. A potential intruder shouldn't know that vulnerabilities have been patched [4]. The vulnerable version of a module is used with manual source code changes. Source code comparison of the vulnerable version with the patched version shows how the source code should be changed. A booby trap to register forensic information in a text file [15], for example logging IP addresses of intrusion attack attempts, is included in the beginning of the changed source code. The PHP code of such a booby trap is seen in Figure 2 and is denoted in later examples by booby_trap().

1) WordPress Wp Symposium 14.11:

The vulnerable file UploadHandler.php (see Figure 3) in the plug-in WordPress Wp Symposium 14.11 accepts for upload files of any type, which means that a malicious shell code file can be uploaded.

In [16], an exploit script is published, which creates a backdoor to protected files on a WordPress site with an uploaded shell code.

```
$ipadress = $_SERVER['REMOTE_ADDR'];
$webpage = $_SERVER['SCRIPT_NAME'];
$browser = $_SERVER['HTTP_USER_AGENT'];
$file = 'attack.log';
$fp = fopen($file, 'a');
$date = date('d/F/Y h:i:s');
fwrite($fp, $ipadress.' - ['.$date.'] '.$webpage.' '.$browser."\r\n");
```

Figure 2. Booby trap code.

```
class UploadHandler {
  …
                        'inline_file_types'=>
  '/ \.(mp4|zip|doc|docx|ppt|pptx|xls|xlsx|txt|pdf|gif|jpe?g|png)$/i',
                    'accept_file_types'=> '/.+$i',
  …
}
```

Figure 3. The vulnerable UploadHandler.php.

```
class UploadHandler {
  …
 booby_trap ( );
  'inline_file_types'=>
'/ \.(mp4|zip|doc|docx|ppt|pptx|xls|xlsx|txt|pdf|gif|jpe?g|png)$/i',
  'accept_file_types'=>
'/ \.(mp4|zip|doc|docx|ppt|pptx|xls|xlsx|txt|pdf|gif|jpe?g|png)$/i',
  …
}
```

Figure 4. The patched UploadHandler.php with an inserted booby trap.

The exploit is a Python script, which is tested before the plug-in is booby trapped. The exploit script gives the name and path of the backdoor if the upload succeeds. Using the backdoor, the file passwd can be retrieved with the command *?cmd=cat+/etc/passwd.*

Shell code upload is prevented by applying the patch shown in Figure 4. Attempts to upload files of unpermitted types create error messages after patching. A booby trap in the beginning of the patch code logs the IP addresses of attempts to exploit the patched vulnerability.

2) WordPress Shopping Cart 3.0.4:

The file banneruploaderscript.php in the plug-in WordPress Shopping cart 3.0.4 should check that only a logged in administrator is allowed to upload files to a WordPress site. However, any logged in user is allowed to upload files since in the condition of the if-statement is an *or-clause* instead of an *and-clause* (see Figure 5).

The published exploit script in [17] is a web form, which uses banneruploaderscript.php to upload files into the folder ./wp-content/plugins/wp-easycart/products/banners/ of a WordPress site. Upload of files of any type is allowed since banneruploaderscript.php trusts administrators and therefore an attacker is allowed to upload malicious files.

The vulnerability is patched by changing the *or-clause* to an *and-clause* in the condition of the *if-statement* in banneruploaderscript.php. After patching, a booby trap, which logs the IP addresses of attempts to exploit the patched vulnerability, is inserted (see Figure 6).

```
$userresult = mysql_query($usersqlquery);
$users = mysql_fetch_assoc($userresult);
if ($users || is_user_logged_in( )) {
  $filename = $_FILES[“Filedata”][“name”];
  $filetmpname = $_FILES[“Filedata”][“tmp_name”];
  $fileType = $_FILES[“Filedata”][“type”];
  $fileSizeMB = ($_FILES[“Filedata”][“size”] / 1024 / 1000);}
```

Figure 5. The vulnerable banneruploadscript.php.

```
$userresult = mysql_query($usersqlquery);
$users = mysql_fetch_assoc($userresult);
booby_trap( );
if ($users && is_user_logged_in( )) {
  $filename = $_FILES[“Filedata”][“name”];
  $filetmpname = $_FILES[“Filedata”][“tmp_name”];
  $fileType = $_FILES[“Filedata”][“type”];
  $fileSizeMB = ($_FILES[“Filedata”][“size”] / 1024 / 1000);}
```

Figure 6. The patched banneruploadscript.php with an inserted booby trap.

```
function mfbfw_admin_options( ){
 $settings = get_option('mfbfw');
 if ( isset($_GET['page']) && $_GET['page'] ==
  'fancybox-for-wordpress' ) {
  if ( isset($_REQUEST['action']) && 'reset' ==
   $_REQUEST['action']) {
   $settings = stripslashes_deep($_POST['mfbfw']);
   $settings = array_map('convert_chars', $settings);
   update_option( 'mfbfw', $settings );
   wp_safe_redirect( add_query_arg('reset', 'true') );
   die; }}}
```

Figure 7. The vulnerable function mfbfw_admin_options.

```
jQuery("a.fancybox").fancybox({
…
'padding': </script><script>alert(Owned by someone) </script>,
…
});
```

Figure 8. An injected piece of JavaScript.

3) Fancybox for WordPress:

The plug-in Fancybox for WordPress 3.0.2 has a cross-site scripting (XSS) vulnerability in the function mfbfw_admin_options (see Figure 7) in the file fancybox.php. This function doesn't validate input data and therefore permits script injection. An attacker can send the script with a web form. The script is injected on the body of a web site and is always triggered when this web page is browsed.

This vulnerability permits arbitrary JavaScript to be injected and executed on every page where Fancybox is used. The vulnerability stems from the fact that the function mfbfw_admin_options doesn't perform the necessary checks on the POST data, which allows a malicious user to inject arbitrary code into the settings for the Fancybox plug-in. The injected script is visible in the source code of the WordPress site (see Figure 8).

```
function mfbfw_admin_options( ){
 $settings = get_option('mfbfw');
 booby_trap( );
 if ( isset($_GET['page']) && $_GET['page'] ==
  'fancybox-for-wordpress' ) {
  if ( isset($_REQUEST['action']) && 'reset' ==
      $_REQUEST['action'] &&
      check_admin_referer( 'mfbfw-options-options')){
      $defaults_array = mfbfw_defaults( );
      update_option( 'mfbfw', $defaults_array );
      wp_safe_redirect( add_query_arg('reset', 'true') );
      die; }}}
```

Figure 9. The patched function mfbfw_admin_options with an inserted booby trap.

```
function wpdm_ajax_call_exec( ){
 if (isset($_POST['action']) && $_POST['action'] ==
  'wpdm_ajax_call'){
  if (function_exists($_POST['execute']))
   call_user_func($_POST['execute'], $_POST);
  else
   echo "function not defined!";
 die( );}}
```

Figure 10. The vulnerable function wpdm_ajax_call_exec.

```
function wpdm_ajax_call_exec( ){
booby_trap( );
   if (isset($_POST['action']) && $_POST['action'] ==
            'wpdm_ajax_call') {
            if ($_POST['execute']=='wpdm_getlink')
                     wpdm_getlink( );
            else
                     echo "function not defined!";
            die(); }}
```

Figure 11. The patched function wpdm_ajax_call_exec with an inserted booby trap.

In the patched source code the input data is validated by checking the source of script injection request (see Figure 9). A booby trap to log attempts to exploit the patched vulnerability is inserted before the if statement.

```
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule .* wp-boobytrap.php
```

Figure 12. Rewrite rules for redirecting requests for missing resources.

4) WordPress Download Manager 2.7.4;

In December 2014 a serious vulnerability in WordPress Download Manager was reported [18]. This vulnerability, for which an exploit script is published in [19] permits remote execution of program code. The script exploits the vulnerable function wpdm_cajax_call_exec (see Figure 10) in the file wpdm-core.php. The function wpdm_cajax_call_exec receives functions sent by a user from the graphical user interface and executes these functions without verification about their existence in the program. This means that an attacker can inject WordPress functions for execution in the program. The exploit script in [19] injects the WordPress function wp_Insert_user, which creates web site users for inputted user names, passwords, and user roles. An attacker can therefore create an administrator for a vulnerable WordPress site.

The functionality of the Python exploit script for WordPress DownLoad Manager 2.7.4 is tested on a vulnerable WordPress site. The WordPress site address is a parameter of the script. After successful script execution, the user information of the created administrator, which has been registered in the database of the WordPress site, is shown.

The patched source code shown in Figure 11, now permits execution of a function only if it exists in the program. A booby trap to log attempts to exploit the patched vulnerability is inserted in the patched source code. An attempt to exploit the patched vulnerability also returns an error message.

B. Redirecting Bad Requests to a Booby Trap

Manually booby trapping all plug-ins used on a typical WordPress installation requires much manual labor. There are simply too much possible vulnerabilities to patch, and it also makes the update process more complicated as the booby traps have to be reapplied after every plug-in update. Since booby trapping plug-ins using this approach requires the attack vector to be known (in order to insert the booby trap at the right location) it does not offer any protection against zero-day attacks.

 ISBN: 978-1-61208-432-9

We have studied an alternative approach that is based on two novel ideas. The first is to redirect requests for missing resources (e.g., files belonging to plug-ins that are not installed) to a special script which handles the requests. This script acts as the booby trap and can be made to do different things depending on the objective. During our testing we have configured it to log the requested URLs together with certain request parameters such as the query string and eventual POST data. If the objective would be to gather as much forensic data about potential intrusion attempts as possible the script could be programmed to emulate known exploits in order to make the attacker believe he actually succeeded. This type of emulation is often necessary since many exploits first attempt to detect whether the actual exploit would succeed or not; the payload itself may not be delivered if the detection fails.

Redirecting requests for missing files to an arbitrary request handler is not a new idea. It is used by many web frameworks, for example the Yii framework [20] and Fat-Free Framework [21], to force requests to go through the main index file of the web application itself. The same method is used here (see Figure 12), but for a different purpose.

By redirecting bad requests many types of attacks against a WordPress installation can be avoided. Even though the attacks caught by the redirection wouldn't have succeeded anyway (since the requested resource would not have been found) we have the opportunity to prevent further attacks (some of which may actually succeed) since we now can classify the IP address that made the request as malicious. This defense mechanism can thus be made a proactive part of the server's security system if it is used to automatically reconfigure the server's firewall to block further connection attempts from the implicated IP addresses.

```
# mod_substitute
AddOutputFilterByType INFLATE;SUBSTITUTE;DEFLATE
text/html text/javascript
Substitute "s|\?wpdmdl|\?fake-wpdmdl|i"
# mod_rewrite
RewriteEngine On
# stop processing when a rewrite has taken place
# and the target exists
RewriteCond %{ENV:STOP} =1
RewriteCond %{REQUEST_FILENAME} -f [OR]
RewriteCond %{REQUEST_FILENAME} –d
RewriteRule ^ - [L]
# replace the query string
RewriteCond %{QUERY_STRING} ^(.*)fake-wpdmdl(.*)
RewriteRule (.*) $1?%1wpdmdl%2 [L,E=STOP:1]
# direct requests containing wpdmdl will be
# caught here
RewriteCond %{QUERY_STRING} wpdmdl
RewriteRule .* wp-boobytrap.php
```

Figure 13. Rewrite and substitution rules for faked query strings.

Since this method of redirection is triggered only by requests for missing resources it obviously does nothing to prevent attacks against plug-ins that are installed and in use by the web site. Such attacks can potentially be mitigated using manual booby trapping, but as mentioned earlier this is very time consuming and can be error prone depending on where the booby trap has to be inserted.

We have explored the possibility of renaming installed plug-ins so that requests using a plug-in's standard URL in a potential exploit would end up being redirected to the booby trap. For this approach to be feasible, no manual modifications to the plug-ins or WordPress itself can be done, since that would make their respective update processes very cumbersome; the same modifications would have to be re-applied every time a plug-in is updated. Our research has shown that this task can be accomplished, at least partially, without editing any existing source code, using something we call faked redirection.

We have identified three ways in which an exploit may end up running code belonging to a WordPress plug-in:

- Requests directly to a file belonging to the plug-in
- Using hooks defined by the plug-in. The request goes to index.php and is internally routed to a function in the plug-in
- Using POST requests to index.php with execution paths similar to those of hooks.

The idea is to rewrite all URLs that can lead to plug-in code execution by non-standard names. Requests for the rewritten URLs would then be internally redirected to the original locations, while requests that have not been rewritten would be redirected to the booby trap.

This way, normal site usage is unaffected since all requests go through the modified URLs, but an attacker attempting to leverage an exploit against a plug-in would fail and end up in our booby trap.

The concept is easier to grasp using an example. Let's take the popular WordPress Download Manager plug-in as an example. Normally, the plug-in resides in wp-content/plugins/download-manager, and one of the hooks it uses is called wpmdl. We now substitute all occurences of wp-content/plugins/download-manager with wp-content/plugins/faked-download-manager and all occurences of wpdml with fake-wpdmdl.

Since we do not want to modify any files belonging to WordPress itself or one of the plug-ins, we use a combination of the mod_substitute and mod_rewrite Apache modules. mod_substitute is used to modify the URLs when the content is served to the browser, while mod_rewrite handles the task of reversing the substitution and eventually redirecting requests to the booby trap. Figure 13 illustrates how faked redirection is used to booby trap the wpdmdl hook that WordPress Download Manager uses.

IV. EXPERIMENTAL RESULTS

Experiments with booby trapped patches to vulnerabilities in WordPress modules and with redirection scripts are presented in this chapter.

1) Results with Booby Trapped WordPress Plug-ins: Intrusion attempts have produced data in log files. Some booby trapped plug-ins utilize WordPress functionality, which is reflected in the contents of log files.

The log from the booby trapped plug-in WordPress Symposium 14.11:

```
80.220.110.12 - [16/March/2015 08:17:32] /wp-
content/plugins/wp-symposium/server/php/index.php
Mozilla/5.0 (Windows NT 6.1; WOW64)
```

```
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/36.0.1985.125 Safari/537.36
```

The log shows the path to the web page, to which the intrusion attempt has tried to upload a shell code. The exploit script fakes the information about the attacker's web browser with a predefined header. Booby trapping the patch of this vulnerable plug-in is successful, since forensic information about exploitation attempts is logged but normal administrative activities are not logged.

The log from the booby trapped plug-in WordPress Shopping Cart 3.0.4 :

```
80.220.110.12 - [01/April/2015 11:22:31] /wp-
content/plugins/wp-
easycart/inc/amfphp/administration/banneruploaderscript.php
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:36.0)
Gecko/20100101 Firefox/36.0
```

The log shows the web page, which is used by the web form for file upload. The information about the intruder's web browser is public, since the intrusion attempt is made from the intruder's computer without any intermediate activity. The booby trap was not triggered by normal administrative activity.

The log from the booby trapped plug-in Fancybox for WordPress 3.0.2:

```
80.220.110.12 - [01/April/2015 10:37:07] /wp-admin/index.php
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:36.0)
Gecko/20100101 Firefox/36.0
80.220.110.12 - [01/April/2015 10:49:33] /wp-
admin/plugins.php Mozilla/5.0 (Windows NT 6.1; WOW64;
rv:36.0) Gecko/20100101 Firefox/36.0
80.220.110.12 - [01/April/2015 10:49:34] /wp-admin/admin-
ajax.php Mozilla/5.0 (Windows NT 6.1; WOW64; rv:36.0)
Gecko/20100101 Firefox/36.0
80.220.110.12 - [01/April/2015 10:50:14] /wp-admin/admin-
post.php Mozilla/5.0 (Windows NT 6.1; WOW64; rv:36.0)
Gecko/20100101 Firefox/36.0
```

Fancybox is an administrative tool. All administrative activity is logged, not only the use of adminpost.php in an exploit web form. Booby trapping the patched vulnerability in Fancybox is therefore not recommended.

Part of the log from the plug-in WordPress Download Manager 2.7.4:

```
80.220.110.12 - [31/March/2015 02:58:16] /index.php
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:36.0)
Gecko/20100101 Firefox/36.0
1.171.73.177 - [31/March/2015 03:26:37] /index.php
128.61.240.66 - [31/March/2015 03:53:49] /index.php
netscan.gtisc.gatech.edu
```

Each access to index.php has been logged, which includes all visits to the main web page of the WordPress site. Booby trapping a patch on the main page of a web site is not recommended.

2) Results with Redirection:

A honeypot WordPress installation was left running in an attempt to log potential exploit attempts. Redirection of request for missing resources was highly successful and many malicious requests were logged, including many aimed at exploiting software other than WordPress. Here's an excerpt showing attempts to detect vulnerable versions of two WordPress plug-ins, which were not installed on the server:

```
2015-04-03T11:45:10+00:00: Unhandled request for "//wp-
content/plugins/revslider/temp/update_extract/revslider/info.php
": $_GET = [ ], $_POST = [ ], $FILES = [ ]
2015-04-03T11:45:10+00:00: Unhandled request for "//wp-
content/uploads/wpallimport/uploads/d0bc023bca54df2d0c54efe
7b9e29311/info.php": $_GET = [], $_POST = [ ], $FILES = [ ]
2015-04-05T17:41:36+00:00: Unhandled request for "//wp-
content/plugins/revslider/temp/update_extract/revslider/info.php
": $_GET = [ ], $_POST = [ ], $FILES = [ ]
```

Initial testing shows that the faked redirection technique has the potential to catch malicious requests:

```
2015-05-20T08:36:10+00:00:
ExploitMocker\RequestHandler\Base\DefaultHandler -
Unhandled request for "/?wpdmdl=13": $_GET =
{"wpdmdl":"13"},
$_POST = [ ], $FILES = [ ]
2015-05-20T08:36:19+00:00:
ExploitMocker\RequestHandler\Base\DefaultHandler -
Unhandled request for "/?wpdmdl=15": $_GET =
{"wpdmdl":"15"},
$_POST = [ ], $FILES = [ ]
```

Without the faked redirection technique in place, the above requests would have succeeded since they are perfectly valid.

V. CONCLUSIONS

Booby trapping, originally proposed for active defense in network hosts against intrusion attacks based on return-oriented programming, has been shown to provide active forensics and proactive intrusion defense for attempts to exploit some patched vulnerabilities in WordPress web sites. However, for some vulnerabilities, booby trapping methodology must be further developed to distinguish between intrusion attempts and normal administrative activity.

If used as an active defense mechanism, redirecting requests for missing resources has the potential to catch attackers before they are able to attempt a successful exploit, assuming that the attackers have to try many exploits until they would find one that has the potential to work.

The technique is easy to implement on existing installations since it is not tied to any particular WordPress plug-ins that are installed on the server. Faked redirection can improve this even further since it can force attackers into the booby trap even if they attempt to leverage exploits that are currently unpatched. However, unlike when merely redirecting missing resources, this technique requires some manual configuration for each plug-in that the operator wants to protect.

Security in WordPress could be improved by enforcing and maintaining an internal registry for plug-ins storing their access methods, e.g. file system location. This would allow the web site administrator to randomly re-locate any installed plug-in, in a similar fashion to our faked re-direction method, without worrying that something may go wrong with the installation. This technique would essentially mitigate the effects of many zero-day vulnerabilities for WordPress installations utilizing third-party plug-ins, by allowing completely unique installation environments.

The plug-in renaming technique could be easier to implement, if plug-in authors would design their plug-ins with the possibility of renaming them in mind. The plug-ins we tested were quite unsuitable for this since their source code contained references to hard-coded plug-in URLs. The application to other similar CMS software

 ISBN: 978-1-61208-432-9

needs to be further investigated to draw general conclusions. Still, based on our previous experience the technique holds promise as a more general solution for implementing active intrusion management into CMS software.

REFERENCES

[1] N. Stakhanova, S. Basu, and J. Wong, "A Taxonomy of Intrusion Response Systems," Int. J. Information and Computer Security, vol. 1, no. 1/2, 2007, pp. 169-184.

[2] C. Carver, J. M. Hill, J. R. Surdu, and U. W. Pooch, "A Methodology for Using Intelligent Agents to provide Automated Intrusion Response," Proc. IEEE System, Man, and Cybernetics Information Assurance and Security Workshop, West Point, IEEE Press, 2000, pp. 110-116.

[3] S. Dai and Y. Du, "Design and Implementation of Dynamic Web Security and Defense Mechanism based on NDIS Intermediate Driver," Proc. 2009 Asia-Pacific Conference on Information Processing, IEEE Press, 2009, pp. 506-509.

[4] F. Araujo, K. Hamlen, S. Biedermann, and S. Katzenbeisser, "From Patches to Honey-Patches: Lightweight Attacker Misdirection, Deception, and Disinformation," Proc. 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14), ACM, 2014, pp. 942-953.

[5] ADHD provides tools for active defense, http://sourceforge.net/projects/adhd/ [retrived: August, 2015]

[6] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," Special Publication 800-94 Rev. 1 (Draft), NIST National Institute of Standards and Technology, U.S. Department of Commerce, 2012.

[7] M. Prandini and M. Ramilli, "Return-Oriented Programming," IEEE Security & Privacy, vol. 10, no 6, Nov.-Dec. 2012, pp. 84-87.

[8] S. Crane, P. Larsen, S. Brunthaler, and M. Franz, "Booby Trapping Software," Proc. New Security Paradigms Workshop (NSPW'13), ACM, 2013, pp. 95-106.

[9] R. Roemer, E. Buchanan, H. Shacham, and S. Savage, "Return-Oriented Programming: Systems, Languages, and Applications," ACM Trans. Information and System Security (TISSEC) – Special Issue on Computer and Communications Security, vol. 15, 2011, pp. 2-34.

[10] K. Onarlioglu, L. Bilge, A. Lanzi, D. Balzarotti, and E. Kirda, "G-Free: Defeating Return-Oriented Programming through Gadget-less Binaries," Proc. 26th Annual Computer Security Applications Conference (ACSAC '10), ACM, 2010, pp. 49-58.

[11] L. Davi, A.-R. Sadeghi, and M. Winandy, "ROPdefender: A Detection Tool to Defend Against Return-Oriented Programming Attacks," Proc. 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS '11), ACM, 2011, pp. 40-51.

[12] R. Skowyra, K. Casteel, H. Okhravi, N. Zeldovich, and W. Streilein, "Systematic Analysis of Defenses against Return-Oriented Programming," in Research in Attacks, Intrusions, and Defenses, Lecture Notes in Computer Science, vol. 8145, Springer, 2013, pp 82-102.

[13] L. Liu, J. Han, D. Gao, J. Jing, and D. Zha, "Launching Return-Oriented Programming Attacks against Randomized Relocatable Executables," Proc. 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE Press, 2011, pp. 37-44.

[14] WordPress Portal. https://wordpress.org/ [retrived: August, 2015]

[15] PHPBook. How to log ip adresses in PHP, http://www.phpbook.net/how-to-log-ip-adresses-in-php.html [retrived: August, 2015]

[16] C. Viviani, WordPress Wp Symposium 14.11 - Unauthenticated Shell Upload Exploit, http://www.exploit-db.com/exploits/35543/ [retrived: August, 2015]

[17] K. Szurek, WordPress Shopping Cart 3.0.4 - Unrestricted File Upload, http://www.exploit-db.com/exploits/35730/ [retrived: August, 2015]

[18] M. Nadeau, Security Advisory – High Severity– WordPress Download Manager, http://blog.sucuri.net/2014/12/security-advisory-high-severity-WordPress-download-manager.html [retrived: August, 2015]

[19] C. Viviani, WordPress Download Manager 2.7.4 - Remote Code Execution Vulnerability, http://www.exploit-db.com/exploits/35533/ [retrived: August, 2015]

[20] Yii framework The Fast, Secure and Professional PHP Framework. http://www.yiiframework.com/ [retrived: August, 2015]

[21] Fat-Free Framework A powerful yet easy-to-use PHP micro-framework designed to help you build dynamic and robust web applications - fast! http://fatfreeframework.com [retrived: August, 2015]

 ISBN: 978-1-61208-432-9