I. J. Computer Network and Information Security, **2015, 4, 21-28**
Published Online March 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2015.04.03

DNS Pharming through PHP Injection: Attack Scenario and Investigation

Divya Rishi Sahu, Deepak Singh Tomar
CSE Department, MANIT, Bhopal, 462003, India
Email: divyarishi.sahu@manit.ac.in, deepaktomar@manit.ac.in

Abstract—With the increase in technology, Internet has provided set of tools and technologies which has enabled web programmers to develop effective websites. PHP is most widely used server side scripting language and more than twenty million of web sites are designed through PHP. It has used as a core script in Web Content Management System (WCMS), such as Joomla, WordPress, Drupal, SilverStripe etc. PHP has also security flaws due to the certain vulnerabilities such as PHP injection, remote file inclusion and unauthorized file creation. PHP injection is a variant of code injection attacks in which PHP script may be exploited to execute remote commands. The contribution of this paper is twofold: First, it presents a unifying view of PHP injection vulnerability, which causes alteration in the 'hosts file'; Second, It introduces an investigation process against alteration in 'hosts file' through PHP injection. This attack has been introduced as a type of DNS pharming. In this investigation process a chain of evidence has been created and an algebraic signature has been developed to detect explained attack.

Index Terms—PHP Code Injection, Command Injection, DNS Pharming, Attack Scenario, Cyber Forensics, Script Kiddie.

I. Introduction

The PHP Hypertext Preprocessor (PHP) is a scripting language designed to build dynamic web sites. It is a widely accepted scripting language by the web developers and is used for writing popular web applications like Wikipedia. As per the survey of W3Techs, 82.2 % from all the web sites using server side scripting language are developed through PHP [1]. Dynamic functionality of PHP script allows programmer to execute instant remote code, to use dynamic variables, and to build new function creation on the fly. This dynamic functionality may sometime become application vulnerability due to the poorly written code and improper configuration. These vulnerabilities may be exploited by the script kiddies. Script kiddies is an unskilled person who seeks out the victim that possesses the vulnerability and injects the script (developed by others) in order to gain root access.

PHP Code injection (PHPCI) is a serious threat of web applications in which the attacker may inject malicious code through entry points of a web application. These entry points may be a query string of URL, text fields of web form or HTTP header fields. In web environment threats such as DNS Cache Poisoning [2-4], ID Spoofing [2-3] and DNS pharming [5] may be exploited by unauthorized user to tamper DNS entry. In DNS Pharming, attacker manipulates the DNS resolution process through tampering the hosts file or exploiting the vulnerabilities in DNS server [6]. Attacker may also launch PHPCI to exploit Domain Name Server.

In this paper DNS pharming and PHPCI attacks have been understood through experimental setup and developing the attack scenario of PHPCI to tamper the windows hosts file. According to practical investigation of attack, evidences have been extracted and correlated to forensically investigate the DNS pharming exploitation through PHP code injection attack. Finally an algebraic signature has been developed and results have been discussed to analyze appropriateness of the developed signature. The experimented results show that proposed approach successfully build chain of evidences subjected to DNS Pharming and provide analyzable and comprehensible information for law enforcement agencies.

II. Background

PHPCI is an application level vulnerability which occurs when user-supplied input is not properly sanitized before being passed to the PHP function. Open Web Application Security Project have rated injection vulnerability at first rank continuously from 2010 to 2013 in top 10 application security risk [7]. In the last few years researchers are trying and suggesting counter measures to handle this vulnerability. Detection and prevention of PHPCI is still a challenge for web security experts.

A. Injection Attacks

Dr. E. Benoist [8] describes the code injection attacks such as PHP injection, XML injection and Shell injection. Finally conclude that sanitization of input strings and reduces the privileges; are the two main mitigation techniques to mitigate the code injection vulnerability. Sanitization of input may be done through creating white list or black list. Arthur Gerkis [9] focus attention on prevention through white list of allowed functions, tags and whatever wishes to allow for input. However this is not also a guarantee of security.

Another researcher Avi Kak [10] identified certain issues of PHP to exploit the SQL injection attack and presents another approach to detect the attack. This is offline approach and based on the analysis of log files. Ioannis Papagiannis [11] suggests that taint tracking implementation in C may also be useful to prevent injection vulnerabilities by modifying the core of the PHP in runtime.

B. DNS Attack Investigtion

Yu Xi et al. [3] summarize the domain name resolution process and provide the concept of restoration, source port randomization and setting time-to-live (TTL) to prevent DNS poisoning. Another researcher Bruce J. Nikkel [12] presents the systematic approach for the forensic analysis of domain names and IP networks. However both works did not touch the area of hosts file which is responsible for assigning the domain to IP address in particular computer.

Lan Green [13] describes the process of DNS cache poisoning among both networks, operated through proxy or without proxy server. These techniques provide security to DNS database and records from outsiders but they have limited scope in web forensic investigation.

Research categorizes the forensic analysis into five categories in which one is time-line analysis. In 1998, Hosmer proposed time-line analysis approach [14]. This analysis is based on logs, scheduling information and memory to develop a timeline of the events that led to the incident. Ali Reza Arasteh et al. [15] proposed log analysis based on computational logic for SYN attack.

III. Attack Scenarios By Exploiting Php Vulnerability

Attack scenario is the combination of steps that attacker uses in attack [16]. It performs a significant role to acquire directive into the development of attack detection and prevention techniques. PHPCI may be launched by creating an unauthorized file on web server.

Attacker may create unauthorized file into vulnerable web server through injecting the commands. Intention of the command exploits two vulnerabilities, first that enables attacker to execute commands remotely and second enables input content to become part of operating system. PHP script to map IP address with domain name and vulnerable to command injection is as follows-

```
<?php
   ob_start();
   system('ping ' . $_GET['user']);
   $contents = ob_get_contents();
   ob_clean();
   $pos = strpos($contents, "IP");
   $ip=substr($contents,($pos+36),17);
   echo $ip;
   …….
?>
```

The functioning of this script is as follows-

Step 1. Script first concatenates the entered domain name with ping command and passes that concatenated string to system() function as a parameter.
Step 2. System() executes this string through command line interface of the OS and returns result into buffer.
Step 3. Retrieves the required IP address from buffer and reply to end user.

This vulnerable PHP script has vulnerability to execute commands from remote host. Second vulnerability due to the Command Line Interface (CLI) provides multiple executions of commands within a single line. Windows operating system uses '&' operator to execute multiple commands. Through '&' operator attacker could enter domain name with malicious command into input field to execute the command. The following vulnerable string creates the unauthorized file say 'hacker.bat' on the server-

```
http://www. domain-name.com & echo. > hacker.bat
```

A. Attack Scenario

The following attack scenario has been developed to demonstrate the DNS pharming attack through exploiting the PHP vulnerabilities.

- Name of Attack: DNS pharming through PHP code injection attack.
- Possible Attacker: having knowledge of DNS resolution, OS file system and PHP scripting.
- Possible Vulnerability: Poor privilege enforcement, improper permission in web server and able to run script on web site.
- Resources Affected: DNS cache of server machine, unauthorized redirection of end user to malicious URL, Disclosure of Operating System.

1. Attacker find the PHP vulnerability to execute commands through web server and inject following string into input field to creates new unauthorized web page say 'hacker.php'.

```
'input_value' & echo. > hacker.php
```

2. Check the presence of unauthorized web page by clicking the link-

```
www.server_domain.com/hacker.php
```

3. Attacker creates new os.php page as in previous step and writes code to obtain the running OS on server machine through injecting the following PHP script.

```
‘input_value’ & echo “<?php
 $agent = $_SERVER['HTTP_USER_AGENT'];
 if(preg_match('/Linux/',$agent))  $os = 'Linux';
   elseif(preg_match('/Mac/',$agent))  $os = 'Mac';
     elseif(preg_match('/Win/',$agent))$os = 'Windows';
 ?>” >> os.php
```

4. Inject subsequent PHP script to write into hacker.php page.

```
‘input_value’ & echo “<?php
     $handle = fopen($_SESSION['hosts-path'], "a");
       fwrite($handle, "\r\n 192.168.5.55");
       fwrite($handle, "\t www.server_domain.com");
     fclose($handle);
 ?>” >> hacker.php
```

Variable string ‘hosts-path’ represents the path of the hosts file according to the OS detected.

5. Executes the embedded PHP script through accessing the link-

```
www.server_domain.com/hacker.php
```

6. This ‘hacker.php’ executes the PHP script and concatenates the false DNS resolution entry of attackers IP address into host file.
7. Victim access web server it automatically redirected to attackers web.

To present the investigation process and severity of PHPCI, an intra-net has been established in the web security lab of MANIT, Bhopal, India. This system is installed on two local servers; one for web forensic server & another for attacker’s web; and three client machines for normal user. Web forensic server has setup with 192.168.5.13 IP and its domain name assigned with www.server domain.com. Attacker’s web is setup with 192.168.5.55 IP. Snapshot of the unauthorized hacker.php along with the altered hosts file is shown in Fig.1.

IV. Evidences For Investigation Of DNS Exploitation

Digital investigation is the process of extraction and correlation of digital evidences suitable for inclusion into criminal investigation. [17] Evidence have been extracted from the attacked system and observed that relevant evidence to investigate the developed attack scenario have exist on Windows registry, log files, file time-stamp and running processes on server.

A. Hosts file

Host file is responsible to carry out DNS resolution by binding the IP address to the domain name. Normally it is created at the time of OS installation. It may be altered by running applications in the administrative privilege mode. Web administrator may check the alteration through manual analysis. Modification time stamp of hosts file is the digital evidence to detect alteration in the hosts file.

B. Registry Entry

Registry-entry of windows operating system plays a vital role as evidence for forensic investigation. Operating system maintains the registry entry for each file modification [18]. For instance; every time that the user chooses a filename in a standard open/save dialog-box of Windows, a new registry entry is added under the following key on Windows 7 or Windows-8:

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\
CurrentVersion\Explorer\ComDlg32\OpenSavePidlMRU
```

C. File Modification Time Stamp

Modification date and time would be vital parameters to detect alteration in the host file. The following DOS command has been executed to detect the modification time of windows files and saved it into text file.

```
dir "C:\Windows\System32\drivers\etc" /T:C >>
"d:\mod_tim.txt"
```

Where /T - Controls which time field displayed or used for sorting; time field C used for Creation Time.

D. OS Installation Time Stamp

OS also maintains the installation date and time. DOS command to find the installation date and time of the operating system is as follows-

```
systeminfo | find /i “install date” >> "d:\sys_info.txt"
```

Difference in installation date/time of OS installation and modification date/time of hosts file describes that ‘hosts file’ has been modified after the installation.

E. Process Activity

Semantic correlation in process activity and its parameters provides the significant forensic evidence to detect an attack. ‘Process Monitor’ captures all the running processes of operating system and the registry entry modified through these processes. It records all the parameters related to process events. This work required seven parameters of process which are Date & Time of process execution, Name of the processes, PID, Operation Performed, Path, Result and Details.

V. Correlation Of Evidences

Further to enhance the investigation process relevant evidences have been correlated and signature has been

developed. Each type of activity has been modeled through algebraic operations and applied the gathered evidences.

A. Modeling of OS Process

Basic structure of process event is carefully chosen to convey the relevant information for investigation. It defines all variable names in a manner that is suitable to their behavior. Algebraic operation of the basic structure for the process is modeled as-

$$\textbf{Process Event: } P_{name} \times P_{ID} \times P_v \times T_{stamp} \longmapsto B_{val}$$

Where-

P_{name} = Name of application which initiate the event
P_{ID} = Identification number of the called process
P_v = Privileges of the user to conduct the event
T_{STAMP} = Server date and time when event occurred
B_{val} = $R \longmapsto \{0, 1\}$;

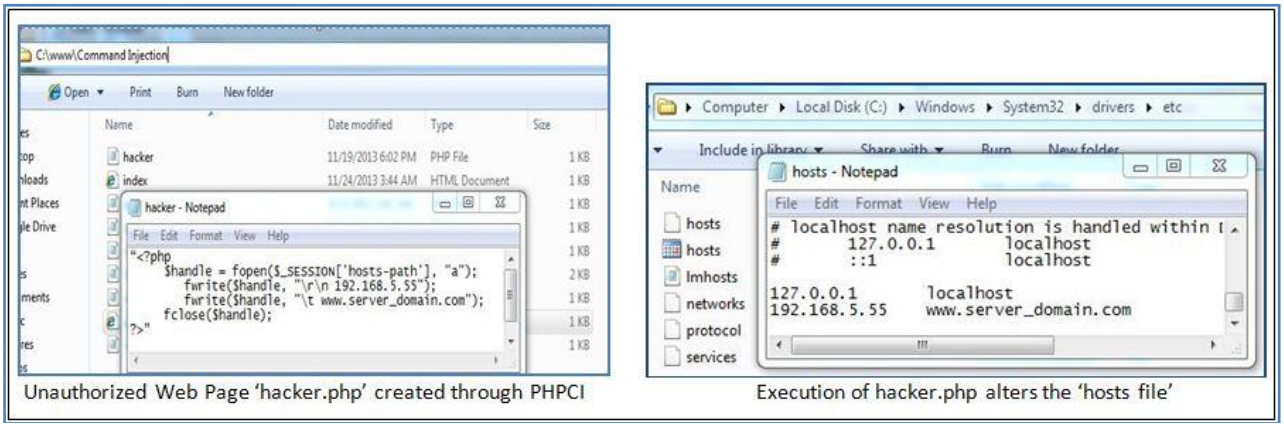

Fig. 1. Snapshots of DNS Pharming through PHPCI

$$B_{val} = \begin{cases} Event\ occured\ successfully & B_{val} = 1 \\ Event\ fails & B_{val} = 0 \end{cases}$$

Events related to reading and writing the operating system files have modeled as:

$$\textbf{Process Read: } P_{name} \times F_{path} \times P_{Nm} \times T_{stamp} \longmapsto B_{val}$$
$$\textbf{Process Write: } P_{name} \times F_{path} \times P_{Ad} \times T_{stamp} \longmapsto B_{val}$$

Where-

F_{path} = Path of the file has been written
P_{Nm} = Normal privileges (Only Read)
P_{Ad} = Administrative privileges (Read, Write)

OS records time-stamp into registry entry for each file altered through read and write events. Event to temper the registry entry has modeled as-

$$\textbf{Process RegEntry: } P_{name} \times R_{path} \times T_{stamp} \longmapsto B_{val}$$

Here- R_{path} = Path of the tempered registry file

B. Modeling of Web Server Log

This section considers the log entries that monitored by the web server. Log file at the server provide lots of information in which source IP, Time Stamp and the link entered by the end user are considered for forensic analysis. These links contained the path of file accessed by the client.

$$\textbf{GET: } IP_s \times U_{rl} \times T_{stamp} \longmapsto B_{val}$$

$$\textbf{POST: } IP_s \times U_{rl} \times T_{stamp} \longmapsto B_{val}$$

Where-

IP_s = IP address of client request the server
U_{rl} = Path of the page accessed

C. Signatures Based on Semantic Relation and Time Synchronization

Semantics is the analysis of behavior and meaning of activity. It concentrates on the relation among signifier like events, their occurrence, result of action and their denotation. Correlation into process event is the best parameter to detect the anomalous behavior in running process. Five processes have been correlated and developed a signature to detect DNS Exploitation through PHPCI attack.

Attacker first creates the unauthorized web page into Server through injecting malicious string into input field. It initiates the web server to execute CLI which is elucidate as-

$$\tilde{E_j}.create(httpd.exe, P_{IDc}, P_{ws}, d_j, t_j) \longmapsto True \qquad (1)$$

To understand the equation following definition of notations are required-

E_j = Sequence of j^{th} event
P_{IDc} = Process ID of the CLI
P_{ws} = Privileges of web server
d_j = Date of j^{th} event occurrence and
t_j = Time of j^{th} event occurrence
j = 1, 2, 3....

Here httpd.exe process is associated with apache server which called to P_{IDc} process with the privileged of server. In operating system every process event occurred with the same privileges of application which initiated that event.

CLI starts the process event to write a new unauthorized file into web server at the path of vulnerable web page which is elucidated in (2).

$$\hat{E}_j.write(cmd.exe, F_{pathw}, P_{ws}, d_j, t_j) \longmapsto True \quad (2)$$

Here P_{name} is cmd.exe defines that event is occurred by the CLI. F_{pathw} is the path of vulnerable web page. Attacker access the unauthorized web page which is recorded by the web access log and its algebraic equation is modeled as-

$$\hat{E}_j.GET(IP_{sj}, L_{up}, d_j, t_b) \longmapsto True \quad (3)$$

Unauthorized web page starts the process event to write into 'hosts file' with administrative privileges.

$$\hat{E}_j.write(httpd.exe, F_{pathh}, P_{Ad}, d_j, t_j) \longmapsto True \quad (4)$$

Here- F_{pathh} = Path of hosts file

Operating system also edits the registry value for the registry entry of 'hosts file'.

$$\hat{E}_j.RegEntry(P_{name}, R_{Ph}, d_j, t_j) \longmapsto True \quad (5)$$

Table 1. Time stamp of used machines

Sr. No.	Host Machine	Installed Operating System	OS Installation Time Stamp	'hosts file' Time Stamp
1.	DIVINE-HP	Microsoft Server-2003	27/07/2013, 17:32:02 PM	19/11/2013, 01:03:07 PM
2.	GH-PC	Microsoft Windows 7 Professional	06/04/2013, 07:06:44 PM	06/11/2009, 03:09:00 AM
3.	FAHIM	Microsoft Windows XP Professional	01/01/2002, 01:12:45 AM	04/06/2011, 03:24:00 AM
4.	PC-HP	Microsoft Windows 7 Professional	28/12/2012, 11:36:26 PM	06/02/2013, 12:18:00 PM
5.	MANISH	Microsoft Windows 8 Enterprise	11/06/2013, 03:08:17 PM	06/09/2013, 10:06:00 AM

To understand the signatures following definition of notations are required-

IP_{si} = IP Address of end user
L_{up} = Link of unauthorized web page created
P_{Ad} = Administrative privileges
R_{Ph} = Registry path for the 'hosts file'

This signature is based on the hypothesis that if web server initiates the process to temper the hosts file, it will be treated as the attack.

$$[\hat{E}_1.create(httpd.exe, PID_c, P_{ws}, d_1, t_1) \,.\, \hat{E}_2.write(cmd.exe, F_{pathw}, P_{ws}, d_1, t_2).\ \hat{E}_3.GET(IP_{s1}, L_{up}, d_i, t_b).\ \hat{E}_4.write(httpd.exe, PID_2, P_{Ad}, d_{i+1}, t_{i+1}) \,.\, \hat{E}_5.RegEntry(P_{name}, R_{Ph}, d_{i+2}, t_{i+2})] \longmapsto True \quad (6)$$

VI. Result Discussion

In this work, evidences subjected to PHPCI attack such as content of windows registry, process behavior and time stamp of files are identified and extracted successfully. Firstly, two evidences have been extracted to notify the alteration in 'hosts file'. First evidence is the registry entry for hosts file. PHPCI alters the content of windows host file. It is clearly visible from the captured evidences that before alteration registry contains only four entries and after altering the hosts file, entry has increased by one which is pointing and shown in Fig. 2 and Fig. 3 respectively.

Second evidence to notify the alteration is the difference between current time stamp of hosts file and time stamp of hosts file at the time of OS installation. The gathered temporal evidences for five different machines are represented in table 1. These two evidences represent the alteration in host file but did not clarify the intention of alteration that it had been altered malevolently or benevolently. The registry entry and file temporal evidences may not be relevant in some cases such as the host file has been modified to redirect the load on second server.

Further semantic correlation of evidences has been done to detect the intention of hosts file alteration. To make the investigation more effective, the processes running on the server during attack have been captured and their behavior has observed for suspicious activity. Going further this file has prepossessed and relevant entries such as processes associated with web server and CLI have retrieved. From the preprocessed processes, investigator yields evidences which are pointed with A, B, C & D in Fig. 4. Here, httpd.exe and cmd.exe have associated with apache http server and CLI respectively. P_{ID} is a unique process identification number assigned instantaneously to each running process.

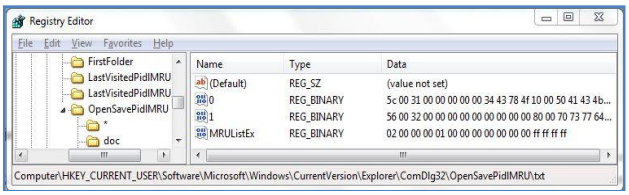

Fig. 2. Windows registry entries before editing hosts file

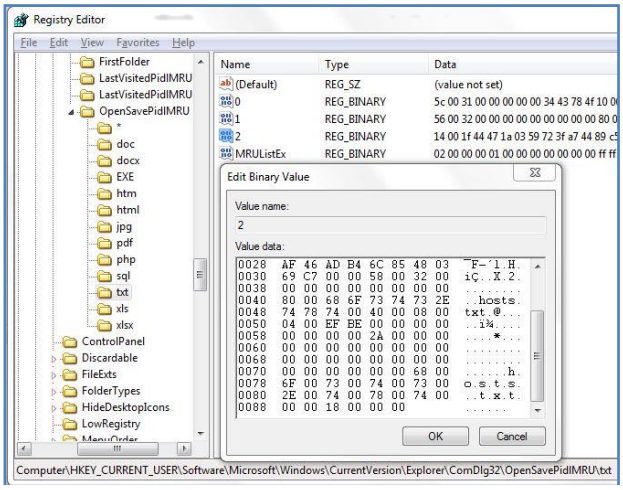

Fig. 3. Windows registry entry after editing hosts file

For this instance OS assigns 3072 PID to httpd.exe and 2460 P_{ID} to CLI as pointed with A and B respectively. Integrate these evidences in (1) returns true result.

$$\hat{E}_1.create(httpd.exe, 2460, P_{ws}, 19/11/13, 01:02:44\ PM) \longmapsto True$$

Entry pointed as B depicts that command-line writes a new unauthorized web page say 'hacker.php' at the location '\Command Injection\'. It illustrate that (2) become true.

$$\hat{E}_2.write(cmd.exe,\ C:\wamp\www\CommandInjection\hacker.php,\ P_{ws},\ 19/11/13,\ 01:02:44\ PM) \longmapsto True$$

Whenever attacker access hacker.php to tempers the hosts file, this activity has been logged into access server log file, which is depicted in Fig. 6. Put values into (3)-

$$\hat{E}_3.GET(192.168.5.55,\ /Command\ Injection/hacker.php,\ 19/11/13,\ 01:03:07\ PM) \longmapsto True$$

Process of web server writes into hosts file also be captured and pointed as C in Fig. 4. According to these values, (4) becomes true.

$$\hat{E}_4.write(httpd.exe,\ C:\Windows\System32\drivers\etc\hosts,\ P_{ws},19/11/13,\ 01:03:07\ PM) \longmapsto True$$

	A	B	C	D	E	F	G
1	Date & Time	Process Name	PID	Operation	Path	Result	Detail
2	19/11/2013 1:02:30 PM	httpd.exe	3072	CreateFile	C:\wamp\www\Command Injection\ip.php	SUCCESS	Desired Access: Generic Read, Disposition: Open, Options: Synchronous IO Non-Al
3	19/11/2013 1:02:44 PM	httpd.exe	3072	CreateFile	C:\Windows\System32\cmd.exe	SUCCESS	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point,
A 4	19/11/2013 1:02:44 PM	httpd.exe	3072	Process Create	C:\Windows\system32\cmd.exe	SUCCESS	PID: 2460, Command line: cmd.exe /c "ping google.com & echo. > hacker.php"
5	19/11/2013 1:02:44 PM	cmd.exe	2460	Thread Create		SUCCESS	Thread ID: 5852
6	19/11/2013 1:02:44 PM	cmd.exe	2460	QueryDirectory	C:\Windows\System32\ping.*	SUCCESS	Filter: ping.*, 1: PING.EXE
7	19/11/2013 1:02:44 PM	cmd.exe	2460	CreateFile	C:\Windows\System32\PING.EXE	SUCCESS	Desired Access: Read Data/List Directory, Execute/Traverse, Read Attributes, Syncl
8	19/11/2013 1:02:44 PM	cmd.exe	2460	Process Create	C:\Windows\system32\PING.EXE	SUCCESS	PID: 1088, Command line: ping google.com
9	19/11/2013 1:02:44 PM	PING.EXE	1088	Thread Create		SUCCESS	Thread ID: 1756
10	19/11/2013 1:02:44 PM	PING.EXE	1088	CreateFile	C:\Windows\System32\PING.EXE	SUCCESS	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options: Synchrc
11	19/11/2013 1:02:47 PM	PING.EXE	1088	Thread Exit		SUCCESS	Thread ID: 1756, User Time: 0.0156001, Kernel Time: 0.0312002
12	19/11/2013 1:02:47 PM	cmd.exe	2460	Thread Create		SUCCESS	Thread ID: 5856
13	19/11/2013 1:02:47 PM	cmd.exe	2460	CreateFile	C:\wamp\www\Command Injection\hacker.php	SUCCESS	Desired Access: Generic Write, Read Attributes, Disposition: Overwritelf, Options:
14	19/11/2013 1:02:47 PM	cmd.exe	2460	QueryOpen	C:\wamp\www\Command Injection\echo	FAST IO DISALLOWED	
B 15	19/11/2013 1:02:47 PM	cmd.exe	2460	WriteFile	C:\wamp\www\Command Injection\hacker.php	SUCCESS	Offset: 0, Length: 3, Priority: Normal
16	19/11/2013 1:02:47 PM	cmd.exe	2460	Thread Exit		SUCCESS	Thread ID: 5856, User Time: 0.0000000, Kernel Time: 0.0468003
17	19/11/2013 1:02:47 PM	cmd.exe	2460	Thread Exit		SUCCESS	Thread ID: 5852, User Time: 0.0000000, Kernel Time: 0.0468003
18	19/11/2013 1:02:47 PM	cmd.exe	2460	Process Exit		SUCCESS	Exit Status: 0, User Time: 0.0000000 seconds, Kernel Time: 0.0468003 seconds, Priva
19	19/11/2013 1:03:07 PM	httpd.exe	3072	QueryDirectory	C:\wamp\www\Command Injection\hacker.php	SUCCESS	Filter: hacker.php, 1: hacker.php
20	19/11/2013 1:03:07 PM	httpd.exe	3072	CreateFile	C:\Windows\System32\drivers\etc\hosts	SUCCESS	Desired Access: Generic Write, Read Attributes, Disposition: Openlf, Options: Sync
C 21	19/11/2013 1:03:07 PM	httpd.exe	3072	WriteFile	C:\Windows\System32\drivers\etc\hosts	SUCCESS	Offset: 914, Length: 16, Priority: Normal
22	19/11/2013 1:03:07 PM	httpd.exe	3072	WriteFile	C:\Windows\System32\drivers\etc\hosts	SUCCESS	Offset: 930, Length: 19
23	19/11/2013 1:03:07 PM	httpd.exe	3072	CloseFile	C:\Windows\System32\drivers\etc\hosts	SUCCESS	
24	19/11/2013 1:03:07 PM	System	4	RegOpenKey	HKU\S-1-5-18	SUCCESS	Desired Access: Maximum Allowed, Granted Access: All Access
D 25	19/11/2013 1:03:07 PM	System	4	RegQueryValue	HKCU\Software\Microsoft\Windows\CurrentVer	SUCCESS	Type: REG_BINARY, Length: 26, Data: 6D 00 73 00 70 00 61 00 69 00 6E 00 74 00 2E 00
26	19/11/2013 1:03:07 PM	System	4	RegCloseKey	HKCU\Software\Microsoft\Windows\CurrentVer	SUCCESS	

Fig. 4. Relevant Process Activities

```
127.0.0.1 - - [19/Nov/2013:12:55:13 +0530] "GET /Cmd-Inj/ HTTP/1.1" 200 169
127.0.0.1 - - [19/Nov/2013:12:55:13 +0530] "GET /Cmd-Inj/ip.php?user=google.com HTTP/1.1" 200 40
127.0.0.1 - - [19/Nov/2013:12:55:13 +0530] "GET /Cmd-Inj/ip.php?user=google.com+%26+echo.+%3E+hacker.php HTTP/1.1" 200 40
192.168.5.55 - - [19/Nov/2013:13:03:07 +0530] "GET / HTTP/1.1" 200 2572
192.168.5.55 - - [19/Nov/2013:13:03:07 +0530] "GET /Command%20Injection/ip.php?user=google.com+%26+echo.+%3E+hacker.php HTTP/1.1" 200 40
192.168.5.55 - - [19/Nov/2013:13:03:07 +0530] "GET /icons/blank.gif HTTP/1.1" 404 217
192.168.5.55 - - [19/Nov/2013:13:03:07 +0530] "GET /DNS/OS-D.php HTTP/1.1" 302 93
192.168.5.55 - - [19/Nov/2013:13:03:07 +0530] "GET /DNS/search.php HTTP/1.1" 302 53
192.168.5.55 - - [19/Nov/2013:13:03:07 +0530] "GET /DNS/write1.php HTTP/1.1" 200 102
192.168.5.55 - - [19/Nov/2013:13:03:07 +0530] "GET / HTTP/1.1" 200 2584192.168.5.55 - - [19/Nov/2013:13:03:07 +0530] "GET /DNS/ HTTP/1.1"
```

Fig. 5. Evidence into server's access log file

System generates a process to edit the registry entry value related to hosts file. This process is captured and pointed as D in Fig. 4. Developed signature detects this process through (5) and generates true result.

$$\hat{E}_5.RegEntry(system,HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\ OpenSavePidlMRU,\ P_{ws},\ 19/11/13,\ 01:03:07\ PM) \longmapsto True$$

According to these evidences result of all five equations are true which generates true result for attack signature represented into (6).

$\hat{E}_1$.create(httpd.exe, 2460, P_{ws}, 19/11/13, 01:02:44 PM) . $\hat{E}_2$.write (cmd.exe, C:\wamp\www\Command Injection \hacker.php, P_{ws}, 19/11/13, 01:02:44 PM) . $\hat{E}_3$.GET(192.168.5.55, /Command Injection/hacker.php, 19/11/13, 01:03:07 PM). $\hat{E}_4$.write (httpd.exe, C:\Windows\System32\drivers\etc\hosts, P_{ws}, 19/11/13, 01:03:07 PM). $\hat{E}_5$.RegEntry (system, HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePidlMRU, P_{ws}, 19/11/13, 01:03:07 PM) ⟼ True

Proposition P_1 be a initiation of CLI through web server, P_2 be a command execution to write a new unauthorized file into web server, P_3 be a calling of unauthorized web page, P_4 be a process to write into host file which is initiated by web server with administrative privileges and P_5 be change in registry entry of hosts file; as discussed in (1), (2), (3), (4) and (5). These events occurred sequentially. Than the attack would be detected by the validity of argument-

$$(P_1 \wedge P_2 \wedge P_3 \wedge P_4 \wedge P_5) \leftrightarrow Q, q \vdash q$$

P_1; P_2; P_3; P_4 and P_5 produce the total 64 combinations for q is false and true. In which (P_1^P_2^P_3^P_4^P_5) ⟷ q is true in 32 conditions. Therefore both (P_1 ^ P_2 ^ P_3 ^ P_4 ^ P_5) ⟷ q and q are true in only one condition when P_1; P_2; P_3; P_4 and P_5 all are also true. This shows that the given argument is valid and attack is detected. The pseudo code of developed signature to detect DNS pharming is-

```
Start
 P1.   GET: httpd.exe
       Ifexist(hacker.php)
          Goto P4
 P2.   System(cmd.exe)
 P2.1  If('string1')
          System(ping.exe)
          Execute 'string 1'
          Display
 P3.   if('string2')
          Execute 'string 2'
          write(cmd.exe, hacker.php, Pws, dj, tj);
          Goto P1
       Else END
 P4.   Write (cmd.exe, hosts, Pws, dj, tj);
 P5.   RegEntry(Pname, RPh, dj, tj);
Finish
```

Figure 6 depicts the Control Flow Graph (CFG) corresponding to the flow control of detection process.

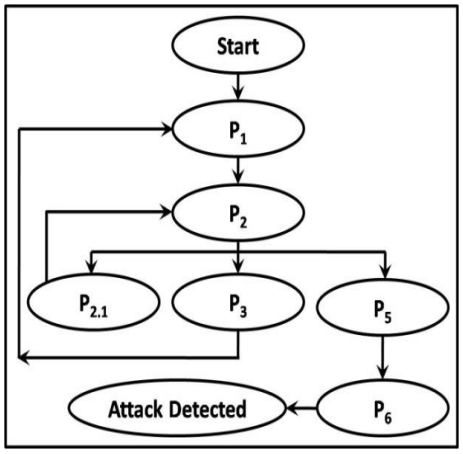

Fig. 6. CFG for developed signature

According to captured evidences in this work all values of preposition P1; P2; P3; P4 and P5 are true and attack is detected successfully. These signatures also may be effective in honey pot, cyber forensic tools, IDS systems, antivirus program and firewall to detect this attack.

VII. Conclusion And Future Scope

The field of cyber forensics has developed (and still in development) as one of the most dynamic and powerful investigative techniques in use on the cutting edge of research. Script kiddies and investigators have been running an endless battle.

In this work, an investigation has been carried out to detect DNS pharming attack through 'hosts file' securitization. Firstly an attack scenario of DNS Pharming through PHP code injection attack has been developed. Secondly, common source of evidences subjected to developed attack scenario have been identified successfully. These evidences have been correlated to build the chain of evidences which provide ease of tracking down cause of incident and depict a complete scene. Moreover, for fast incidence matching, an algebraic signature has been developed from identifying chain of evidences.

Finally developed algebraic signature has been verified through successfully applying the gathered evidences. Deduced that developed algebraic signature helps to improve attack investigation process of Intrusion Detection System, Honey Pots or Anti-virus Programs. One of the benefits of developed algebraic signature is that it provides smooth evidence tracking from chain of evidence and also pointed out cause of vulnerability

One of the limitations of signature based detection is that it cannot detect unknown vulnerabilities. As code injection vulnerability is growing exponentially, the number of signatures for code injection also increases alarmingly. To handle this problem in future, the focus will be to develop the detection strategy for more signatures.

REFERENCES

[1] W3Techs, "Usage of server-side programming languages for websites", [Online]. Available: http://w3techs.com/technologies/overview/programming language/all [Accessed: 10 August 2014].

[2] R. Bassil, R. Hobeica, W. Itani, C. Ghali, A. Kayssi, A. Chehab, "Security analysis and solution for thwarting cache poisoning attack in the domain name system", in 19th Int. Conf. on Telicommunication (ICT), pp. 1-6, IEEE, Jounieh (2012), DOI:10.1109/ICTEL.2012.6221233

[3] Yu Xi, Chen Xiaochen, Xu Fangqin, "Recovering and protecting against DNS cache poisoning attacks", in Intl. Conf. of Information Technology (ICIT), Computer Engineering and Management Sciences, IEEE Computer Society, pp. 120-123. Nanjing, Jiangsu(Sept., 2011), DOI:10.1109/ICM.2011.266.

[4] T. Mantoro, S. A. Norhanipah, A. F. Bidin, "An Implementation on Domain Name System Security Extensions Framework for the Support of IPv6 Environment", in Int. Conf. on Multimedia Computing and Systems (ICMCS), pp. 1-6. IEEE, Ouarzazate (April, 2011) DOI: 10.1109/ICMCS.2011.5945627.

[5] M. Janbeglou, M. Zamani and S. Ibrahim, "Redirecting outgoing requests toward a fake DNS server in a LAN", in IEEE Int. Conf. on Software Engineering and Service Sciences (ICSESS), pp. 29-32. Beijing (July-2010) DOI: 10.1109/ICSESS.2010.5552339.

[6] Dr. Wenliang, "DNS Pharming Attack Lab", [Online]. Available: http://www.cis.syr.edu/ wedu/seed/Labs/Attacks DNS/DNS.pdf [Accessed: 19 September 2013].

[7] OWASP: "OWASP top 10-2013 the ten most critical web application security risks June 2013" [Online]. Available: https://www.owasp.org/index.php/Top 10 2013-Top 10 [Accessed: 1 July 2013].

[8] Dr. E. Benoist, "Injection Flows (part 2) Shell Injection, PHP Injection, XML Injection, 2013 [Online] Available: www.benoist.ch/SoftSec/slides/injectionFlows/slidesInjectionFlows2.pdf [Accessed: 17 August 2013].

[9] Arthur Gerkis, "Obvious and not so obvious PHP code injection and evaluation" 20th May-2010, [Online]. Available: http://phpsecurity.org/2010/05/20/mops-submission-07-our-dynamic-php/ [Accessed: 11th August 2013].

[10] Avi Kak, "Lecture 27: Web Security: PHP Exploits and the SQL Injection Attack", April-2013, [Online]. Available:https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture27.pdf [Accessed: 17 August 2013].

[11] Ioannis Papagiannis, Matteo Migliavacca, Peter Pietzuch, "PHP Aspis: Using partial taint tracking to protect against injection attacks" in Proc. of 2nd USENIX Conf. on web application development, pp. 13-24. USA (June, 2011).

[12] Bruce J. Nikkel, "Domain name forensics: a systematic approach to investigating an internet presence"' in Int. J. of Digital Forensics & Incident Response-Elsevier, vol: 1, pp. 247-255 (2004).

[13] Lan Green, "DNS spoofing by the man in the middle SANS Institute", January-2005 [Online]. Available: http://www.sans.org/readingroom/whitepapers/dns/dns-spoofing-man-middle-1567 [Accessed: 28 July 2013].

[14] Chet Hosmer, "Time lining computer evidence", [Online]. Available: http://www.wetstonetech.com/f/timelining.pdf [Accessed: May 25, 2013].

[15] Ali Reza Arasteh, Mourad Debbabi, Assaad Sakha, Mohamed Saleh, "Analyzing multiple logs for forensic evidence" in Int. J. of Digital Forensics & Incident Response- Elsevier, Vol: 4, pp. 82-91, (September-2007) DOI:10.1016/j.diin.2007.06.013.

[16] Dr. Eric Cole, "Constructing Attack Scenarios for Attacker Profiling and Identification", [Online]. Available:http://www.securityhaven.com/docs/ConstructingAttackScenariosforAttackerProfilingandIdentificationv6.pdf [Accessed: 30 June 2013].

[17] K. K. Sindhu and Dr. B. B. Meshram, "Digital Forensic Investigation Tools and Procedures", in inter. J. Computer Network and Information Security, Vol: 4, Issue: 4, pp. 39-48 (2012).

[18] Brendan Dolan-Gavitt, "Forensic analysis of the Windows registry in memory", in Int. J. of Digital Forensics & Incident Response- Elsevier, vol: 5, pp. S26-S32 (2008).

Authors' Profiles

Divya Rishi Sahu is currently pursuing his Ph. D. in CSE department from Maulana Azad National Institute of Technology (MANIT), Bhopal, India. He obtained B. E. (Information Technology) from IGEC, Sagar and M Tech (Information Security) from MANIT, Bhopal. He has published more than 10 research papers.

Dr. Deepak Singh Tomar obtained his B. E., M. Tech. and Ph. D. degrees in CSE department. He is currently Assistant Professor of CSE department at NIT-Bhopal, India. He is co-investigator of Information Security Education Awareness (ISEA) project under Govt. of India. He has more than 19 years of teaching experience. He has guided 24 M Tech and 2 PhD Thesis. Besides this he guided 70 B Tech and 15 MCA projects. He has published more than 44 papers in national & international journals and conferences. He is holding positions in many world renowned professional bodies. His present research interests include web mining and cyber security.

How to cite this paper: Divya Rishi Sahu, Deepak Singh Tomar,"DNS Pharming through PHP Injection: Attack Scenario and Investigation", IJCNIS, vol.7, no.4, pp.21-28, 2015.DOI: 10.5815/ijcnis.2015.04.03

