Remote Execute 공격 탐지를 위한 /proc/kcore에 대한 연구

박재홍° ,황성철, 강흥식
인제대학교 컴퓨터공학부
mr8kor@kornet.net°, mslove1@nate.com, hskang@cs.inje.ac.kr

A Study Of /proc/kcore For Detection Against Remote Execute Attack

JaeHong Park° ,SungChul Hwang, HeungSeek Kang
Dept. of Computer Engineering, Inje University

요 약

오늘날 컴퓨터 보안 분야에서는 이론적인 시스템 보호 방법뿐만 아니라 이미 침입을 당한 시스템으로부터 침입 과정을 분석하고 이를 바탕으로 문제점을 보완하고 새로운 보호 방법을 찾기도 한다. 이러한 과정은 침입 시스템에 대한 컴퓨터 포렌식이라는 과정을 거쳐 수행하게 된다. 컴퓨터 포렌식은 로그 분석부터 패킷 분석에 이르기까지 다양한 방법을 이용한다. 최근 들어 컴퓨터 포렌식을 우회하는 Remote Execute 공격방법이 발견되었는데 이 공격은 기존의 많은 포렌식 절차를 무력화시킨다는 위험성을 가진다. 본 논문에서는 Remote Execute 공격 실험을 통하여 그 위험성을 알리고 대처방안을 제안한다. 본 논문에서 제안하는 /proc/kcore 분석 및 백업 메커니즘은 Remote Execute 공격에 대한 컴퓨터 포렌식을 가능하게 한다.

1. 서 론

시스템에 불법적인 침입징후가 발견되거나 포착되면 관리자는 컴퓨터 포렌식을 수행한다. 컴퓨터 포렌식은 여러 가지 절차에 따라 진행되는데 일반적으로 가장 먼저 수행하는 작업으로는 침입당한 시스템에 남겨진 로그의 분석이다. 로그 분석 작업은 syslog 데몬에 의한 파일 기반의 로그 및 tcpdump와 유사한 패킷 캡쳐 도구를 통한 패킷 로그 분석이 이루어지는데 신뢰성 있는 침입 증거 및 악의적인 작업내용이 발견되지 않는다면 해당 시스템의 파일 시스템과 실제 메모리의 이미지가 기록되는 /proc/kcore를 분석하는 보다 심층적인 단계로 접어들게 된다. 하지만 Remote Execute 공격은 공격이 성공했을 경우에 컴퓨터 포렌식의 핵심 요건이 되는 침입 관련 정보가 시스템에 전혀 기록되지 않기 때문에 컴퓨터 포렌식이 무의미 해진다는 것이다.

본 논문에서는 Remote Execute 공격의 실험을 통해 컴퓨터 포렌식의 우회가 가능함을 확인하고 /proc/kcore을 이용하는 대처방안에 대해서 알아본다. 본 논문의 구성으로는 2절에서 Remote Execute 공격에 대한 실제 예를 들어 알아보고 3절에서는, 이를 막기 위한 /proc/kcore 분석 및 백업을 위한 전송 메커니즘을 제안한다. 그리고 4절에서 결론을 맺고 향후 연구 방향에 대해 설명한다.

2. 관련연구

2.1 Remote Execute 공격

Remote Execute 공격은 악성 코드를 물리적인 디스크에 접근하기 전에 메모리에서 처리하는 방법과 스크립트 해석기에 적재 시키는 방법을 사용한다. 후자의 경우는 특별한 도구를 사용하지 않고 리눅스 시스템에 기본적으로 운용중인 도구를 이용하므로 잠재적인 위험성은 매우 높다고 할 수 있다. Remote Execute는 서버/클라이언트 형태로 공격이 이루어지는데 [그림 1]은 Remote Execute Server의 동작 순서이다.

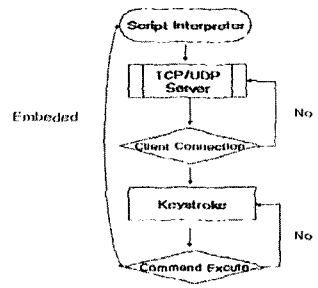

그림 1 Remote Execute Server Flowchart

[그림 2]는 Remote Execute Server를 구현한 스크립트이다.

```
#!/bin/gawk -f
function Remote_Execute_Server_Prototype(Server,Prompt) {
        while(1) {
                do {
                print Prompt |& Server;
                Server |& getline Execute;
                if(Execute) {
                        while((Execute |& getline)>0) print |& Server
                        close(Execute)
                }
                } while(Execute!="exit")
                close(Server)
        }
}
BEGIN {
        Port=ARGV[1]
        if(Port=="NULL") {
        printf "./Remote_Execute_Server.awk [Port]\n"
        exit
        }
        Server="/inet/tcp/" Port "/0/0"
        Prompt="Remote_Execute_Server> "
        Remote_Execute_Server_Prototype(Server,Prompt)
}
```

그림 2 Remote Execute Server Script

이 프로그램은 awk 스크립트 해석기를 실행하면서 표준 입출력을 스크립트 해석기로 복사해 터미널이나 디스크에 로그를 남기지 않는다. 스크립트의 실험은 리눅스에서 이루어졌고 [그림 3]은 Remote Execute 공격을 수행하는 과정을 보여주고 있다. 스크립트 해석을 스크립트 해석기에 임베디드 시킨 후 telnet을 통해 접속하였고 임의의 명령을 수행하고 파일을 정상적으로 생성하였다.

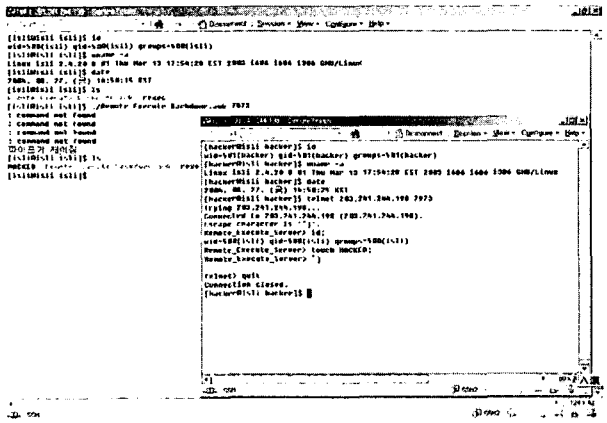

그림 3 Remote Execute Attack

이것이 컴퓨터 포렌식를 어떻게 우회하는지는 [그림 4]를 참조한다. Remote Execute 공격의 결과가 터미널이나 디스크에 로그를 남기지 않았음을 보여주고 있다.

```
[isli@isli isli]$ tail -n 6 .bash_history
id
uname -a
date
ls
./Remote_Execute_Backdoor.awk 7979
tail .bash_history
[isli@isli isli]$ su - root
Password:
[root@isli root]# tail /var/log/messages
Aug 27 12:07:03 isli sshd(pam_unix)[6418]: session closed for user mr8
Aug 27 14:25:07 isli  8월 27 14:25:07 su(pam_unix)[9911]: session opened for user hacker by mr8(uid=0)
Aug 27 14:53:29 isli  8월 27 14:53:29 su(pam_unix)[9263]: session closed for user isli
Aug 27 14:53:41 isli sshd(pam_unix)[9215]: session closed for user mr8
Aug 27 14:53:44 isli sshd(pam_unix)[10215]: session opened for user mr8 by (uid=0)
Aug 27 14:54:28 isli  8월 27 14:54:28 su(pam_unix)[10243]: session opened for user isli by mr8(uid=0)
Aug 27 15:18:37 isli  8월 27 15:18:37 su(pam_unix)[10370]: session opened for user root by mr8(uid=500)
Aug 27 15:19:25 isli  8월 27 15:19:25 su(pam_unix)[10378]: session closed for user root
Aug 27 15:19:46 isli  8월 27 15:19:46 su(pam_unix)[10428]: session opened for user root by mr8(uid=500)
Aug 27 15:19:54 isli  8월 27 15:19:54 su(pam_unix)[10464]: session opened for user root by mr8(uid=501)
You have new mail in /var/spool/mail/root
[root@isli root]# tail /var/log/secure
Aug 25 17:24:49 isli xinetd[2770]: START: sgi_fam pid=3188 from=(no address)
Aug 25 17:37:37 isli sshd[3394]: Accepted password for mr8 from 203.241.244.196 port 3186 ssh2
Aug 25 22:44:26 isli sshd[3808]: Accepted password for mr8 from 203.241.244.196 port 1783 ssh2
Aug 26 21:18:30 isli sshd[6413]: Accepted password for mr8 from 203.241.244.196 port 4592 ssh2
Aug 27 00:43:08 isli sshd[6951]: Accepted password for mr8 from 203.241.244.196 port 1822 ssh2
Aug 27 12:03:01 isli sshd[9168]: Accepted password for mr8 from 203.241.244.196 port 2891 ssh2
Aug 27 12:03:13 isli sshd[9215]: Accepted password for mr8 from 203.241.244.196 port 2892 ssh2
Aug 27 14:39:58 isli adduser[9899]: new group: name=hacker, gid=501
Aug 27 14:39:58 isli adduser[9899]: new user: name=hacker, uid=501, gid=501, home=/home/hacker, shell=/bin/bash
Aug 27 14:53:44 isli sshd[10215]: Accepted password for mr8 from 203.241.244.196 port 1508 ssh2
```

그림 4 .bash_history, syslogd log

실험결과, 쉘에서 일어나는 키스트로크를 기록하는 파일인 .bash_history에 흔적이 남지 않았으며 telnet 접속과 임의의 명령을 수행하였으나 syslog 데몬에 의해 기록되는 로그도 없음을 확인할 수 있다. 간단한 예이지만, Remote Execute 공격이 성공적으로 수행될 경우 기본적인 리눅스의 로그정책이 쉽게 무력화되어 컴퓨터 포렌식의 우회가 가능함을 알 수 있다. 이후, 파일 시스템 덤프와 /proc/kcore의 분석을 시도하게 되는데 파일 시스템 덤프는 침입 당한 시점의 파일 시스템 자체를 이미지화 시켜서 백업 파일 시스템을 생성하는 것으로 이것은 분석을 위한 상황 구축을 목적으로 진행된다.

/proc/kcore는 시스템의 물리적 메모리에 대한 가상 이미지로 이것의 크기는 물리적 메모리의 크기와 정확히 일치하지만 메모리에 접근 할 때에만 생성되므로 실제 디스크 용량을 차지하지는 않는다. /proc/kcore는 메모리를 이용하는 모든 작업들이 기록되는데 이러한 특성 때문에 컴퓨터 포렌식의 핵심적인 분석 자료로 사용된다. /proc/kcore는 컴퓨터의 전원이 꺼짐과 동시에 초기화가 되기 때문에 앞선 파일 시스템 덤프에서 침입이 이루어진 시점을 파악하여 /proc/kcore를 보관하게 된다. /proc/kcore는 메모리를 이용하는 바이너리 및 문자 데이터로 이루어져 있으며 바이너리의 경우 다양한 패턴으로 인해 분석에 문제가 있다. [그림 5]는 /proc/kcore의 문자 데이터를 추출하여 Remote Execute 공격의 증거를 획득한 화면이다.

```
[root@isli root]# strings -t d /proc/kcore > /root/kcore_strings.dump &
[1] 11798
[root@isli root]# cat kcore_strings.dump | grep "Remote_Execute" | tail
267452488 ./Remote_Execute_Backdoor.awk 3133
267455496 vi Remote_Execute_Backdoor.awk
267455568 ./Remote_Execute_Backdoor.awk
267599526 499638A3 '0  28  0  ~/Remote_Execute_Backdoor.awk
267599576 49963884 '1  19  4  ~/Remote_Execute_Backdoor.awk
267599626 49963925 '2  26  0  ~/Remote_Execute_Backdoor.awk
267599676 49963966 '3  3  6  ~/Remote_Execute_Backdoor.awk
267599725 499640A6 '4  21  0  ~/Remote_Execute_Backdoor.awk
267599775 499640A7 '5  22  29  ~/Remote_Execute_Backdoor.awk
267599826 499640B9 '6  23  5  ~/Remote_Execute_Backdoor.
[1]+  Done                    strings -t d /proc/kcore >/root/kcore_strings.dump
You have new mail in /var/spool/mail/root
[root@isli root]#
```

그림 5 /proc/kcore

3. /proc/kcore 로깅 메커니즘

Remote Execute 공격의 실험으로 위험성과 그 방어책으로 /proc/kcore의 분석이 있음을 알 수 있다. 그러나, /proc/kcore는 전원이 소거되면 내부의 내용이 모두 사라지기 때문에 불법적인 침입자가 reboot 또는 halt 명령을 내렸을 경우, /proc/kcore를 통한 컴퓨터 포렌식은 불가능해 진다. 이 문제점을 해결하기 위한 최선의 방안은, /proc/kcore의 실시간 전송을 통한 백업이라고 할 수 있다. /proc/kcore 파일의 크기가 물리적 메모리 용량과 일치하기 때문에 수정사항이 생길 때 마다 전체 파일을 송신하는 것은 시간적인 딜레이는 물론 용량상의 문제가 발생하고 컴퓨터 포렌식을 수행하는 관리자가 필요한 것은 바이너리가 아니라 해석이 가능한 문자 데이터이기 때문에 /proc/kcore의 전송 메커니즘은 [그림 6]과 같이 구성된다.

시스템이 부팅되고 리눅스가 운용환경으로 정상적으로 진입한 직후부터 수행되어야 하므로 /etc/init.d/pkcore 데몬 스크립트가 제어를 담당하고 /proc/kcore의 바이너

리 데이터를 제거하기 위한 문자 데이터 매칭을 실시한다. 문자 데이터 매칭은 지정한 값에 대한 단위 추출 및 전체 문자 데이터의 보관이 이루어진다. 추출된 문자 데이터는 파일로 기록하는 호스트 기반과 메일이나 원격 로그서버에 저장하는 네트웍 기반을 함께 사용한다. 파일 로그 및 원격 로그서버에 저장하는 것은 syslog 데몬에 위험하며 일정한 크기가 되면 관리자의 외부 메일로 전송하는 방안을 택한다.

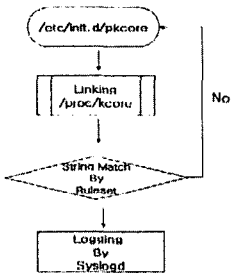

그림 6 /proc/kcore Logging Mechanism

중요한 부분은, 문자 데이터의 추출인데 크게 세 가지 방법을 적용할 수 있다. 첫째는, 정규표현식을 이용해 문자열로 판단된 라인 단위로 전송한다, 두 번째는, 관리자가 지정한 문자열에 일치하는 라인만 전송한다. 마지막으로, 기존의 침입탐지 시스템의 룰 셋을 적용하는 것이다. 앞으로, Remote Execute를 통한 침입이 활성화 될 것으로 예상되므로 기존의 침입탐지 시스템의 룰셋과 관리자 지정 패턴을 조합하여 사용해야 한다.

본 논문은, /proc/kcore를 통한 Remote Execute 공격의 탐지를 다루었지만, 정상적으로 탐지된 /proc/kcore의 패턴은 검증을 거쳐 방화벽 및 침입탐지 시스템과 연동하여 Remote Execute 공격의 차단[그림 7]에도 이용할 수 있다

.

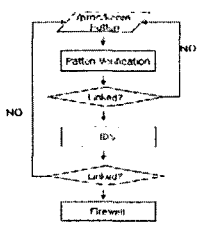

그림 7 /proc/kocre Link Flowchar (IDS,Firewall)

IDS의 룰셋은 일반적으로 문자 기반으로 작성되기 때문에 /prco/kcore의 로그 자료를 적용하기 위한 별도의 변환 작업이 필요하지 않고 룰셋을 적용시킬 IDS의 룰셋 규칙만 지켜주면 된다. 룰셋에 적용된 Remote Execute의 패턴은 패턴이 적용된 IDS의 방화벽 연동기능을 통하여 침입/탐지/차단이 가능하다. 일반적인 IDS와 방화벽 연동으로 Remote Execute 공격을 막는 방안도 생각할 수 있으나 시스템에 합법적인 계정을 가진 내부 침입자가 Remote Execute 공격을 가할 경우를 감안한다면, /proc/kcore의 분석를 통한 문자 추출 및 연동은 의미가 있다.

4. 결론 및 향후 연구과제

본 논문에서는, Remote Execute 공격의 실험을 통하여 위험성을 확인했으며 대응방안으로 /proc/kcore의 분석을 통한 전송 메커니즘에 대해서 제안했다. 그리고, /proc/kcore의 로그 자료를 IDS의 룰셋으로 등록시키고 방화벽과 연동시키므로 Remote Execute 공격의 탐지 및 차단도 가능함도 알 수 있다.

본 논문에서 제시한 /proc/kcore를 이용한 탐지 및 로그 전송 메커니즘은 Remote Execute 공격의 탐지 및 차단은 물론 침입 이후에 수행하는 컴퓨터 포렌식을 위한 증거자료의 보존에도 효과를 기대할 수 있다

본 논문의 Remote Excute 공격의 탐지 및 차단을 위한 메커니즘은 문자 데이터의 추출 및 가공을 거친 로깅과 로깅 정보의 응용에 중점을 두고 있는데 이것은, 아직까지 Remoe Excute 공격이 평문(plain text) 형태로 사용되기 때문이다. 만약, 침입을 시도하는 공격자가 Remote Execute 공격에 사용되는 모든 데이터를 암호화하여 진행한다면 탐지 및 차단이 어려워지며 무엇보다, /proc/kcore를 통한 컴퓨터 포렌식 작업시 암호화된 자료를 해독해야 하는 문제에까지 봉착하게 된다.

앞으로 이 부분에 대한 다양한 연구가 이루어야 할 것으로 생각한다.

5.참고문헌

[1] 심현철,강용혁,엄영익,"임베디드 Linux 시스템 기반 프로세스 동시 디버깅을 지원하는 원격 디버거 설계 및 구현",한국정보처리학회 논문지 A, 2003.10

[2] 심현철,강용혁,엄영익,"리눅스 환경에서의 다중 프로세스 응용에 대한 원격 디버깅 기법",한국정보과학회 논문지 C, 2002.12

[3] 나형준,김문기,이병호,"시스템 콜 인터셉트와 로깅 시스템을 이용한 리눅스 기반 자원 접근제어 모듈(LPM) 설계",2003.04

[4] 이재국,김현식,"리눅스 파일시스템에서의 로그 기반 침입 복구 기법",정보과학회 2003년 춘계학술대회,2003.04

[5] 윤인숙,장범환,정태명,"로그 분석을 통한 서비스 관리 시스템 설계 및 구현",2002.04

[6] vangelis,http://www.wowhacker.org,2004.07