Exploits remots i locals

José María Alonso Cebrián
Jordi Gay Sensat
Antonio Guzmán Sacristán
Pedro Laguna Durán
Alejandro Martín Bailón
Jordi Serra Ruiz

PID_00208379

UOC
Universitat Oberta de Catalunya

Índex

Introducció..... 5

1. *Exploits* remots..... 7
1.1. Entorns de treball per a creació d'*exploits*..... 9
1.2. Metasploit 9
1.3. Un exemple amb el Metasploit 10
1.4. Definició de l'*exploit*..... 14

2. *Exploits* locals..... 17
2.1. *Fuzzing*..... 18

Introducció

Hem pogut veure com un programa funcionalment correcte podria ser insegur a causa d'un error. Aquests errors poden ser aprofitats per programes per a treure'n algun partit que incideixi en la seguretat del sistema. Aquests programes que treuen partit dels errors es coneixen com a *exploits*. En aquest capítol ens familiaritzarem amb els *exploits* abans de començar a veure com poden ser creats a partir del capítol següent.

Classificar tots els tipus d'*exploits* que hi ha és una tasca àrdua, sobretot tenint en compte la quantitat de característiques que es poden utilitzar per a establir les classes. Es podria escollir la plataforma maquinari que afecten o el sistema operatiu per al qual estan dissenyats. És innegable que quan s'analitza la seguretat d'un sistema aquests dos ordres de classificació són molt importants. Des del punt de vista de la gestió de la seguretat potser l'impacte i la perillositat són classificacions per tenir molt en compte, ja que els plans de reparació han d'estar guiats per un ordre clar de criticitat.

Hi ha una pàgina web, http://www.milw0rm.com/, en què la categorització es fa depenent de si l'*exploit* afecta una vulnerabilitat remota o local, encara que a més es fan dues distincions més: *exploits* destacats i *exploits* per a aplicacions web.

Captura del lloc milw0rm.com

En aquesta pàgina es publiquen *exploits* de tot tipus per a gairebé qualsevol error que es trobi. La publicació d'aquests *exploits* és una mica caòtica, atès que els administradors publiquen el text de l'*exploit* tal com és reportat al web. Per tant, encara que tots els *exploits* solen conservar una estructura similar, no tots segueixen un mateix patró o estructura, com es fa en altres pàgines com Secunia o Security Focus.

Volem recomanar milworm com a font de coneixement constructiu i mai destructiu per a comprendre millor el funcionament dels *exploits* i els trucs usats normalment en l'explotació dels errors.

Des del punt de vista acadèmic, el més important per a nosaltres seria establir un ordre basat en l'arquitectura de l'*exploit*, és a dir, entendre com està creat, quin és el patró de creació de l'*exploit* i quina l'arquitectura interna.

En els apartats propers estudiarem les característiques d'alguns *exploits* locals i remots i un entorn de treball creat per a la creació i ús d'*exploits*, Metasploit. Les arquitectures internes dels *exploits* les anirem veient en els capítols propers.

1. *Exploits* remots

Aquests *exploits*, capaços de treure partit d'un error, són sens dubte els més populars, perillosos, efectius i desitjats pels creadors d'*exploits*. A més, són els més cotitzats.

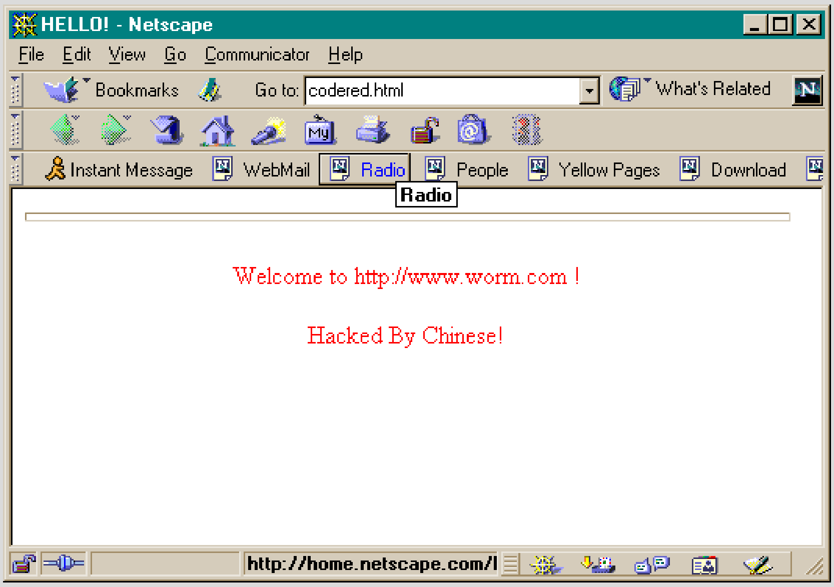
Un *exploit* que permeti aconseguir privilegis d'administrador en una màquina remota de la qual només coneixem el nom o la IP i un port amb un servei vulnerable a l'escolta és el mateix que la possibilitat d'utilitzar una o mil màquines com si fossin de propietat.

Això fa que les companyies de desenvolupament de programari hagin d'estar especialment preocupades en les mesures de protecció dels serveis que s'exposen a la xarxa en un servidor.

Un clar exemple d'aquests *exploits* remots són els cucs que hem comentat en el primer capítol. Un cuc que es difon per una xarxa com *Sasser*, *Blaster* o *Code Red* ha de tenir una manera de copiar-se d'una màquina a una altra. Aquests cucs, per exemple, tenien en comú que aprofitaven serveis de xarxa vulnerables explotables en remot.

Code Red

Code Red es va posar a Internet el dia 13 de juliol de l'any 2001. Utilitzava com a patró de replicació servidors Microsoft Windows NT 4.0 o 2000 que executaven el programari de servidor d'Internet Information Server 4.0 amb el servei vulnerable Indexing Service 2.0. La vulnerabilitat era un desbordament de memòria intermèdia que permetia executar codi injectat en memòria per a fer un *defacement*, és a dir, una suplantació de la pàgina principal del servidor web, per una que deia "hacked by Chinese".



Servidor infectat per *Code Red*

Per a infectar una màquina des d'una altra, *Code Red* llançava un *exploit* contra el programa del servei d'indexació que s'executava en el servidor web. El servei utilitzava el programa `default.ida` que tenia un desbordament de memòria intermèdia en el paràmetre d'entrada i amb una petició com la següent s'aconseguia infectar d'un servidor a un altre:

```
GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%
u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u909
0%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a HTTP/1.0
```

Com es pot observar, la capçalera contenia gran nombre de caràcters `N`, usats per a desbordar la memòria intermèdia i col·locar el codi desitjat en memòria. Només es necessitava localitzar un servidor vulnerable i llançar una petició correctament malformada per a aconseguir infectar el sistema.

En aquest cas, l'*exploit* feia ús del protocol HTTP, però es podria aplicar a qualsevol servei vulnerable que rebi dades des de la xarxa. Un servidor FTP que no controli la mida del camp del nom d'usuari o un servidor DNS que accepti IP d'un rang no vàlid podrien ser exemples de serveis vulnerables utilitzables per a explotar un servidor en remot.

MSN Messenger 7.X (MS07-054)

Un altre exemple d'*exploits* remots és el que es va publicar l'11 de setembre del 2007 per al client de missatgeria instantània MSN Messenger 7.X (MS07-054) que permetia l'execució remota de codi per mitjà del protocol de vídeo de la càmera web.

L'error es produïa en enviar un paquet de càmera web malformat. Es va descobrir que en el protocol de comunicació hi havia una falta de validació pel que feia a la mida assignable a les variables, i donava lloc a un error de desbordament de *heap*. Els errors de desbordament de *heap* els veurem durant aquest curs, però la idea consisteix a desbordar les variables situades en la part del *heap*, destinada a valors dinàmics. Aquest error va permetre crear un *exploit* que generava paquets de vídeo malformats i que aconseguia, en els entorns vulnerables, una consola remota en l'equip atacat.

Alguns *exploits* remots requereixen la interacció amb la víctima o un entorn autenticat perquè puguin ser explotats. El gran risc és l'aparició d'*exploits* remots que permetin executar codi arbitrari en la màquina atacada en serveis àmpliament usats, com ja va passar en el passat amb els cucs esmentats.

1.1. Entorns de treball per a creació d'*exploits*

A l'hora de crear un *exploit* per a un sistema remot és important conèixer tots els detalls possibles sobre aquest. L'arquitectura i les mesures de protecció d'aquests sistemes no són iguals i, per descomptat, és possible que el codi que s'executi tampoc no ho sigui. No serà el mateix crear un *exploit* per a un sistema operatiu Windows XP amb el *service pack* 2 en idioma anglès que per a un sistema operatiu Windows XP amb el *service pack* 3 en alemany. Per exemple, l'error anterior relatiu a la càmera web al programa Microsoft MSN Messenger només retornava una consola remota si es tractava d'un sistema operatiu Microsoft Windows 2000 amb el *service pack* 4.

Per a poder crear d'una manera automatitzada *exploits* que funcionin en plataformes similars, s'ha treballat en la creació d'un entorn de treball que permeti al desenvolupador d'exploits, amb petits canvis, crear un *exploit* que funcioni en totes les plataformes vulnerables.

1.2. Metasploit

Metasploit és un projecte de codi obert basat en aquest concepte. Aquesta eina no està pensada únicament per a la creació d'*exploits*, sinó també per a l'execució i prova d'aquests en entorns de tests de penetració.

Pàgina web

Pot ser baixat des de l'URL del projecte i tenen versions per a sistemes Microsoft Windows i sistemes Unix: http://www.metasploit.org/

Metasploit Framework 3.2

Un dels conceptes que cal tenir en compte quan s'utilitza el Metasploit és la diferenciació entre un *exploit* i un *payload*.

L'*exploit* és la part necessària per a l'aprofitament d'un error. És a dir, si l'error fos del tipus en el qual es pot produir un desbordament de memòria intermèdia en un paràmetre, llavors en el Metasploit s'anomenaria *exploit* el programa que aconsegueix desbordar aquest paràmetre i obtenir el control de programa per a executar qualsevol codi.

Per contra, el *payload* serà la part independent del tipus d'error que volem que l'atac executi. És a dir, suposem que volem que després de llançar l'*exploit* un determinat port obtingui una consola en la màquina. En aquest entorn el *payload* serà el codi necessari per a crear la consola que s'executi en aquest port.

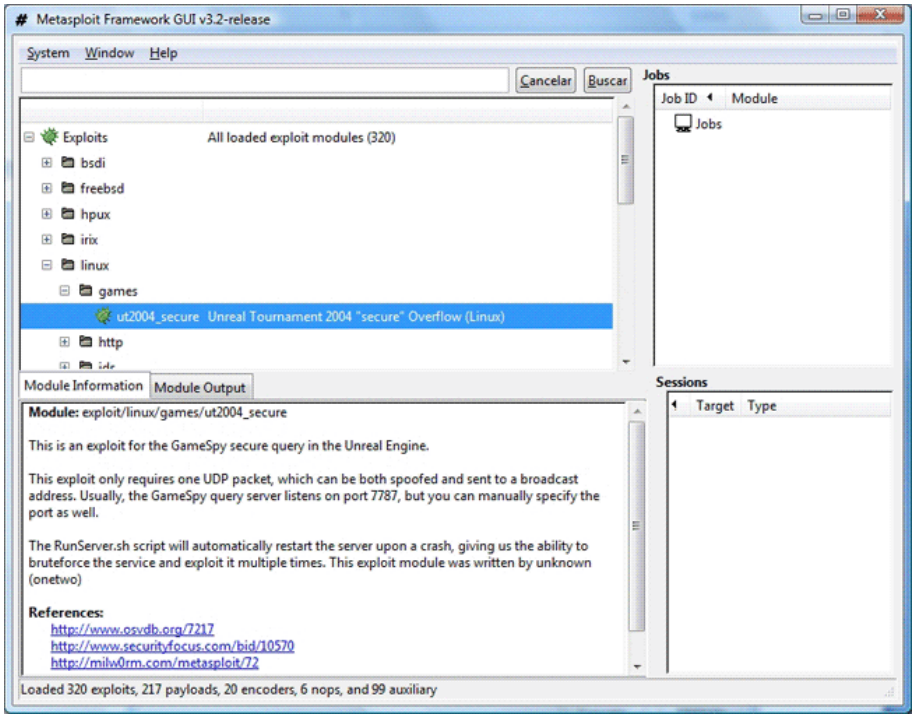
El Metasploit proporciona una llista de *payloads* programats, que poden ser des d'executar una ordre *ADD USER* en un sistema per a aconseguir que es creï un compte d'usuari fins a aconseguir una connexió de control remot per VNC en el sistema vulnerat.

1.3. Un exemple amb el Metasploit

El Metasploit, una vegada unit l'*exploit* en concret i el *payload* adequat, generarà un *exploit* complet que permetrà explotar l'error en un servidor vulnerable amb l'execució del *payload* seleccionat.

Per a veure el funcionament d'aquest entorn d'escriptura d'*exploits* farem una simulació d'aprofitament d'una vulnerabilitat de l'Unreal Tournament 2004 en una plataforma Linux. Aquest programari és un servei de xarxa que permet jugar múltiples usuaris al joc Unreal vulnerable a un error de desbordament de memòria intermèdia.

Per a crear aquest *exploit* n'hi ha prou de desplegar el node d'*exploits*, seleccionar la plataforma vulnerable i triar el tipus de programari vulnerable. És a dir, "Linux", "Games", "Unreal Tournament 2004".



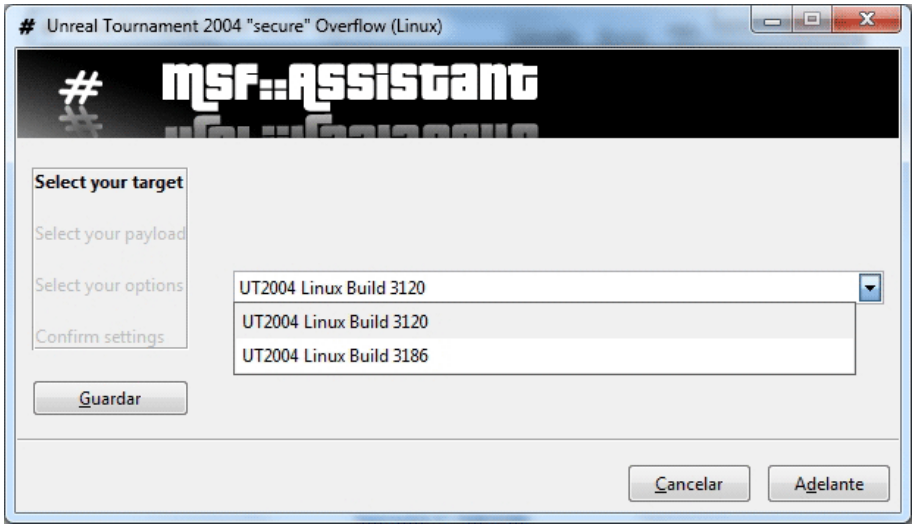
Creació d'*exploit* per a l'Unreal Tournament 2004

Aquest error és explotable mitjançant l'enviament d'un paquet UDP malformat i permet l'execució de codi arbitrari en la màquina vulnerable, amb la qual cosa seria possible des d'executar qualsevol ordre fins a crear una consola utilitzable en remot per l'atacant.

Com es pot veure en la part d'informació del mòdul, no solament tenim la descripció de l'error i el funcionament d'aquest, sinó que a més està acompanyat dels enllaços de consulta a les bases de dades d'expedients de seguretat i fins i tot, en aquest cas, a la publicació d'un *exploit* complet en Milw0rm.com

Per a la creació d'aquest *exploit* d'exemple n'hi hauria prou de fer doble clic sobre l'element en qüestió, és a dir, sobre l'*exploit* segons la terminologia utilitzada pel Metasploit.

Automàticament apareixerà un auxiliar que ens guiarà per la creació de l'*exploit*. En aquest cas, en primera instància ens permet seleccionar la versió vulnerable del programari per al qual es vol construir un *exploit*. Això pot ser perquè, de vegades, el paràmetre vulnerable canvia de mida, ubicació, nom, manera de ser cridat o de ser explotat en diferents versions vulnerables. En aquest exemple, com es pot veure en la figura següent, es fa distinció entre la versió 3120 i la 3186.



Selecció de la versió

Una vegada seleccionada la versió del programari objectiu, haurem de passar a configurar el *payload*, és a dir, el codi que s'ha d'executar després d'haver explotat correctament la vulnerabilitat. Cada *payload* tindrà una sèrie d'opcions diferents per configurar, que seran seleccionades en el tercer pas de l'auxiliar. Una vegada acabat de configurar el *payload*, l'*exploit* ja estarà creat.

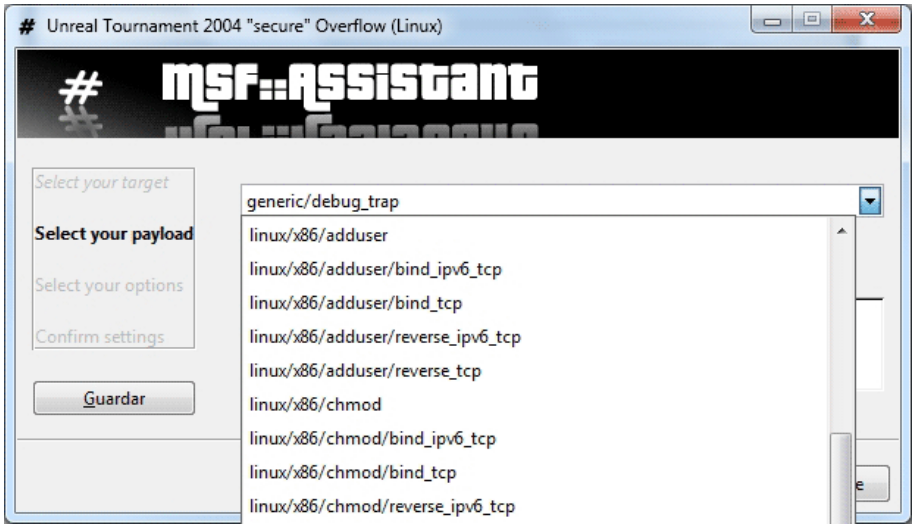
La llista de *payloads* que apareixerà és pròpia de cada sistema operatiu i la proporcionarà automàticament el Metasploit. Sobre les tècniques de postexplotació, és a dir, de quines són les accions per seguir després de l'explotació reeixida d'un error, hi ha molts corrents i tècniques diferents.

Per a sistemes Windows hi ha opcions com la consecució d'una connexió remota per mitjà de VNC o fins i tot la instal·lació d'un mòdul complet de *scripts* automatitzats per a l'execució d'accions en la màquina vulnerada.

El Meterpreter és una biblioteca d'enllaços dinàmics (*.dll*) especialment creada per a l'automatització de *scripts* que s'injecta en la memòria del sistema vulnerat. Permet fer mitjançant *scripts* una multitud d'accions com crear una consola, pujar o baixar fitxers, enumerar processos, matar-los o crear-ne d'altres de nous utilitzant simplement crides a *scripts*.

En sistemes Linux les opcions són menys espectaculars, però no menys efectives. Hi ha l'opció d'executar l'ordre `chmod`, que permet canviar els permisos d'un fitxer concret, enumerar directoris, crear usuaris o executar qualsevol altra ordre.

Mètodes comuns, amb implementacions diferents en cada sistema operatiu, són els d'afegir usuaris al sistema o retornar una consola remota del sistema vulnerat amb els privilegis del compte que utilitza el servei vulnerat.



Selecció del *payload*

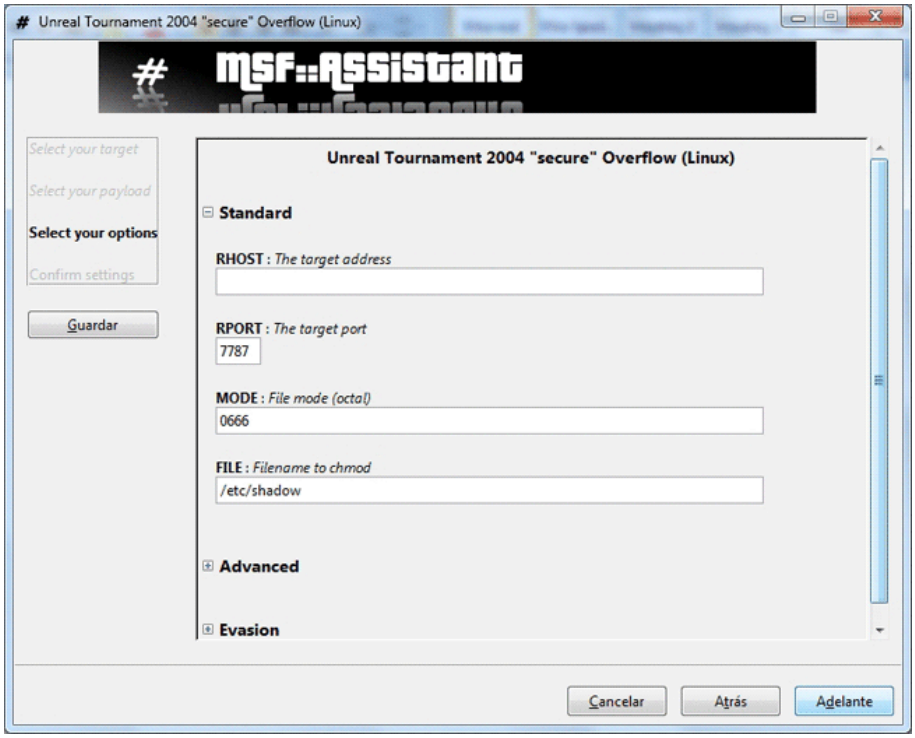
Com es pot apreciar en la figura anterior, la llista de *payloads* disponibles és llarga. Per tant, és una opció perfecta per als creadors d'*exploits*, ja que amb construir la capçalera de l'*exploit* per a un nou error descobert tenen a la seva disposició un volum alt de comportaments ja construïts per a automatitzar-lo.

Per a aquest exemple s'ha seleccionat el *payload* `linux/x86/chmod`. Amb aquest *payload* es podrà canviar, mitjançant l'ús de l'ordre *chmod*, la llista de permisos associats a un fitxer.

Per a acabar la construcció de l'*exploit* seria necessari configurar la informació necessària final de l'*exploit*. En aquest exemple caldria configurar els paràmetres relatius al programari vulnerable, és a dir l'adreça IP de l'ordinador remot, que es configuraria en el camp `RHOST`, i el port pel qual està escoltant el servei vulnerable de l'Unreal Tournament 2004, que es configuraria en el camp `RPORT`.

A més, per descomptat, és necessària la informació referent al *payload*. En aquest cas, el fitxer al qual es volen canviar els permisos, que es configuraria en el camp `FILE` i la nova llista de permisos que es vol establir, que es configuraria en el camp `MODE`.

A més, com es pot apreciar en la figura següent, hi ha característiques avançades que poden ser configurades per a ajustar correctament la construcció de l'*exploit*. Si el servidor tingués alguna configuració especial canviada amb el tipus de codificació o amb l'enviament de paràmetres, podria ser ajustat en aquesta part.



Configuració del *payload*

A més de les opcions avançades, alguns *exploits* vénen acompanyats d'opcions d'evasió. Aquestes opcions d'evasió estan pensades per a canviar les signatures dels atacs i aconseguir que l'enviament d'un d'aquests *exploits* no pugui ser detectar per sistemes de detecció d'intrusions o tallafocs que puguin estar defensant la xarxa.

Com es pot veure en la imatge, aquest *exploit*, mancant ser configurada l'adreça IP del servidor per atacar, canvia els permisos de l'arxiu `/etc/shadow` perquè pugui ser llegit per qualsevol usuari. Aquest arxiu emmagatzema els *hashes* de les contrasenyes dels usuaris i per defecte només pot ser llegit per l'usuari *root*.

Una vegada acabat l'auxiliar, l'*exploit* ja està llest per a ser llançat i esperar els resultats. Com es pot veure, és una eina molt còmoda per a l'ajust i creació d'*exploits* a la carta, però que requereix la creació de les capçaleres dels *exploits* per a estar actualitzada i al dia amb els nous errors descoberts.

1.4. Definició de l'*exploit*

No obstant això, el nostre interès és conèixer com s'ha escrit i com s'ha generat l'*exploit* amb aquesta eina. Si fem un cop d'ull al codi font del mòdul de l'*exploit* podrem accedir a la manera com està generat cadascun d'aquests. N'hi ha prou amb fer clic amb el botó dret i seleccionar l'opció contextual de mostrar el codi font del mòdul.

Si mirem el codi de l'*exploit*, podrem veure que utilitza un llenguatge propi per a treballar amb el Metasploit. No és un llenguatge complex i la seva alta estructuració permet que sigui fàcilment llegit per qualsevol usuari amb certs coneixements tècnics.

A continuació es mostra un extracte de l'*exploit* llançat anteriorment:

```
def exploit

connect_udp

buf = make_nops(1024)
buf[24, 4] = [target['Rets'][1]].pack('V')
buf[44, 4] = [target['Rets'][0]].pack('V')
buf[56, 4] = [target['Rets'][1]].pack('V')
buf[48, 6] = "\x8d\x64\x24\x0c\xff\xe4" #LEA/JMP

buf[0, 8] = "\\secure\\"
buf[buf.length - payload.encoded.length,
payload.encoded.length] = payload.encoded

udp_sock.put(buf)
handler

disconnect_udp

end
```

Extracte d'*exploit* per a l'Unreal Tournament 2004

Com es pot observar, hi ha una funció *exploit* que genera un paquet UDP per a enviar al servidor. Llegint el codi es pot veure com es genera una matriu de *NOP* de 1.024 elements. Aquests *NOP* saturaran la memòria intermèdia del servidor amb instruccions que no fan res, i permetran a l'atacant col·locar el *payload* a la zona de memòria que vulgui.

Com es pot veure, el *payload*, independentment del *payload* triat per l'usuari, es tracta com una seqüència de text que s'introdueix en una matriu que el programador de l'*exploit* col·loca on correspon. En aquest cas, la col·loca al final del paquet UDP que serà llançat al servidor web.

En aquest exemple, a l'inici de la matriu *buf* es col·loca l'*exploit* que farà que el programa busqui les instruccions del *payload* a la zona de memòria que serà carregada amb el final de la variable *buf*, on s'ha col·locat el *payload*.

L'ús de les instruccions *NOP* que s'introdueixen entre la capçalera de l'*exploit* i el *payload* evita haver d'encertar l'adreça exacta de memòria on comença el *payload*. Serà suficient de tenir una idea aproximada de l'adreça, amb un marge d'error de 1.024 instruccions, ja que si el comptador de programa troba una instrucció *NOP*, executarà la instrucció següent. Aquest procés anirà passant d'instrucció *NOP* a instrucció *NOP* fins que arribi a l'adreça de memòria on es troba el *payload*. Aquesta acció és un bon truc quan no se sap exactament l'adreça de memòria.

Una bona metàfora per a explicar això seria imaginar que tenim una pistola d'aigua i volem, des d'una distància d'uns cinc metres, omplir una ampolla. Si apuntem a la boca de l'ampolla, ens resultarà molt complicat omplir-la, i encara més si només disposem d'un tret. En canvi, si col·loquem un embut a la boca de l'ampolla, podrem encertar amb molta més facilitat. Com més gran sigui l'embut que col·loquem, més trigarà l'aigua a arribar a l'ampolla, però més fàcil serà encertar i al final l'aigua arribarà, que és el nostre objectiu. Exactament igual passa amb els *NOP* i el *payload* en aquest exemple.

2. *Exploits* locals

No tots els *exploits* són remots, i l'impacte dels *exploits* locals pot ser d'alta importància. Vam veure com un error explotable en local permetia saltar les proteccions de la consola Wii per a accedir a tot el sistema i poder executar tot el codi que vulguem.

A més, el nombre de programes als quals es pot accedir en local és molt més gran que els oferts per xarxa, i amb la distribució adequada de l'*exploit* es poden convertir en remots igualment.

> Exemples actuals són els *exploits* que es distribueixen localment per mitjà dels dispositius d'emmagatzematge USB. Al vell estil dels virus distribuïts per disquets, molts cucs aprofiten *exploits* locals i la transmissió humana per a distribuir-se massivament. Un exemple d'això ha estat el cuc *Conficker* per als sistemes operatius Windows Vista.

Altres *exploits* es generen arran d'errors que es troben en el maneig dels fitxers d'entrada que maneja un programa.

> Exemples típics d'això són els MP3 maliciosos que exploten una vulnerabilitat en un reproductor d'àudio i una vegada prenen el control es connecten a un servidor i baixen un programari maliciós.

Això s'ha repetit fins a la sacietat amb fitxers ZIP o RAR, amb fitxers de vídeo o àudio Quicktime o MPEG. Usualment els programes emmagatzemen els fitxers en un format de propietat més o menys conegut que es pot analitzar per a descobrir la manera en la qual s'emmagatzemen les dades.

Aquests *exploits* es poden explotar de manera local enviant a un usuari un fitxer malformat perquè l'obri, esperant que usi una versió de programari vulnerable a l'*exploit* desenvolupat. També es pot arribar a explotar aquests errors creant una pàgina web que carregui automàticament el connector d'un programari vulnerable i un fitxer malformat o utilitzant la publicació d'aquests *exploits* amb noms suggeridors a les xarxes d'intercanvis d'arxius *P2P*.

A continuació es mostra un extracte del codi que genera un *exploit* per a les versions inferiors a la 0.9.6 del reproductor *VLC Media Player*. El codi genera un fitxer amb extensió *.rt* que provoca, en el cas de l'*exploit* aquí exposat que s'executi la calculadora de Windows. Executar la calculadora de Windows és un exemple recurrent en les proves de concepte dels sistemes Microsoft Windows, ja que, a més de ser innocu, demostra que és possible executar qualsevol programa.

```
[...]
open(my $rt, "> s.rt");
print $rt "\x3C\x77\x69\x6E\x64\x6F\x77\x20\x68\x65".
          "\x69\x67\x68\x74\x3D\x22\x32\x35\x30\x22".
          "\x20\x77\x69\x64\x74\x68\x3D\x22\x33\x30".
          "\x30\x22\x20\x64\x75\x72\x61\x74\x69\x6F".
          "\x6E\x3D\x22\x31\x35\x22\x20\x62\x67\x63".
          "\x6F\x6C\x6F\x72\x3D\x22\x79\x65\x6C\x6C".
          "\x6F\x77\x22\x3E\x0D\x0A\x4D\x61\x72\x79".
          "\x20\x68\x61\x64\x20\x61\x20\x6C\x69\x74".
          "\x74\x6C\x65\x20\x6C\x61\x6D\x62\x2C\x0D".
          "\x0A\x3C\x62\x72\x2F\x3E\x3C\x74\x69\x6D".
          "\x65\x20\x62\x65\x67\x69\x6E\x3D\x22".
          $char x 72 . $eip . $jmp . $addr . $nop x 12 .
          $shellcode . $char x 1024 .
          "\x22\x2F\x3E\x0D\x0A\x3C\x62\x72\x2F\x3E".
          "\x3C\x74\x69\x6D\x65\x20\x62\x65\x67\x69".
 [...]
```

Extracte d'*exploit* VLC Media Player

En aquest cas podem observar com s'escriu un fitxer al qual s'afegeixen algunes variables com ara `$eip`, `$addr`, `$shellcode` o `$nop` amb valors configurables que permeten que aquest *exploit* sigui adaptat canviant el *shellcode*, les adreces de memòria o el matalàs de *NOP*.

2.1. *Fuzzing*

Descobrir aquest tipus d'errors en programari local sol ser una tasca fàcilment automatitzable. Lluny han quedat les tècniques de verificació de programari basades en casos, ja que el nombre de possibilitats d'entrades que es pot donar avui dia a un programari utilitzat en milions d'equips és tan gran que la verificació basada en casos es queda molt curta.

Per a això s'utilitzen les tècniques de *fuzzing*. Consisteixen a enviar dades aleatòries a les entrades dels programes i llançar tantes dades aleatòries amb la major dispersió i factor d'entropia possibles buscant detectar errors no controlats en el programari.

Així, la manera més habitual de localitzar i provar *exploits* locals com el que s'ha mostrat per al VLC Media Player sol ser utilitzar un codi en Perl, Python o fins i tot C que generi un fitxer amb l'extensió adequada perquè s'obri per defecte amb l'aplicació desitjada.

Aquests fitxers contindran capçaleres vàlides i invàlides, dades parcialment correctes o totalment incorrectes que aniran variant aleatòriament fins que es produeixi un error en l'aplicació. És en aquest moment en el qual es comprova quines dades d'entrada han generat la caiguda o el comportament anòmal del programa per a descobrir un error explotable.

Encara que *a priori* el procés per a explotar una vulnerabilitat de tipus local pugui semblar més complicat, pot arribar a tenir un gran impacte si es descobreix en un programari que s'executi en l'àmplia majoria dels ordinadors, com

és l'Adobe Flash Player o l'Adobe Reader. De fet, el període de vida d'un *exploit* per a aquest tipus d'eines sol ser molt més gran, ja que la majoria d'aquest tipus de programari normalment es troba sense pegats aplicats en els ordinadors.

Al contrari del que ocorre amb el Metasploit, no hi ha entorns de treball de generació d'*exploits* multiplataforma per a aquest tipus d'errors, per la qual cosa cadascun es mostrarà de la manera que el descobridor consideri apropiada.