LinuxFocus article number 282
http://linuxfocus.org

Externe aanvallen

door Eric Detoisien
<valgasu(at)club-internet.fr>

Over de auteur:

Eric Detoisien is een expert in computer beveiliging. Dol als hij is om alles dat te maken heeft met beveiliging, is hij een van de experts van de rstack groep - www.rstack.org -

Vertaald naar het Nederlands door:
GH Snijders
<ghs(at)linuxfocus.org>

Kort:

Dit artikel werd voor het eerst gepubliceerd in een speciale editie van Linux Magazin Frankrijk, gefocust op beveiliging. De redacteur en de autheurs waren zo vriendelijk om LinuxFocus toe te staan om ieder artikel uit deze speciale uitgave te publiceren. Aansluitend, zal LinuxFocus deze presenteren zodra de vertaling naar het Engels gereed is. Met dank aan alle mensen die betrokken zijn bij dit werk. Deze abstract zal gereproduceerd worden voor artikelen van dezelfde oorsprong.

Dit artikel laat verschillende externe aanvallen zien, die een cracker kan gebruiken tegen machines in een netwerk. We zullen de aanval op het netwerk zelf bespreken, een paar aanvallen op applicaties en de Denial Of Service aanvallen.

Netwerk aanvallen

De netwerk aanvallen berusten op zwakheden die direct gerelateerd zijn aan de protollen of de implementatie ervan. Er bestaan er vele. Echter, de meeste zijn varianten van de vijf goed bekende netwerk aanvallen.

Fragment Aanvallen

Deze aanval gaat voorbij de bescherming van IP filters. Voor de implementatie ervan, gebruiken crackers twee verschillende methodes: Tiny Fragments en Fragment overlapping. Deze aanvallen zijn enigzins gedateerd, daar moderne firewalls hier reeds lang mee overweg kunnen.

Tiny Fragments

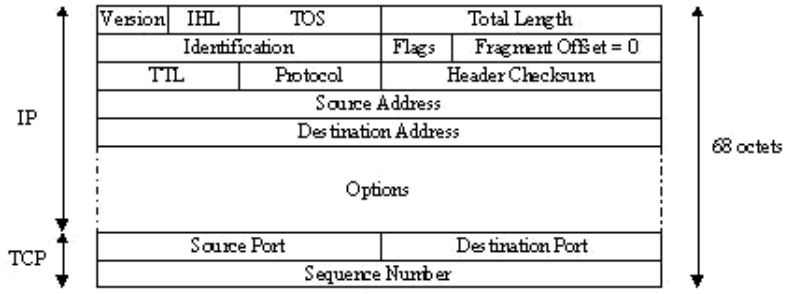
Volgens RFC (*Request For Comment*, vraag om commentaar) 791 (IP), moeten alle Internet nodes (routers) in staat zijn om pakketten van minimaal 68 bytes te transporteren, zonder deze te fragmenteren (op te delen). De minimale grootte van een IP pakket is 20 bytes, zonder opties. Als er opties aanwezig zijn, bedraagt de maximale grootte 60 bytes. Het IHL (*Internet Header Length*) bevat de lengte van het pakket in 32 bit woorden. Dit veld gebruikt 4 bit, het aantal mogelijke waarden is dus 2^4 - 1 = 15 (de waarde 0000 is niet toegestaan). Tenslotte, het *Fragment Offset* veld, welke het begin van de eerste byte van het fragment, in verhouding tot het volledige datagram, aangeeft, is 8 bytes blocks groot. Een data fragment bevat dus minimaal 8 bytes. Dit maakt samen echt 68 bytes.

De aanval bestaat uit het opzetten van een TCP connectie, verdeeld over 2 IP pakketten (gefragmenteerd). Het eerste IP pakket van 68 bytes bevat alleen de eerste 8 bytes van de TCP header (bron en doel poorten en volgorde nummer). De data in het tweede IP pakket bevat dan het TCP connectie verzoek (SYN flag is 1, ACK flag is 0).

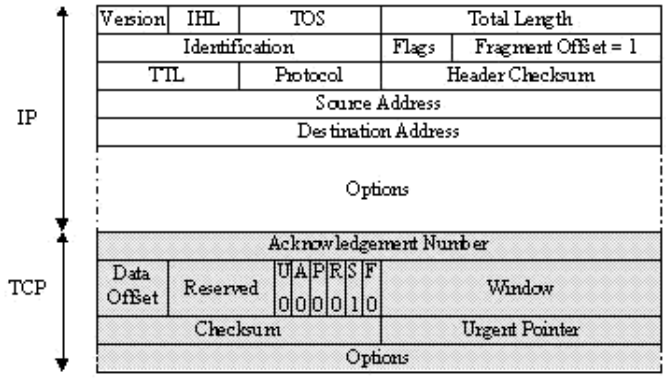
Echter, IP filters passen dezelfde regel toe aan alle fragmenten in een pakket. Het filter van het eerste fragment (Fragment Offset = 0) definieert de regel, waarop deze ook wordt toegepast op de andere fragmenten (Fragment Offset = 1) zonder enige andere controle. Dus, wanneer gedefragmenteerd op IP niveau van de doel machine, wordt het verbindings verzoek opnieuw opgebouwd en doorgegeven aan de TCP laag. De verbinding wordt opgezet, ondanks het IP filter ertussen, welke dit had moeten tegenhouden.

Afbeelding 1 en 2 laten beide fragmenten zien en afbeelding 3 laat het gedefragmenteerde pakket op de doelmachine zien:

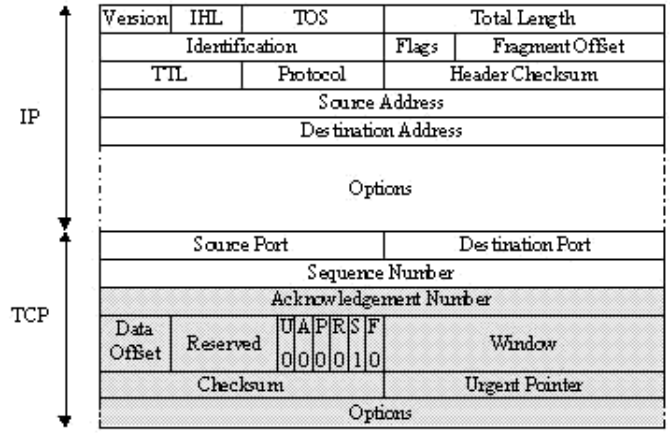
Afb.1: Fragment 1



Afb.2: Fragment 2



Afb.3: Gedefragmenteerd pakket

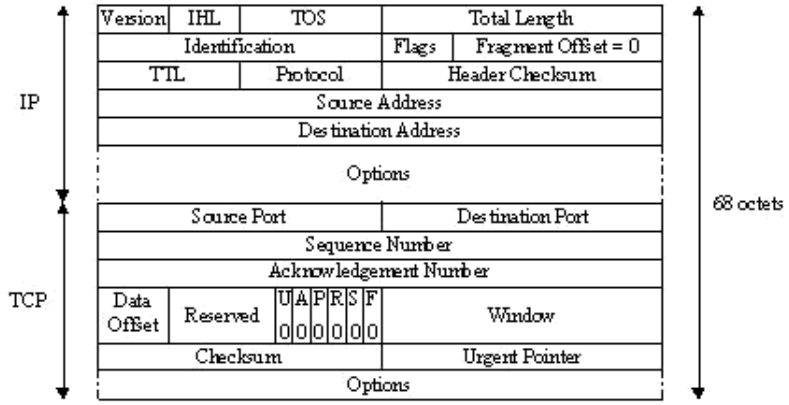


Fragment Overlapping

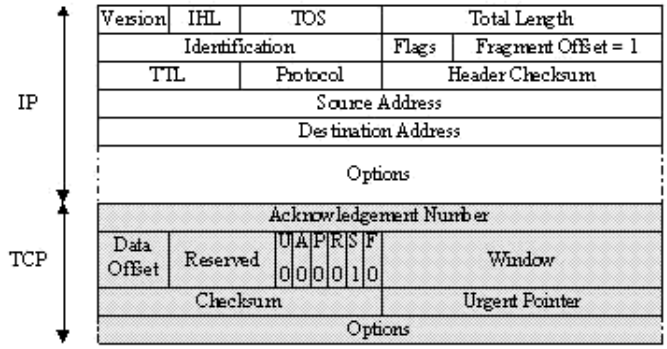
Nog steeds aan de hand van RFC 791 (IP), als twee IP fragmenten overlappen, zal de tweede de eerste overschrijfen. De aanval bestaat uit het verdelen van een IP pakket in twee fragmenten. Het IP filter accepteerd de eerste, welke 68 bytes bevat (zie Tiny Fragments) en daar het geen TCP connectie verzoek bevat (SYN = 0, ACK = 0). Wederom zal deze regel ook worden toegepast aan de volgende fragmenten. Het tweede pakket (Fragment Offset = 1) bevat het echte verbindings verzoek, en wordt geaccepteerd door het IP filter omdat deze niet ziet dat hier een verbinding wordt opgezet. Dus, tijdens het defragmenteren zal de data van het tweede fragment de data van de eerste overschrijven, te beginnen na de 8ste byte (de Fragment Offset = 1). Het opnieuw geassembleerde pakket is dan een geldig verbindings verzoek voor de doelmachine. De connectie wordt opgezet ondanks het IP filter ertussen.

Afbeeldingen 1 en 2 geven beide fragmenten weer en afbeelding 3 laat het gedefragmenteerde pakket op de doel machine zien:

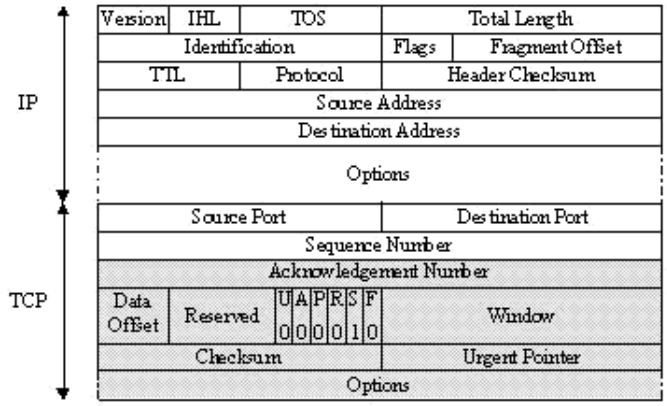
Afb.4: Fragment 1



Afb.5: Fragment 2



Afb.6: Gedefragmenteerd pakket



IP Spoofing

Het doel van deze aanval is om het IP adres van een machine over te nemen. Dit stelt de cracker in staat om ofwel de oorsprong van zijn aanval te verbergen (gebruikt id Denial Of Service aanvallen) of om gebruik te maken van een vertrouwens relatie tussen twee machines. Hier zullen we het tweede gebruik van IP Spoofing bekijken.

Het principe van de aanval bestaat, voor de cracker, uit het vervalsen van zijn eigen IP pakketten (met programma's als `hping2` of `nemesis` in welke hij, onder andere, het bron IP zal veranderen. IP Spoofing wordt ook vaak Blind Spoofing genoemd. De antwoorden van de valse pakketten gaan niet naar de machine van de cracker, daar het bron adres is aangepast. Dus gaan ze naar de gespoofde machine. Er bestaan echter twee methodes om toch antwoorden terug te krijgen:

1. *Source Routing*: Het IP protocol kent een optie genaamd Source Routing (bron omleiding) welke je in staat stelt aan te geven, welke route de IP pakketten zouden moeten nemen. Deze route bestaat uit een serie IP adressen die de pakketten dienen te volgen. Voor de cracker is het voldoende om een route aan te geven die langs een router gaat die onder zijn controle staat. Tegenwoording verwerpen de meeste TCP/IP stack implementaties deze optie;
2. *Re-routing*: router tabellen die gebruik maken van het RIP protocol kunnen veranderd worden door ze RIP pakketten te sturen met nieuwe routing informatie. Dit wordt gedaan om de pakketten om te leiden naar een router die de cracker controleerd.

Deze technieken zijn nauwelijks bruikbaar: de aanval wordt uitgevoerd zonder de pakketten te kennen die van de doelserver afkomstig zijn.

Blind Spoofing wordt gebruikt tegen services als rlogin of rsh. Het authenticatie mechanisme van deze services kijkt alleen maar naat het bron IP adres van de client machine. Deze redelijk bekende aanval (Kevin Mitnick gebruikte het tegen Tsutomu Shimomura's machine in 1994) vereist een paar stappen:

- het IP adres vinden van een vertrouwde machine door, bijvoorbeeld `showmount -e` welke laat zien waar de bestandssystemen worden geëxporteerd, of `rpcinfo` welke meer informatie geeft;
- de vertrouwde host verstoren met een SYN Flooding, bijvoorbeeld (straks meer over Denial Of Service). Dit is van belang om te voorkomen dat de machine antwoorden verstuurd op de pakketten die verstuurd worden door de doel server. Anders zou deze proberen TCP RST pakketten te sturen welke de verbindingspogingen zouden afbreken;
- voorspellen van TCP volgorde nummers: ieder TCP pakket wordt geaccosieerd met een initieel volgorde nummer. De OS TCP/IP stack genereerd deze op een lineare manier, afhankelijk van de tijd random (willekeurig) of pseudo-random, afhankelijk van de gebruikte systemen. De cracker kan alleen systemen aanvallen die voorspelbare volnummers genereren (linieer of tijd-gebaseerd);
- tde aanval bestaat uit het openen van een TCP connectie op de gewilde poort (rsh bijvoorbeeld). Voor een beter begrip zullen we ter herinnering even kijken naar het openings mechanisme van een TCP connectie. Dit gebeurt in drie stappen (TCP Three Way Handshake):
    1. de vrager stuurt een pakket met de TCP SYN flag en een *x* volgnummer naar de doelmachine;
    2. het doel antwoordt met een pakket met de TCP SYN flag en de ACK flag (met een *X=1* bevestigings nummer) geactiveerd. Het volgnummer is *y*;
    3. de vrager stuurd een pakket met de TCP ACK flag (met een *y+1* bevestigingsnummer) terug naar de doelmachine.

Tijdens de aanval, ontvangt de cracker niet de SYN-ACK die door het doel wordt verstuurd. Om de connectie toch op te kunnen zetten, voorspeld hij het *y* volgnummer om een pakket te kunnen sturen met het juiste ACK nummer *y+1*. De connectie wordt vervolgens opgezet met het IP adres als authenticatie. De cracker is nu in staat om commando's te sturen naar de rsh service, zoals `echo ++ >> /.rhosts` om meer toegangsrechten te verkrijgen. Om dit te doen, gebruikt hij een pakket met de TCP PSH flag (*Push*): de ontvangen data wordt dan onmiddelijk naar de volgende laag gestuurd (in dit geval de rsh

service). Vervolgens kan hij inloggen op een service als rlogin of rsh zonder IP Spoofing.

De cracker gebruikt de machine A, terwijl C de vertrouwde machine voorsteld. De A(C) aanduiding betekend dat het pakket is verstuurd vanaf A met het gespoofde IP adres van C. Opmerking: er bestaat een programma met de naam `mendax` welke deze IP Spoofing mechanismes iplementeerd.

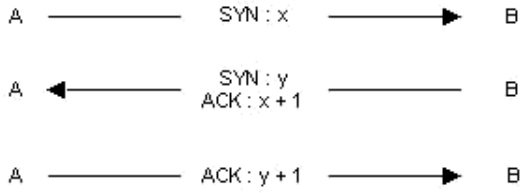
TCP Session Hijacking

TCP Session Hijacking (letterlijk: het kapen van een TCP sessie), staat de cracker toe om een TCP stroom om te leiden. Hiermee kan een cracker een wachtwoord bescherming (zoals in telnet of ftp) omzeilen. De noodzaak van luisteren (sniffen) beperkt deze aanval tot het fysieke netwerk van het doelwit. Alvorens de details te bespreken, zullen we eerst de basis principes van het TCP protocol toelichten.

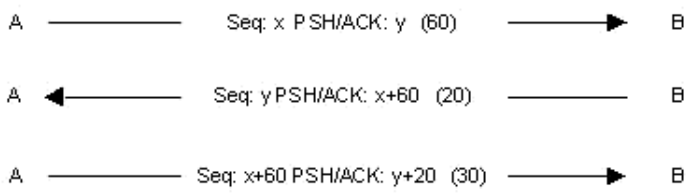
We zullen hier niet het mystery van het TCP protocol uit de doeken doen, maar ons concentreren op de belangrijkste punten, om deze aanval te begrijpen. De TCP header bevat verschillende velden:

- de bron en de doel poort, waarmee de connectie tussen twee machines wordt geïdentificeerd;
- het volgnummer dat iedere verzonden byte aangeeft;
- het bevestigings (ACK) nummer, corresponderend met de laatst ontvangen byte;
- de interessante flags zijn:
  - SYN, welke de volgnummers synchroniseerd zodra er een verbinding wordt opgezet;
  - ACK, de bevestigings (acknowledgement) flag van een TCP segment;
  - PSH, welke de ontvanger verteld de data door te sturen naar de applicatie.

Afbeelding 8 laat zien hoe het opzetten van een TCP connectie
Afb.8: Three Way Handshake



Hier, startte machine A een TCP connectie met machine B.
Afbeelding 9 geeft een doorvoer van TCP data weer:



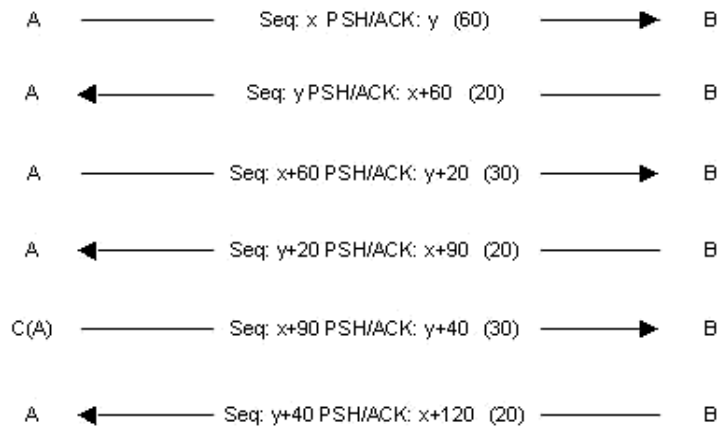
De volgnummers zullen veranderen met het aantal verzonden bytes aan data. Het volgnummer wordt aangeduid met *Seq*(sequentie), het bevestigingsnummer (ACK) kan gevonden worden na de PSH en ACK flags en het aantal verzonden data bytes staat tussen haakjes.

Deze aanval veroorzaakt een staat van desynchronisatie aan beide kanten van de TCP connectie, en maakt het zo mogelijk om de verbinding te kapen. Een connectie raakt gedesynchroniseerd als het volgnummer van de volgende verzonden byte van de A machine afwijkt van het volgnummer van de volgende byte, te ontvangen door B. Andersom ook.

In het voobeeld van afbeelding 9, aan het eind van de eerste stap, al s B z'n pakket ontvangt, wacht A op een pakket met een bevestigings nummer van *x+60*. Als het volgende pakket dat door B wordt verstuurd, niet dit volgummer bevat, worden A en B beschouwd als gedesynchroniseerd.

Dus, een cracker met machine C wil een opgezette Telnet sessie tussen A en B kapen. Eerst luisterd C het telnet verkeer (TCP poort 23) tussen A en B af. Als de cracker denkt dat A de tijd heeft gehad om zich te authenticeren bij de Telnet service op de B machine, desynchroniseerd hij de A machine tegen B. Om dit te doen, creëert hij een pakket met het bron IP adres van A en het TCP bevestigingsnummer dat door B wordt verwacht, wat door de B wordt geaccepteerd. Behalve het verstoren van de TCP verbinding, stelt dit de cracker in staat een commando te injecteren in de telnet sessie die was gestart door A. In feite is dit pakket in staat om data te vervoeren (PSH flag = 1).

Afbeelding 10 geeft deze aanval weer:
Afb.10: TCP Session Hijacking



De B machine accepteerd het commando, verstuurd door C, en bevestigt dit door een pakket naar A te sturen met de ACK flag. Ondertussen, als A een pakket stuurd naar B, wordt het afgewezen omdat het volgnummer afwijkt van degene die door B wordt verwacht.

Een probleem dat vervolgens optreedt: de Ack Storm. Er worden zeer veel ACKs gegenereerd. Dit gebeurd als A een TCP pakket stuurd met een ongeldig volgnummer (A is immers gedesynchroniseerd), B wijst het af en stuurt A een ACK met het volgnummer dat het verwacht. A ontvangt deze ACK, en daar het volgnummer niet klopt met degene die werd verwacht, stuurt hij ook een ACK naar B en B doet het nog eens...

Dit ACK Storm probleem kan worden opgelost als de cracker gebruik maakt van ARP Spoofing. In dat

geval, zal de C machine de ARP cache van B vergiftigen, door deze wijs te maken dat het IP adres van A nu bij het MAC adres van C hoord. Deze technieken worden geïmplementeerd door het `hunt` programma.

ARP Spoofing

Deze aanval, die ook wel ARP Redirect (ARP omleiding) wordt genoemd, leidt het netwerk verkeer van een of meerdere machines naar de machine van de cracker. Het wordt gedaan op het fysieke netwerk van de slachtoffers. Laten we eens kijken hoe wat het ARP protocol ook al weer was en hoe het werkt.

Het ARP protocol (*Address Resolution Protocol*) implementeerd het resolutie mechanisme om een IP adres te vertalen naar een Ethernet MAC adres. Netwerk apparatuur communiceerd door het uitwisselen van Ethernet frames (in een Ethernet netwerk, uiteraard), op de datalink laag. Om in staat te zijn om deze informatie te delen, is het nodig dat de netwerk kaarten over een uniek Ethernet adres beschikken: dit is het MAC adres (*MAC=Media Access Control*).

Zodra er een IP pakket wordt verstuurd, dient de zender het MAC adres van de ontvanger te weten. Om het te krijgen, wordt er een broadcast ARP verzoek vestuurd naar alle machines in het lokale netwerk. Dit verzoek vraagt: "Wat is het MAC adres dat bij dit IP adres hoort?". De machine met dit IP adres antwoordt met een ARP pakket, en vertelt de zender zo het gevraagde MAC adres. Vanaf dat punt weet de zender het MAC adres dat bij het IP adres hoort, waarnaar pakketten verzonden moeten worden. Deze wetenschap wordt een tijdje in een cache (buffer) opgeslagen (om verzoeken te voorkomen, iedere keer dat er een IP pakket wordt verstuurd).

Deze aanval vergiftigd de cache van de doelmachine. De cracker verstuurd ARP antwoorden naar de doel machine, om deze te vertellen dat het nieuwe MAC adres er een is die overeenkomt met een gateway (bijvoorbeeld) IP adres naar het adres van de cracker. De machine van de cracker zal vervolgens al het verkeer ontvangen dat verstuurd wordt naar de gateway. Zo is het dus voldoende voor hem om naar het verkeer te luisteren (en/of aan te passen). Daarna zal hij de pakketten doorsturen naar de echte bestemming, zodat niemand de verandering opmerkt.

ARP Spoofing is bruikbaar als een lokaal netwerk gebruik maakt van switches. Deze sturen de Ethernet farmes naar verschillende poorten (kabels), aansluitend met het MAC adres. Een sniffer kan in zo'n geval nauwelijks frames opvangen, voorbij z'n eigen kabel. Zo staat ARP Spoofing toe om te luisteren naar het verkeer tussen twee machines die zich op verschillende poorten bevinden.

Om een ARP Spoofing aanval te implementeren, kan een cracker gebruik maken van een pakket generator zoals `ARPSpoof` of `nemesis`. Voorbeeld: de "slachtoffer" machine (`10.0.0.171`), z'n default gateway (`10.0.0.1`) en de machine van de cracker (`10.0.0.227`). Voor de aanval is het resultaat van een traceroute:

```
[root@cible -> ~]$ traceroute 10.0.0.1
traceroute to 10.0.0.1 (10.0.0.1), 30 hops max, 40 byte packets
 1  10.0.0.1  (10.0.0.1)  1.218 ms  1.061 ms  0.849 ms
```

En de ARP cache van de doel machine is:

```
[root@cible -> ~]$ arp
Address      HWtype  HWAddress        Flags Mask  Iface
10.0.0.1     ether   00:b0:c2:88:de:65   C        eth0
10.0.0.227   ether   00:00:86:35:c9:3f   C        eth0
```

De cracker gebruikt dan `ARPSpoof`:

```
[root@pirate -> ~]$ arpspoof -t 10.0.0.171 10.0.0.1
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
```

De verzonden pakketten zijn ARP pakketten die de ARP cache van de `10.0.0.171` machine vergiftigen met een ARP Reply (antwoord) welke zeggen dat het MAC adres dat hoort bij `10.0.0.1` nu `00:00:86:35:c9:3f` is.

De ARP cache van de `10.0.0.171` machine wordt:

```
[root@cible -> ~]$ arp
Address      HWtype  HWAddress        Flags Mask  Iface
10.0.0.1     ether   00:00:86:35:c9:3f   C        eth0
10.0.0.227   ether   00:00:86:35:c9:3f   C        eth0
```

Om te controleren dat het verkeer nu via de `10.0.0.227 machine loopt, is het voldoende om een nieuwe traceroute uit te voeren op de 10.0.0.1 gateway:`

```
[root@cible -> ~]$ traceroute 10.0.0.1
traceroute to 10.0.0.1 (10.0.0.1), 30 hops max, 40 byte packets
 1  10.0.0.227  (10.0.0.227)  1.712 ms  1.465 ms  1.501 ms
 2  10.0.0.1  (10.0.0.1)  2.238 ms  2.121 ms  2.169 ms
```

Nu kan de cracker het verkeer tussen 10.0.0.171 en 10.0.0.1 afluisteren. Hij moet niet vergeten om IP routing op zijn 10.0.0.227 te activeren.

DNS Spoofing

Het DNS protocol (*Domain Name Systen*) vertaald een domein naam (bijvoorbeeld www.test.com) in zijn IP adres (bijvoorbeeld 192.168.0.1 en vice versa. Deze aanval gebruikt valse antwoorden om de DNS verzoeken verstuurd door een "slachtoffer". Deze aanval leunt op twee methoden.

DNS ID Spoofing

De header van het DNS protocol bevat een identificatie veld om antwoorden en verzoeken te identificeren. Het doel van de DNS ID Spoofing is om een foutief antwoord terug te sturen op een DNS verzoek, voor de echte DNS server kan

antwoorden. Om dit te doen, dient het ID van het verzoek voorspeld te worden. Lokaal kan dit door eenvoudig het netwerk af te luisteren. Echter, op afstand wordt dit iets lastiger. Er bestaan verschillende methodes:

- iedere mogelijkheid van het ID veld proberen. Niet erg realistisch met 65535 mogelijkheden (dit veld is 16 bit);
- een paar honderd DNS verzoeken in de juiste volgorde sturen. Vanzelfsprekend is deze methode niet echt betrouwbaar;
- een server vinden die voorspelbare IDs genereerd (bijvoorbeeld, ID verhoogt met 1). Dit soort zwakke plekken kan gevonden worden in sommige versies van Bind of op Windows 9x machines.

In ieder geval, het is nodig om eerder te antwoorden dan de echte DNS server. Dit kan bijvoorbeeld door de echte te crashen met een Denial Of Service aanval.

Om te slagen, moet de aanvaller een DNS server (ns.attaquant.com) onder controle met de autoriteit over een domein (attaquant.com). De doel DNS server (ns.cible.com) wordt verondersteld om voorspelbare volgnummers te genereren (verhogen met 1 bij ieder verzoek).

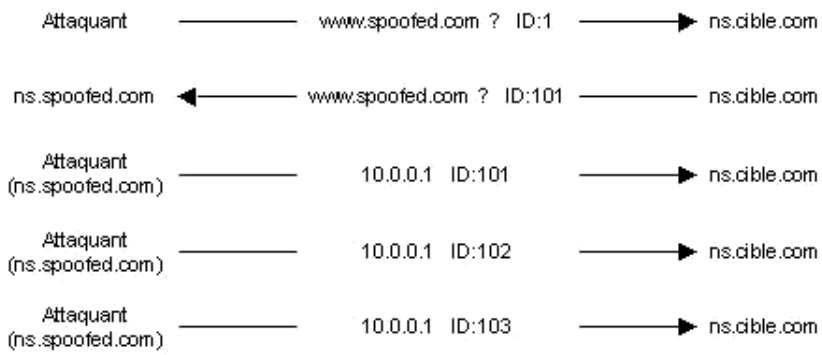
De aanval bestaat uit vier stappen:

1. de aanvaller stuurd een DNS verzoek voor de naam www.attaquant.com naar de DNS server van het cible.com domein, zoals te zien in Afbeelding 11;

   Afb.11: DNS verzoek naar ns.cible.com

2. de doel DNS server stuurd het verzoek naar de DNS van het attaqaunt.com domein;
3. de aanvaller is in staat om het verzoek af te luisteren om het ID te verkrijgen (in ons voorbeeld is het ID 100);
4. de aanvaller past het IP adres aan dat geaccosiërd wordt met een machine naam, hier de "slachtoffer machine" is www.spoofed.com, welke 192.168.0.1 als IP adres. De cracker stuurt een DNS verzoek om de naam www.spoofed.com op te zoeken naar ns.cible.com. Onmiddelijk daarna stuurt hij een zooi aangepaste DNS antwoorden (met als IP adres degene van de site van de aanvaller (10.0.0.1) naar dit zelfde verzoek met het gespoofde bronadres van de DNS server van het spoofed.com domein. Het ID van ieder antwoord zal verhoogd worden met 1, te beginnen met degene die ontvangen werd tijdens het eerste verzoek (ID = 100) om de kans op het krijgen van het juiste ID nummer te verhogen. Dit is voor het geval dat ns.cible.com ondertussen ook al andere DNS verzoeken heeft beantwoord en dus zijn DNS ID heeft verhoogd. Afbeelding 12 geeft deze stappen weer.

   Afb.12: DNS ID Spoofing



De cache van de doel DNS server wordt dan vergiftigd en de volgende machine die vraagt om een resolutie van de www.spoofed.com naam zal het IP adres van de aanvaller krijgen en zal doorgestuurd worden naar zijn site. Dit laatste kan een "kopie" zijn van de echte site om de internet gebruikers te neppen en vertrouwelijke informatie te stelen.

DNS Cache Poisoning

DNS servers gebruiken een cache om lokaal antwoorden op eerdere verzoeken een poosje te bewaren. Dit is om te voorkomen dat er tijd wordt verspilt met het altijd ondervragen van de DNS server die de autoriteit over het gevraagde domein heeft. Dit tweede type van DNS Spoofing bestaat uit het vergiftigen van deze cache met valse informatie. Hier is een voorbeeld:

We behouden de parameters van het vorige voorbeeld. Hier zijn de verschillende stappen van de aanval:

- stuur een DNS verzoek om de naam www.attaquant.com te resolven naar de DNS server van het cible.com domein;
- de doel DNS server stuurd een verzoek om de www.attaquant.com naam op te zoeken naar de DNS server van de aanvaller;
- de DNS server van de aanvaller stuurt een antwoord met aangepaste records, welke hem toestaan om een machine naam aan een IP adres toe te wijzen dat bij de aanvaller hoort. Bijvoorbeeld, de www.cible.com site zou een aangepast DNS record kunnen hebben, en het IP adres van de www.attaquant.com bevatten in plaats van de echte.

Applicatie aanvallen

Applicatie aanvallen richten zich op specifieke zwakken plekken gevonden in applicaties. Een aantal kan echter worden samengevat op type.

Het configuratie probleem

Een van de eerste beveiligings problemen die gevonden kan worden in applicaties komt van configuratie fouten. Er zijn twee soorten van fouten: standaard installatie en foutieve configuratie.

Software, zoals web servers, met standaard installatie bestanden bevatten vaak voorbeeld sites die door crackers gebruikt kunnen worden om vertrouwelijke informatie te benaderen. Zo kunnen ze bijvoorbeeld scipts gebruiken om de bron data te krijgen via slechte dynamische pagina's. Verder, kan zulk een installatie een remote beheer interface bevatten met een standaard login/wachtwoord (te vinden in de applicatie beheer handleiding). De cracker is dan in staat om te veranderen wat hij wil op de site.

De belangrijkste zwakheden die door een slechte configuratie worden gegenereerd zijn toegangslijsten (ACLs) met slechte parameters. Zo kan de cracker prive pagina's of databases benaderen.

Als een klassiek voorbeeld van misconfiguratie, de fouten in Lotus Domino web server parameters komen veel voor. Na de installatie bevatten de Lotus configuratie parameters geen toeganslijst (Access Control List). Het ligt voor de hand dat als de *names.nsf* Lotus database kan worden benaderd met een web browser, zonder authenticatie, het mogelijk is om heel veel informatie op te doen zoals de Lotus gebruikersnamen.

Bugs

Slecht programmeren van software leidt altijd tot bugs. Dit zijn de belangrijkste zwakke plekken. Zodra ze ontdekt worden, staan ze toe dat er ongeauthoriseerde commando's worden uitgevoerd, om de broncode van dynamische pagina's te krijgen, om een service niet langer beschikbaar te maken, om de controle over machine te verkrijgen, etc. De bekendste en meest interressante in termen van misbruik is de buffer overflow.

Buffer overflow

De buffer overflow is een zwakke plek die wordt veroorzaakt door slecht geschreven software. Het steekt de kop op, op het moment dat er een variabele als een argument voor een functie wordt gekopiëerd naar een buffer zonder dat de grootte ervan wordt gecontroleerd. Als de variabele groter is dan de geheugen ruimte die is gereserveerd voor deze buffer, is het genoeg om de buffer overflow te laten gebeuren. De exploitatie ervan vindt plaats door de variabele een deel van een programma mee te geven. Als een cracker succes heeft met deze aanval, krijgt hij de mogelijkheid om op afstand commando's uit te voeren op de doelmachine met de rechten van de aangevallen applicatie. Meer hierover in de series over veilig programmeren:

- Beveiligings gaten vermijden tijdens het ontwikkelen van een applicatie - deel 1
- Beveiligings gaten vermijden tijdens het ontwikkelen van een applicatie - deel 2: geheugen, stack en functies, shellcode
- Beveiligings gaten vermijden tijdens het ontwikkelen van een applicatie - deel 3: buffer overflows
- Beveiligings gaten vermijden tijdens het ontwikkelen van een applicatie - deel 4: format strings
- Beveiligings gaten vermijden tijdens het ontwikkelen van een applicatie - deel 5: race conditions
- Beveiligings gaten vermijden tijdens het ontwikkelen van een applicatie - Deel 6: CGI scripts

Scripts

Slecht programmeren van scripts tast vaak de beveiliging van een systeem aan. Er zijn manieren om zwakke plekken in Perl scripts te gebruiken om bestanden uit de web root te lezen of niet-geauthoriseerde commando's uit te voeren. Deze programmeer problemen worden besproken in de CGI script artikelen hierboven (Deel 6).

Man in the Middle

Het belangrijkste doel van deze aanval is om het verkeer tussen twee machines om te leiden. Dit is voor het onderscheppen, veranderen of vernietigen van data die circuleert tijdens de communicatie. Deze aanval is meer een concept dan een echte aanval. Er bestaan verschillende aanvallen die het principe van Man in the Middle toepassen, zoals de DNS Man in the Middle welke DNS Spoofing gebruikt om het verkeer tussen een web server en een web client af te tappen. Een recente applicatie is opgedoken om SSH verkeer af te tappen.

Denial of service

Deze aanval is goed bekend daar het zal leiden tot het niet langer beschikbaar zijn van een service (specifieke applicatie) of van een doel machine. We onderscheiden twee soorten Denial Of Service: aan de ene kant zijn er diegenen die een bug in een applicatie exploiteren, aan de andere kant diegenen die gebruik maken van een slechte implementatie van een protocol, of zwakheden van een protocol.

Applicatie denial of service

Als de zwakke plekken van een applicatie kunnen leiden tot de mogelijkheid van het overnemen van de controle over een machine (bijvoorbeeld met een buffer overflow), kunnnen ze ook leiden tot het onbeschikbaar raken van een service door een gebrek aan bronnen of door een crash.

Netwerk denial of service

Er bestaan verschillende soorten van Denail of Service met behulp van protocollen die onderdeel uitmaken van de TCP/IP stack.

SYN Flooding

We hebben reeds gezien dat een TCP connectie wordt opgezet in drie stappen (TCP Three Way Handshake). SYN Flooding exploiteert dit mechanisme. De drie stappen bestaan uit het zenden van een SYN, het ontvangen van een SYN-ACK en een ACK sturen. Het idee is om op de doelmachine een grote hoeveelheid TCP connecties te laten

wachten. Om dit te doen, stuurt de cracker heel veel verbindingsverzoeken (SYN flag = 1), de doel machine stuurt een SYN-ACK terug als antwoord op de ontvangen SYN. De cracker zal niet antwoorden met een ACK, en dus zal de doelmachine voor iedere ontvangen SYN een wachtende TCP verbinding hebben. Daar deze half-open connecties geheugen gebruiken, zal na een tijdje de doelmachine vollopen en geen nieuwe verbindingen meer accepteren. Dit type Denial Of Service treft alleen de doelmachine.

De cracker gebruikt een SYN Flooder zoals een synk4, geeft de doel TCP poort aan en gebruikt random bron IP adressen om te voorkomen dat de aanvallende machine geïdentificeerd kan worden.

UDP Flooding

Deze Denial Of Service gebruikt de niet-verbonden modus van het UDP protocol. Er wordt een UDP Packet Storm (veel UDP pakketten) gestart, tegen een enkele machine of tegen twee machines. Een dergelijke aanval tussen twee machines leidt tot verstoppingen in het netwerk, en het opraken van beschikbare bronnen op beide machines. De verstopping is belangrijker, daar UDP verkeer voorrang heeft op TCP verkeer. Het TCP protocol beschikt over een mechanisme om om te gaan met verstoppingen, voor het geval de bevestiging van een pakket na lange tijd arriveerd: dit mechanisme past de frequenty aan van het zenden van TCP pakketten, en verminderd deze. Het UDP protocol beschikt niet over een dergelijk mechanisme: na een tijdje zal het UDP verkeer de volledige bandbreedte opslokken, en slechts een klein deel overlaten voor het TCP verkeer.

Het bekendste voorbeeld van UDP Flooding is de *Chargen Denial of Service* aanval. De implementatie van deze aanval is eenvoudig: het is voldoende om een communicatie op te zetten tussen de Chargen service van een machine en de echo service van een andere. De chargen service genereerd karakters, de echo service de ontvangen data terugstuurd. De cracker stuurt dan UDP pakketten naar poort 19 (chargen) naar een van de "slachtoffers", met het gespoofde IP adres van en bronpoort van de ander. In dit geval zal de bronpoort de UDP poort 7 (echo) zijn. De UDP Flooding leidt tot verzadigen van de beschikbare bandbreedte tussen beide machines. Een volledig netwerk kan het slachtoffer zijn van een UDP FLooding.

Packet Fragment

De Denial of Service van het Pakket Fragment type gebruikt de zwakheden van sommige TCP/IP stacks betreffende de IP defragmentatie (opnieuw assembleren van IP fragmenten.

Een bekende aanval die dit gebruikt is Teardrop. De fragmentatie offset van het tweede segment is kleinder dan de grootte van de eerste en dus zal de offset toegevoegd worden aan de grootte van de eerste. Dit betekend dat het eerste fragment de tweede bevat (overlap). Tijdens defragmentatie, kunnen sommige systemen deze uitzondering niet aan, en kan dit leiden tot het wegvallen van service. Er bestaan varianten van deze aanval: bonk, boink en newtear bijvoorbeeld. De Ping of Death Denial of Service exploiteerd slecht management van ICMP defragmentatie, door meer data te sturen dan de maximale grootte van een IP pakket. Deze verschillende soorten van Denial of Service leiden tot een crash van de doel machine.

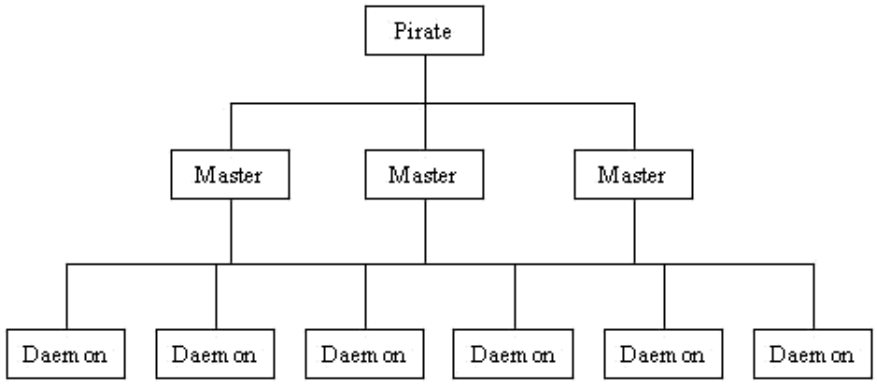
Smurfing

Deze aanval gebruikt het ICMP protocol. Wanneer een ping (ICMP Echo bericht) wordt verstuurd naar een broadcast adres (bijvoorbeeld 10.255.255.255), wordt deze naar alle machines in het netwerk verstuurd. Het principe van de aanval bestaat uit spoofen van de ICMP ECHO REQUEST pakketten, met de doelhost als bron adres. De cracker stuurt een continue ping stroom naar het netwerk broadcast adres en alle machines zullen antwoorden met een ICMP ECHO REPLY bericht. De stroom wordt dan vermenigvuldigd met het aantal hosts in het netwerk. In dat geval zal de doelhost worden getroffen door de Denial of Service, daar het vele verkeer dat deze aanval genereert tot een verstopping van het netwerk kan leiden.

Distributed denial of service

De Distributed Denial of Service (verdeelde Dos, DDos) verzadigd het aangevallen netwerk. Het idee is om veschillende bronnen (daemons) te gebruiken voor de aanval en masters om ze onder controle te houden. De bekendste DDoS tools zijn Tribal Flood Network (TFN), TFN2K, Trinoo en Stacheldraht. Afbeelding 13 toont een typisch DDoS netwerk:

Afb.13: DDoS netwerk



De cracker gebruikt masters om eenvoudig controle te houden over de bronnnen. Uiteraard, dient hij een connectie (TCP) op te zetten naar de masters om de aanval in te stellen en voor te bereiden. De masters sturen alleen commando’s naar de bronnen via UDP. Zonder de masters zou de cracker moeten connecten naar alle bronnen. De origine van de aanval zou dan veel eenvoudiger achterhaald kunnen worden en de implementatie zou veel meer tijd in beslag nemen.

Iedere daemon en master spreken elkaar door specifieke berichten uit te wisselen, die afhankelijk zijn van de gebruikte tool. Deze communicaties kunnen ook versleuteld en/of geauthenticeerd zijn. Om de daemons te installeren en de masters, gebruikt de cracker bekende zwakke plekken (buffer overflows op services als RPC, FTP, enzovoort). De aanval zelf is een SYN Flooding of een Smurf aanval. Het resultaat van zo’n Denial of Service aanval is om een volledig netwerk onbereikbaar te maken.

Conclusie

Vandaag de dag wordt de beveiliging tegen externe aanvaller sterker en sterker, maar helaas geld dit niet voor de interne beveiliging. Deze "arme relatie" van beveiliging tegen crackers laat nog steeds aantrekkelijke perspectieven open voor

lokale aanvallen, zoals TCP Sessie Hijacking, ARP Spoofing en DNS Spoofing. Verder, het voorspellen van volgnummers (het hart van IP Spoofing) en de Fragment aanval varianten duiken alleen maar op door bugs in de uitrusting van het OS. Wat applicatie aanvallen betreft, hebben ze goede tijden in het vooruitzicht, met het oog op de groeiende complexiteit van web gerelateerde applicaties en het korter worden van de deadlines voor ontwikkelaars en beheerders. De Denial of Service aanval zal angstaanjagend blijven in zijn gedistribueerde vorm, zolang gebruikers falen het belang in te zien van de beveiliging vna hun machines.

Links

- RFC 1858 - Beveiligings overwegingen voor IP Fragment Filtering: sunsite.dk/RFC/rfc/rfc1858.html
- IP Spoofing Uitgelegd - Phrack 48 www.phrack.org/
- Eenvoudige actieve aanval tegen TCP - Laurent Joncheray: www.insecure.org/stf/iphijack.txt
- DNS ID Hacking - ADM Crew: packetstorm.securify.com/groups/ADM/ADM-DNS-SPOOF/ADMID.txt
- DoS Project’s "trinoo" - David Dittrich: staff.washington.edu/dittrich/misc/trinoo.analysis
- Het vreemde verhaal van de DENIAL OF SERVICE aanvallen tegen GRC.COM: GRC.COM: grc.com
- hping2: www.kyuzz.org/antirez/hping.html
- nemesis: www.packetfactory.net/Projects/nemesis/
- mendax: packetstorm.securify.com/Exploit_Code_Archive/mendax_linux.tgz
- hunt: lin.fsid.cvut.cz/~kra/index.html
- dsniff: www.monkey.org/~dugsong/dsniff/
- fragrouter: packetstorm.securify.com/UNIX/IDS/fragrouter-1.6.tar.gz

---

Site onderhouden door het LinuxFocus editors team © Eric Detoisien "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org	Vertaling info: fr --> -- : Eric Detoisien <valgasu(at)club-internet.fr> fr --> en: Georges Tarbouriech <gt(at)linuxfocus.org> en --> nl: GH Snijders <ghs(at)linuxfocus.org>

2005-01-14, generated by lfparser_pdf version 2.51