Angriffe von außen

by Eric Detoisien
<valgasu(at)club−internet.fr>

About the author:

Eric Detoisien ist IT−Sicherheitsexperte, leidenschaftlich an allen Gebieten des Themas Sicherheit interessiert und einer der Fachleute der rstack−Gruppe − www.rstack.org −

Abstract:

Dieser Artikel ist in einer Sonderausgabe zum Thema Sicherheit im *Linux Magazine France* erschienen. Der Herausgeber, die Autoren und die Übersetzer haben es freundlicherweise gestattet, daß alle Artikel dieser Ausgabe in loser Folge in LinuxFocus publiziert werden. So bietet Ihnen LinuxFocus diese Artikel einen nach dem anderen an, je nach ihrer Übersetzung ins Englische. Vielen Dank allen Personen, die sich mit dieser Arbeit beschäftigen. Diese Zusammenfassung wird vor einem jeden dieser Artikel wiederholt.

Der Artikel beschreibt die verschiedenen Angriffsmöglichkeiten von außen, die ein Pirat gegen ein Rechnernetzwerk benutzen kann. Wir kommen bei diesem Thema auf prinzipielle Netzattacken, auf Angriffe durch Programmlücken und auf Denial−of−Service−Attacken zu sprechen.

Angriffe auf ein Netzwerk

Netzattacken stützen sich auf Verwundbarkeiten, die direkt an die Protokolle oder an ihre Umsetzung gebunden sind. Davon gibt es eine große Zahl. Dennoch sind die meisten von ihnen bloß Varianten der fünf heutzutage bekanntesten Angriffe.

Fragment−Angriffe

Dieser Angriff umgeht den von IP−Filteranlagen angebotenen Schutz. Um ihn durchzuführen, benutzen Piraten zwei Methoden; die „Tiny Fragments“ und das „Fragment Overlapping“. Diese Attacken haben eine lange Geschichte, die heute verwendeten Firewalls

berücksichtigen sie längst.

Tiny Fragments

Gemäß RFC (*Request For Comment*) 791 (IP) müssen alle Internet−Knoten (Router) Pakete von 68 Bytes Größe übertragen können, ohne sie der Optimierung wegen aufzusplitten. Tatsächlich beträgt die minimale Headergröße eines IP−Paketes 20 Bytes, wenn keine Optionen vorhanden sind, ansonsten ist das Maximum 60 Bytes. Das IHL−Feld (*Internet Header Length*) enthält die Headergröße, gemessen in 32−Bit−Worten. Dieses Feld belegt 4 Bits, die Zahl der möglichen Werte ist 2^4 − 1 = 15 (es darf den Wert 0000 nicht annehmen). Die Maximalgröße des Headers beträgt also wirklich 15*4 = 60 Bytes. Schließlich wird das Feld *Fragment Offset*, das den Abstand des ersten Bytes des Fragments vom vollständigen Datagramm angibt, in Blöcken zu 8 Byte geschrieben. Ein Datenfragment ist also mindestens 8 Bytes lang. Damit sind wir genau bei einer Gesamtsumme von 68 Bytes angekommen.

Die Attacke besteht darin, eine TCP−Verbindungsanfrage auf zwei IP−Pakete aufzuteilen. Das erste von 68 Bytes Größe enthält an Daten lediglich die ersten 8 Bytes des TCP−Headers (Herkunfts− und Zielport sowie die Sequenznummer). Die Daten des zweiten IP−Pakets enthalten folglich die TCP−Verbindungsanfrage (Flag SYN auf 1 und Flag ACK auf 0).

Nun wenden IP−Filter aber dieselbe Filterregel auf alle Fragmente eines Pakets an. Das Filtern des ersten Fragments (Fragment Offset gleich 0) bestimmt also die Regel, die auf die anderen (mit Fragment Offset gleich 1) angewandt wird, ohne jede weitere Art der Kontrolle. Bei der Zusammensetzung auf der IP−Ebene im Zielrechner wird so die Verbindungsanfrage wieder aufgebaut und an die TCP−Ebene geleitet. Die Verbindung kommt dann trotz des IP−Filters zustande.

Die Abbildungen 1 und 2 zeigen die beiden Fragmente, und die Abbildung 3 zeigt das auf dem Zielrechner zusammengesetzte Paket:

Abb. 1: Fragment 1

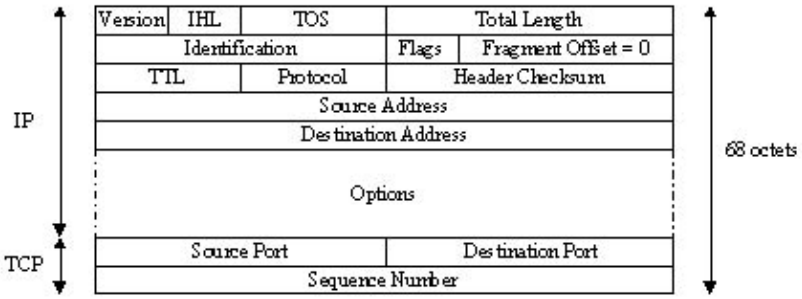


Abb. 2: Fragment 2

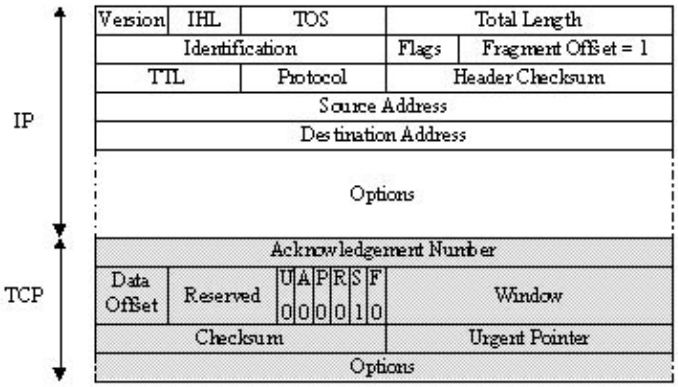
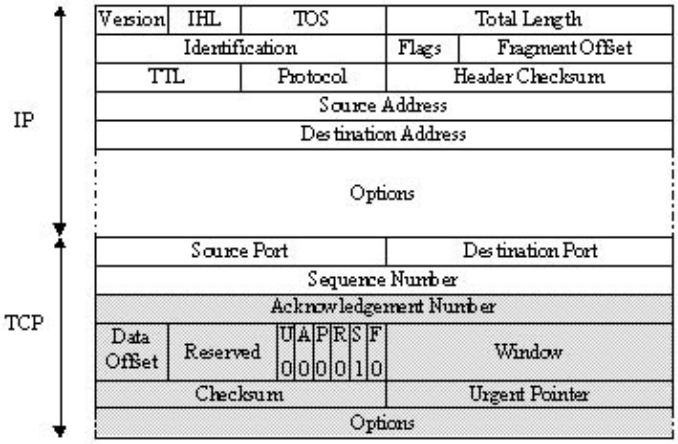


Abb. 3: Zusammengesetztes Paket



Fragment Overlapping

Abermals gemäß RFC 791 (IP) löscht, wenn sich zwei IP−Fragmente überlagern, das zweite das erste. Die Attacke besteht darin, ein IP−Paket in zwei Fragmenten zu bilden. Der IP−Filter akzeptiert das erste von 68 Byte Länge (siehe Tiny Fragments), weil es keine TCP−Verbindungsanfrage enthält (Flags SYN und ACK = 0). Diese Akzeptanzregel wird wie zuvor auf weitere Fragmente des Pakets angewandt. Das zweite (mit einem Fragment Offset von 1), das die tatsächlichen Verbindungsdaten enthält, wird nun vom IP−Filter akzeptiert. Bei der Defragmentierung löschen so die Daten des zweiten Pakets diejenigen des ersten vom Ende des achten Bytes an (denn der Fragment Offset ist gleich 1). Das zusammengesetzte Paket stellt folglich für den Zielrechner eine gültige Verbindungsanfrage dar. Die Verbindung kommt trotz des IP−Filters zustande.

Die Abbildungen 4 und 5 zeigen die beiden Fragmente, und die Abbildung 6 zeigt das auf dem Zielrechner zusammengesetzte Paket:

Abb. 4: Fragment 1

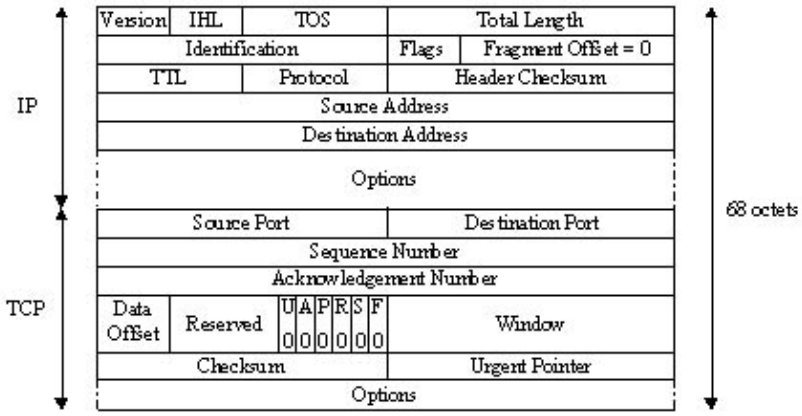


Abb. 5: Fragment 2

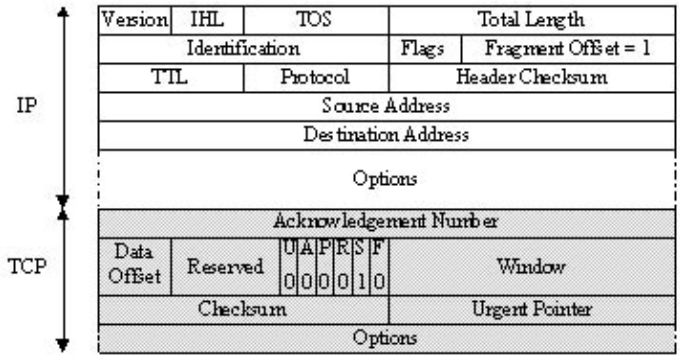
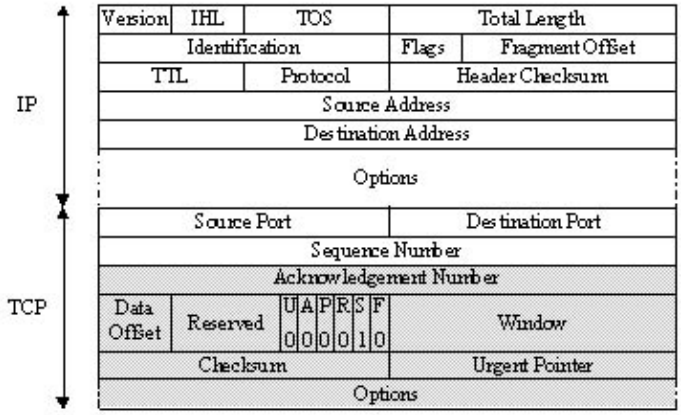


Abb. 6: Zusammengesetztes Paket



IP–Fälschung

Das Ziel dieser Attacke ist der Klau der IP–Adresse einer Maschine, der es dem Piraten ermöglicht, die Herkunft seines Angriffs zu verschleiern (wird in den Denial–of–Service–Attacken gemacht, über die wir später sprechen). Oder er kann Nutzen aus einer vertraulichen Verbindung zweier Rechner ziehen. Darüber werden wir an dieser Stelle also sprechen.

Das zugrundeliegende Prinzip dieses Angriffs besteht darin, daß der Pirat die eigenen IP–Pakete umformt (mit Programmen wie `hping2` oder `nemesis`) zu solchen, in denen er u.a. die IP–Herkunftsadresse ändert. Die IP–Fälschung wird auch oft ein „blinder Angriff“ („Blind Spoofing“) genannt. In der Tat können mögliche Antworten der versandten Pakete nicht auf der Maschine des Piraten ankommen, da die Quelle ja verfälscht ist. Sie richten sich also an den vorgetäuschten Rechner. Dennoch gibt es zwei Methoden, die Antworten zu erhalten:

1. das *Source Routing*: das I–Protokoll bietet eine Option namens Source Routing, die die Spezifikation des Weges der IP–Pakete ermöglichen. Dieser Weg ist eine Folge von Router–IP–Adressen, die die Pakete benutzen müssen. Es genügt dem Piraten, für die Rückkehr der Pakete einen Weg zu einem von ihm kontrollierten Router anzugeben. Heutzutage weisen die meisten Implementationen des TCP/IP–Stacks Pakete mit dieser Option zurück;
2. den *Wiederversand*: Router–Tabellen, die das Routing–Protokoll RIP verwenden, können geändert werden, indem man ihnen RIP–Pakete mit neuen Routing–Informationen schickt mit dem Ziel, die Pakete noch einmal über einen Router zu senden, den der Pirat beherrscht.

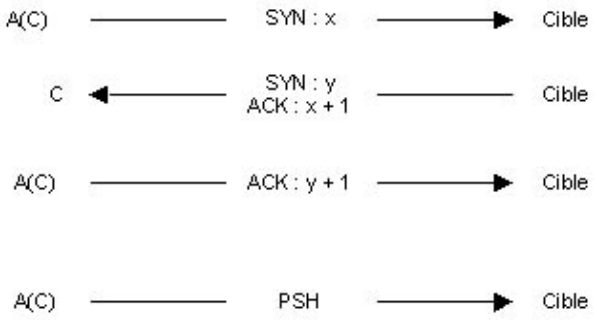
Diese Techniken sind nicht mehr (oder nur noch mühsam) benutzbar: die Attacke wird daher durchgeführt, ohne die Pakete zu kennen, die vom Zielrechner emittiert werden.

Das Blind Spoofing wendet sich gegen Dienste vom Typ rlogin oder rsh. Deren Authentifikationsmechanismus gründet sich nämlich einzig auf die IP−Absenderadresse des Client−Rechners. Dieser recht gut bekannte Angriff (vor allem dank Kevin Mitnick, der ihn 1994 gegen den Rechner von Tsutomu Shimomura benutzte) vollzieht sich in mehreren Etappen:

- Bestimmung der IP−Adresse der Maschine, die Vertrauen genießt, z.B. unter der Verwendung von `showmount -e`, das zeigt, wohin Dateisysteme exportiert werden, oder von `rpcinfo`, das zusätzliche Informationen liefert;
- Außer−Gefecht−Setzen des vertrauten Rechners, etwa per SYN−Überschwemmung (kommt später in diesem Artikel bei den Denial−of−Service−Attacken). Das ist notwendig, damit dieser Rechner nicht mehr auf Pakete antworten kann, die vom Zielrechner verschickt werden. Andernfalls würde er TCP RST−Pakete aussenden, die das Ende des Verbindungsversuchs bedeuteten;
- Vorhersage der TCP−Sequenznummern: jedem TCP−Paket ist eine initiale Sequenznummer zugeordnet. Der TCP/IP−Stack des Betriebssystems erzeugt sie linear, zeitabhängig oder (pseudo−)zufällig, je nach System. Der Pirat kann die Attacke nur auf solchen Systemen durchführen, die vorhersagbare Sequenznummern generieren (also linear oder zeitabhängig);
- Der Angriff besteht im Einleiten einer TCP−Verbindung auf dem gewünschten Port (z.B. rsh). Zum besseren Verständnis vergegenwärtigen wir uns den Mechanismus der Einleitung einer TCP−Verbindung. Sie wird in drei Schritten vorgenommen („TCP Three Way Handshake“):
    1. Der Anfragende schickt ein Paket, welches das TCP SYN−Flag und eine Sequenznummer *x* enthält, an den Zielrechner;
    2. Letzterer antwortet mit einem Paket, dessen TCP SYN− und ACK−Flags (mit der Bestätigungsnummer *x+1*) gesetzt sind. Dessen Sequenznummer sei *y*;
    3. Der Anfragende schickt ein Paket zurück an den Zielrechner, welches das TCP ACK−Flag (mit der Bestätigungsnummer *y+1*) umfaßt.

Während der Attacke erhält der Pirat das vom Zielrechner geschickte SYN−ACK nicht. Damit die Verbindung hergestellt werden kann, schätzt er die Sequenznummer *y*, um ein Paket mit der richtigen ACK−Nummer (*y+1*) zu schicken. Die Verbindung wird dank der Authentifikation per IP−Adresse eingerichtet. Der Pirat kann so dem rsh−Dienst einen Befehl wie `echo ++ >> /.rhosts` schicken, um weitere Rechte zu erlangen. Dafür baut er ein Paket mit dem Flag TCP PSH (*Push*): die empfangenen Daten werden unmittelbar an die höhere Ebene weitergereicht, hier an den rsh−Dienst, damit der sie verarbeitet. Es ist dem Piraten dann möglich, sich direkt über einen Dienst wie rlogin oder rsh ohne IP−Fälschung mit dem Rechner zu verbinden.

Die Abbildung 7 faßt die Schritte der IP−Fälschung zusammen:
Abb. 7: IP−Fälschung, angewendet auf den Dienst rsh



Der Pirat benutzt den Rechner A, während C die vertraute Maschine bezeichnet. Die Notation A(C) bedeutet, daß das Paket von A mit der gefälschten IP−Adresse von C geschickt wird. Erwähnenswert ist die Existenz des Programms `mendax`, das diese verschiedenen Mechanismen der IP−Fälschung in die Tat umsetzt.

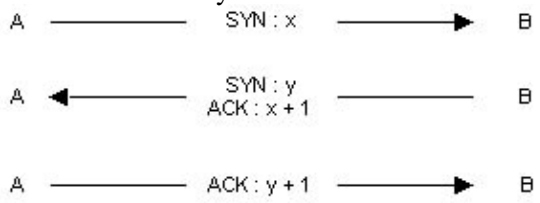
TCP Session Hijacking

Das „Entführen einer TCP−Verbindung“ ermöglicht die Umleitung eines TCP−Flusses. Ein Pirat kann dadurch einen Paßwortschutz (wie telnet oder ftp) umgehen. Die Notwendigkeit des passiven Lauschens (*sniffing*) beschränkt den Einsatzbereich dieser Attacke auf das physische Netzwerk des Zielrechners. Bevor wir in die Details gehen, werden wir einige fundamentale Prinzipien des TC−Protokolls erklären.

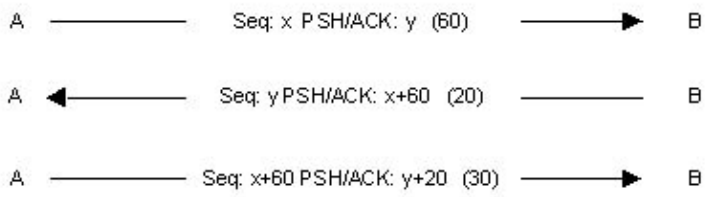
Wir werden hier nicht die Geheimnisse des TC−Protokolls offenbaren, sondern lediglich die essentiellen Punkte zum Verständnis des Angriffs präzisieren. Der TCP−Header besteht aus mehreren Feldern:

- dem Herkunfts− und dem Zielport, um die Verbindung zwischen zwei Maschinen zu bestimmen;
- der Sequenznummer, die jedes der gesendeten Bytes bestimmt;
- der Bestätigungsnummer, die mit der Bestätigungsnummer des letzten empfangenen Bytes korrespondiert;
- den uns interessierenden Flags:
    - ♦ SYN, das die Sequenznummern während eines Verbindungsaufbaus synchronisiert;
    - ♦ ACK, das Flag zur Bestätigung eines TCP−Segments;
    - ♦ PSH, das dem Empfänger anweist, die Daten der Applikation heraufzugeben.

Die Abbildung 8 stellt einen TCP−Verbindungsaufbau schematisch dar (Three Way Handshake):
Abb. 8 : Three Way Handshake



In diesem Fall hat Rechner A eine TCP−Verbindung zum Rechner B angefordert. Dementsprechend zeigt die Abb. 9 einen TCP−Datentransfer:
Abb. 9 : Datenaustausch:



Die Sequenznummern werden sich je nach der Bytezahl der gesendeten Daten ändern. Die Sequenznummer wird durch *Seq* dargestellt, die Bestätigungsnummer findet sich nach den PSH− und ACK−Flags, und die Bytezahl der gesendeten Daten steht in Klammern.

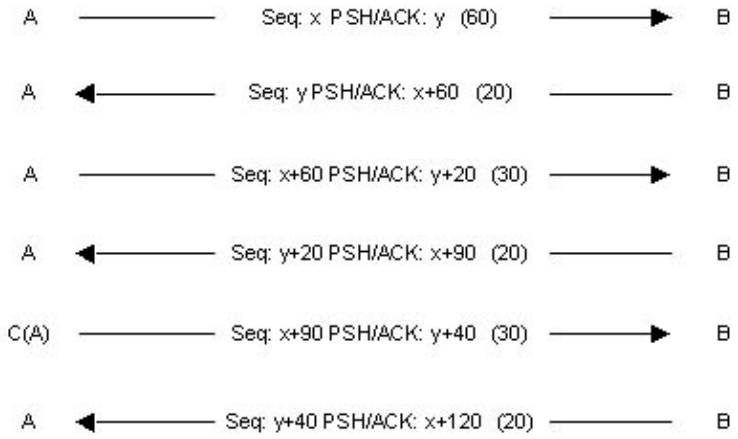
Dieser Angriff erzeugt einen Zustand der Desynchronisation auf jeder Seite der TCP−Verbindung, der den Diebstahl der Sitzung ermöglicht. Eine Verbindung wird desynchronisiert, wenn die Sequenznummer des

nächsten vom Rechner A geschickten Bytes sich von derjenigen des nächsten vom Rechner B empfangenen Bytes unterscheidet. Und umgekehrt gibt es ebenfalls eine Desynchronisation, wenn die Sequenznummer des nächsten von B geschickten Bytes sich von derjenigen des nächsten von A empfangenen Bytes unterscheidet.

Im Beispiel der Abbildung 9 erwartet A am Ende der ersten Etappe, wenn B sein Paket empfangen hat, eines mit der Bestätigungsnummer *x+60* Wenn das nächste Paket, das B schickt, eine andere Nummer enthält, werden A und B demnach als desynchronisiert bezeichnet.

Ein konkreter Fall: Ein Pirat mit dem Rechner C will eine zwischen A und B hergestellte Telnet−Sitzung stehlen. Zuerst belauscht C den Telnet−Verkehr zwischen A und B. Sobald der Pirat glaubt, daß A genug Zeit hatte, um sich bei dem Telnet−Dienst von B zu authentifizieren, desynchronisiert er A in seiner Kommunikation mit B, weswegen er ein Paket mit der IP−Quelladresse von A und der von B erwarteten TCP−Bestätigungsnummer schnürt. B akzeptiert daraufhin dieses Paket. Außer der Desynchronisation der TCP−Verbindung gestattet es dem Piraten, ein Kommando durch die zuvor von A initiierte Telnet−Sitzung zu geben. Denn das Paket kann Daten transportieren (PSH−Flag gleich 1).

Die Abbildung 10 faßt den Angriff zusammen:
Abb. 10: TCP Session Hijacking



B akzeptiert bereitwillig das von C geschickte Kommando, er quittiert den Paketempfang also mit einem Paket an A mit dem ACK−Flag. Wenn unterdessen A ein Paket an B geschickt hat, wird es nicht akzeptiert, da seine Sequenznummer nicht die von B erwartete ist.

Dann taucht ein Problem auf: der ACK−Sturm. Dabei handelt es sich um eine Masse ACKs, die erzeugt werden. Sie erscheinen, wenn A ein TCP−Paket mit ungültiger Sequenznummer schickt (denn A ist desynchronisiert), B es verwirft und A ein ACK mit der erwarteten Sequenznummer schickt. Auf seiner Seite empfängt A dieses ACK, und da die Sequenznummer nicht mit der erwarteten übereinstimmt, schickt er seinerseits ein ACK an B, und B wiederholt das Ganze...

Das Problem des ACK−Sturms kann gelöst werden, wenn der Pirat die ARP−Fälschung benutzt. In diesem Fall verfälscht er den ARP−Cache von B, indem er ihm sagt, daß die IP−Adresse von A von nun an zur MAC−Adresse von C gehört. Diese verschiedenen Techniken werden vom Programm `hunt` implementiert.

ARP−Fälschung

Dieser Angriff, auch ARP−Umleitung genannt, leitet den Netzverkehr einer oder mehrerer Maschinen zum Piratenrechner. Er vollzieht sich im physischen Netz des Opfers. Vorher erinnern wir uns des Nutzen und der Funktionsweise des AR−Protokolls.

Das AR−Protokoll (*Address Resolution Protocol*) führt den Vorgang der Auflösung einer IP−Adresse in eine MAC−Ethernetadresse durch. Geräte in einem Netzwerk kommunizieren, indem sie auf der Datenverbindungsebene Ethernet−Päckchen austauschen (im Fall eines Ethernet−Netzes natürlich). Um diese Informationen austauschen zu können, müssen die Netzwerkkarten eine einzigartige Adresse auf Ethernet−Ebene besitzen; dabei handelt es sich um die MAC−Adresse (*Media Access Control*).

Sobald ein IP−Paket verschickt werden muß, braucht der Absender die MAC−Adresse des Empfängers. Dafür wird eine ARP−Anfrage an alle Maschinen des physischen lokalen Netzes rundgesendet. Diese Anfrage sagt: „Welche MAC−Adresse gehört zu dieser IP−Adresse?“. Die Maschine mit der zugehörigen IP−Adresse antwortet mit einem ARP−Paket, das dem Fragenden die gesuchte MAC−Adresse angibt. Von da an besitzt der Absender die MAC−Adresse, die mit der IP−Zieladresse des zu sendenden Pakets übereinstimmt. Diese Übereinstimmung wird eine bestimmte Zeitlang in einem Puffer gespeichert (um zu vermeiden, für jedes zu sendende Paket eine neue Anfrage stellen zu müssen).

Dieser Angriff korrumpiert den Puffer des Opfers. Der Pirat schickt ARP−Antwortpakete zum Zielrechner, die besagen, daß die neue MAC−Adresse, die mit der IP−Adresse eines Gateways (z.B.) übereinstimmt, die seinige sei. Der Piratenrechner empfängt demzufolge den ganzen Verkehr, der an das Gateway gerichtet wird, er braucht ihm dann also nur noch passiv zuzuhören (oder ihn zu modifizieren). Er leitet danach die Pakete zum tatsächlichen Bestimmungsort weiter.

Das ARP−Fälschen nützt dann, wenn ein lokales Netzwerk Switches benutzt. Diese leiten Ethernet−Päckchen gemäß der MAC−Adresse auf die verschiedenen Ports um. Es ist also einem Lauscher unmöglich, Päckchen jenseits seiner physischen Leitung zu fangen. Das ARP−Fälschen gestattet es damit, Verkehr zwischen Rechnern an verschiedenen Leitungen auf der Ebene des Switches abzuhören.

Um einen Angriff mittels ARP−Fälschung in die Tat umzusetzen, wird der Pirat einen ARP−Paketgenerator wie `ARPSpoof` oder `nemesis` benutzen. Sei das Opfer `10.0.0.171`, sein Standard−Gateway `10.0.0.1` und der Piratenrechner `10.0.0.227`. Vor dem Angriff gibt ein Traceroute folgendes Resultat:

```
[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
```

Und der ARP−Cache des Zielrechners ist:

```
[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
```

Der Pirat startet dann `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
```

Was da geschickt wird, sind ARP−Pakete, die den ARP−Cache der Maschine `10.0.0.171` mit ARP−Antworten verfälschen. Diese behaupten, daß die mit `10.0.0.1` assoziierte MAC−Adresse nunmehr `00:00:86:35:c9:3f` sei.

Fortan ist der ARP−Cache der Maschine `10.0.0.171`:

```
[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
```

Um zu prüfen, ob der Verkehr jetzt über die Maschine `10.0.0.227` läuft, genügt es, erneut eine Traceroute zum Gateway `10.0.0.1` aufzurufen:

```
[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
```

Der Pirat kann künftig den Verkehr von `10.0.0.171` zu `10.0.0.1` abhören. Er darf nicht vergessen, das IP−Routing auf seinem Rechner `10.0.0.227` zu aktivieren (mittels IP−Forwarding − aktiviere den Schlüssel `net.ipv4.ip_forward` in der `/etc/sysctl.conf` oder nimm `fragrouter`).

DNS−Fälschung

Das DNS−Protokoll (*Domain Name System*) konvertiert einen Domänennamen (z.B. www.test.com) in seine IP−Adresse (z.B. `192.168.0.1`) und umgekehrt, d.h. eine IP−Adresse in ihren Domänennamen. Der Angriff besteht darin, falsche Antworten auf DNS−Anfragen, die ein Opfer schickt, ankommen zu lassen. Es gibt zwei prinzipielle Methoden, um das durchzuführen.

DNS ID−Fälschung

Der Header des DNS−Protokolls beinhaltet ein Identifikationsfeld, das es erlaubt, die Antworten passend zu den Anfragen zu machen. Ziel der DNS ID−Fälschung ist es, vor dem DNS−Server eine falsche Antwort auf eine DNS−Anfrage zu geben. Dafür muß er die ID der Anfrage vorwegnehmen. Lokal ist das einfach: man belauscht das Netz. Aus der Ferne gestaltet es sich komplizierter. Doch es gibt mehrere Methoden:

- das Ausprobieren aller Möglichkeiten des ID−Feldes. Diese Methode ist nicht sehr realistisch, da es 65535 Möglichkeiten dafür gibt (weil das Feld mit 16 Bits kodiert wird);
- das Senden von ein paar hundert DNS−Anfragen nacheinander − recht naheliegend, aber kaum zuverlässig;
- das Finden eines Servers, der vorhersehbare IDs erzeugt (z.B. Erhöhung der ID um 1), diese Lücke gibt es bei manchen Bind−Versionen und auf Windows 9x−Rechnern.

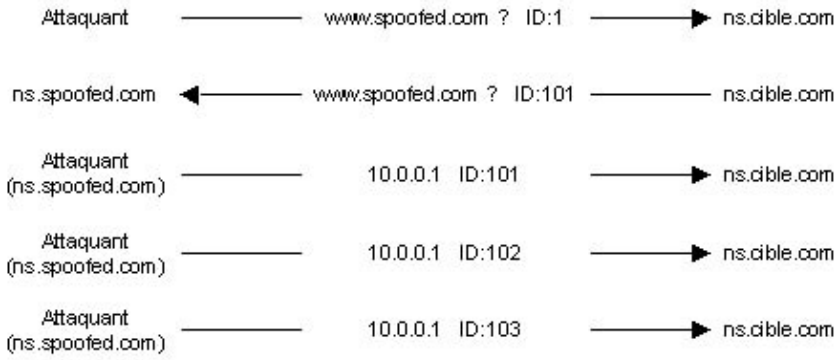
In allen Fällen ist es nötig, vor dem DNS−Server zu antworten und ihn außer Gefecht zu setzen − z.B. mit einem Denial−of−Service−Angriff.

Um sein Ziel zu erreichen, muß der Angreifer einen DNS−Server kontrollieren (`ns.attaquant.com`), der wiederum die Domäne `attaquant.com` beherrscht. Der Ziel−DNS−Rechner (`ns.cible.com`) habe

vorhersagbare Sequenznummern (Erhöhung um 1 bei jeder neuen Anfrage).

Die Attacke gliedert sich in vier Schritte:

1. Der Angreifer schickt eine DNS−Anfrage für den Namen `www.attaquant.com` an den DNS−Server der Domäne `cible.com`, wie Abbildung 11 zeigt;
   Abb. 11: Senden einer DNS−Anfrage an `ns.cible.com`
   

2. Der Ziel−DNS−Server hat daraufhin die Anfrage an den DNS der Domäne `attaquant.com` übertragen;
3. Der Angreifer ist in der Lage, die Anfrage abzuhören, um ihre ID zu bekommen (in unserem Beispiel eine ID von 100);
4. Der Angriff verfälscht die IP−Adresse, die einem Rechnernamen zugeordnet ist, hier ist das Opfer `www.spoofed.com`, das normalerweise die IP−Adresse `192.168.0.1` hat. Der Pirat emittiert eine DNS−Anfrage zur Namensauflösung von `www.spoofed.com` an `ns.cible.com`. Sofort danach sendet er eine Menge falscher DNS−Antworten (die als IP−Adresse diejenige des Angreifers, `10.0.0.1`, angeben) auf ebendiese Anfrage, wobei er als IP−Quelladresse diejenige des DNS−Servers der Domäne `spoofed.com` vortäuscht. Die ID einer jeden Antwort wird um 1 inkrementiert, beginnend mit derjenigen, die beim zweiten Schritt ermittelt wurde (ID = 100) − um die Chance zu vergrößern, daß darunter die richtige Nummer der ID−Antwort fällt, falls `ns.cible.com` auf weitere Anfragen geantwortet und so seine DNS−ID erhöht hat. Die Abbildung 12 zeigt die Details dieses Schrittes. Abb. 12: DNS ID−Fälschung
   

Der Cache des DNS−Zielservers ist damit verfälscht, und die nächste Maschine, die eine Auflösung des Namens `www.spoofed.com` verlangt, bekommt die IP−Adresse des Piratenrechners. Sie wird auf dessen Seite umgeleitet, die eine Kopie der echten Seite sein könnte, um Internet−Nutzer zu täuschen und ihnen vertrauliche Informationen zu entlocken.

DNS Cache−Verfälschung

DNS−Server haben einen Cache, der während einer bestimmten Zeit lokal die Antworten auf vergangene Anfragen speichert, um zu vermeiden, daß jedes Mal zeitaufwendig der Nameserver der gewünschten Domäne gefragt werden muß. In der zweiten Art der DNS−Fälschung wird dieser Cache mit falschen Informationen gefüllt. Hier ein Beispiel von Cache−Fälschung:

Gleiche Umstände wie beim vorherigen Beispiel. Das sind die verschiedenen Stufen des Angriffs:

- Eine DNS−Anfrage zur Auflösung des Namens `www.attaquant.com` wird zum DNS−Server der Domäne `cible.com` geschickt;
- Der DNS−Zielserver sendet folglich eine Anfrage zur Auflösung des Namens `www.attaquant.com` an den DNS−Server des Angreifers;
- Der DNS−Server des Angreifers schickt eine Antwort mit falschen Einträgen, durch die er einen Namen zu einer von ihm kontrollierten IP−Adresse zuordnen kann. Zum Beispiel könnte die Seite `www.cible.com` einen falschen DNS−Eintrag haben, die die IP−Adresse von `www.attaquant.com` statt der richtigen IP−Adresse liefert.

Angriffe über Applikationen

Angriffe über Applikationen stützen sich im wesentlichen auf Lücken, die es in verwendeten Programmen gibt. Dennoch können bestimmte Attacken in Klassen eingeordnet werden.

Konfigurationsprobleme

Eines der ersten Sicherheitsprobleme, das Applikationen verursachen, ist das der Konfigurationsfehler. Wir unterscheiden zwei Arten von Fehlern: die Standard−Installationen und die schlechten Konfigurationen im eigentlichen Sinne.

Programme wie Webserver haben, wenn sie standardmäßig installiert sind, häufig Beispielseiten, die von den Piraten genutzt werden können, um an vertrauliche Informationen zu kommen. Zum Beispiel kann es dort Skripte geben, die es ermöglichen, die Quellen von dynamischen Seiten zu erhalten oder Informationen über das benutzte System. Darüber hinaus ist bei einer solchen Installation evtl. standardmäßig ein Interface zur Fernadministration mit einem Login/Paßwortschutz verfügbar (zu finden im Administrationshandbuch der Applikation). Der Pirat hat also die Seite zu fassen und kann sie nach seinem Gutdünken verändern.

Die meisten Fehler, die aus einer schlechten Konfiguration entstehen, sind Zugriffslisten mit falschen Parametern. Der Pirat hat dann Zugang zu privaten Seiten und Datenbanken.

Ein klassisches Beispiel von Konfigurationsproblemen sind die häufigen Fehler in den Parametern des Webservers Lotus Domino. Denn während der Installation dieses Servers sind die Konfigurationsdatenbanken von Lotus ohne jegliche Zugriffsliste zugänglich. Wenn konkret die Lotus−Datenbank *names.nsf* über einen Web−Browser ohne Authentifikation zugänglich ist, kann man dadurch zahlreiche Informationen erhalten, etwa die Namen sämtlicher Lotus−Nutzer. Diese Datenbank ist nur ein Beispiel, und Lotus Domino enthält davon eine beträchtliche Zahl mit sensiblen Daten.

Bugs

Schlechte Programmierung von Software führt zwangsläufig zu *Bugs*. Sie sind die Quelle der wichtigsten Sicherheitslücken. Wenn diese Verwundbarkeiten entdeckt werden, erlauben sie es, nicht autorisierte Befehle auszuführen, den Quellcode dynamischer Seiten zu bekommen, einen Dienst auszuschalten, den Rechner zu übernehmen etc. Die bekanntesten dieser Bugs und die interessantesten, was ihre Ausbeutung betrifft, sind die des Typs *Buffer Overflow*.

Buffer Overflow

Der Speicherüberlauf ist eine Lücke, an der schlechte Programmierung schuld ist. Ein Buffer Overflow ereignet sich dann, wenn eine Variable einer Funktion als Argument übergeben und in einen Speicherplatz kopiert wird, ohne daß ihre Größe geprüft wird. Es genügt, daß die Variable größer als der reservierte Speicherbereich ist, damit ein Buffer Overflow entsteht. Er wird ausgebeutet, indem in der Variable ein Programmfragment abgelegt wird, das fähig ist, freundlicherweise eine Shell starten zu lassen, falls hier ein Buffer Overflow aufgetreten ist. Falls der Pirat Erfolg mit diesem Angriff hat, erhält er also ein Mittel, aus der Ferne Kommandos auf dem Zielrechner auszuführen mit den Zugriffsrechten der angegriffenen Applikation. Der Buffer Overflow und seine Nutzung sind Gegenstand einer Serie von Artikeln über die sichere Programmierung:

- Vermeiden von Sicherheitslöchern beim Entwickeln einer Applikation – Teil 1
- Vermeiden von Sicherheitslöchern beim Entwickeln einer Applikation – Teil 2: Speicher, Stack und Funktionen, shellcode
- Vermeiden von Sicherheitslöchern beim Entwickeln einer Applikation – Teil 3: Buffer Overflow
- Vermeiden von Sicherheitslöchern beim Entwickeln einer Applikation – Teil 4: format strings
- Vermeiden von Sicherheitslöchern beim Entwickeln einer Applikation – Teil 5: Race Conditions
- Vermeiden von Sicherheitslöchern beim Entwickeln einer Applikation – Teil 6: CGI−Skripte

Skripte

Schlechte Programmierung von Skripten schlägt sich oft auf die Systemsicherheit nieder. Tatsächlich gibt es Wege, Lücken in Perl−Skripten auszunutzen, die es gestatten, Dateien aus dem Web−Wurzelverzeichnis zu lesen oder unerlaubte Programme auszuführen. Auf diese programmiertechnischen Probleme wird in einem der o.a. Artikel hingewiesen, der von den Fallen der Entwicklung in Perl und PHP handelt (Teil 6, „CGI−Skripte“).

Der Mann in der Mitte

Ziel dieses Angriffs ist die Umleitung des Verkehrs zwischen zwei Maschinen, um die im Laufe der Kommunikation übertragenen Daten abzuhören, zu verändern oder zu vernichten. Es handelt sich dabei mehr um ein Konzept als um eine vollständige Attacke. Es gibt verschiedene Angriffe, die das Prinzip vom Mann in der Mitte umsetzen, etwa der *DNS Man in the Middle*, der DNS−Fälschung benutzt, um den Verkehr zwischen einem Client und einem Webserver umzulenken. Ebenso wurde kürzlich eine Applikation erstellt, um SSH−Verkehr umzuleiten.

Denial−of−Service

Dieser Angriff trägt seinen Namen zu Recht, denn er führt zur Ausschaltung eines Dienstes (einer bestimmten Applikation) oder eines Zielrechners. Wir unterscheiden zwei Arten von Denial−of−Service; die eine hat ihren Ursprung in der Ausnutzung eines Programmfehlers, die andere rührt von einer schlechten Implementation eines Protokolls oder dessen Schwächen her.

Denial−of−Service bei Applikationen

Genau wie Lücken in einer Applikation die Möglichkeit der Kontrollübernahme eines Rechners nach sich ziehen (z.B. Buffer Overflow), können sie auch einen Denial−of−Service herbeiführen. Die Anwendung steht also nicht mehr zur Verfügung, weil die ihr zugeordneten Ressourcen nicht mehr ausreichen oder weil sie abgestürzt ist.

Denial−of−Service bei Netzwerken

Es gibt verschiedene Typen von Denial−of−Service, die die Protokollspezifikationen des TCP/IP−Stacks benutzen.

SYN−Überschwemmung

Wir haben gesehen, daß sich eine TCP−Verbindung in drei Phasen aufbaut (TCP Three Way Handshake). Die SYN−Überschwemmung nutzt diesen Mechanismus aus. Die drei Etappen sind die Aussendung eines SYN, der Empfang eines SYN−ACK und die Aussendung eines ACK. Das Prinzip ist, den Zielrechner auf eine große Zahl von TCP−Verbindungen warten zu lassen. Dafür schickt der Pirat sehr viele Verbindungsanfragen (SYN−Flag auf 1), der Zielrechner antwortet darauf mit SYN−ACKs. Der Pirat antwortet niemals mit einem ACK, und somit reserviert der Zielrechner für jedes empfangene SYN eine TCP−Verbindung. Weil diese halboffenen Verbindungen Speicherressourcen verbrauchen, ist der Rechner nach einer Weile belegt und kann keine Verbindungen mehr annehmen. Diese Art von Denial−of−Service beeinflußt nur den Zielrechner.

Der Pirat benutzt einen SYN−Flooder wie synk4, wählt den anvisierten TCP−Port und ordnet die Verwendung zufälliger IP−Absenderadressen an, um jede Identifikation des Piratenrechners zu vermeiden.

UDP−Überschwemmung

Dieser Denial−of−Service nutzt den nichtverbundenen Modus des UD−Protokolls aus. Er erzeugt einen „UDP Packet Storm“ (große Menge von UDP−Paketen), entweder in Richtung auf einen Rechner oder zwischen zwei Rechnern. Eine solche Attacke zwischen zwei Maschinen hat eine Netzüberlastung zur Folge sowie eine Ressourcensättigung der beiden Opfer. Die Überlastung ist wichtiger, da der UDP−Verkehr Priorität gegenüber dem TCP−Verkehr hat. Wirklich besitzt das TC−Protokoll einen Mechanismus der Überlastungskontrolle; falls die Bestätigung eines Paketes erst nach einer langen Verzögerung eintrifft, paßt er die Häufigkeit der TCP−Paketsendungen an, der Absatz sinkt. Das UD−Protokoll hat keinen solchen Mechanismus, nach einer Zeitlang besetzt der UDP−Verkehr die ganze Bandbreite und läßt nur eine winzige Menge TCP−Verkehr zu.

Das bekannteste Beispiel von UDP−Überschwemmung ist der *Chargen Denial of Service Attack*. Seine praktische Umsetzung ist einfach, es reicht aus, den Dienst `chargen` eines Rechners mit dem Dienst `echo` eines anderen kommunizieren zu lassen. `chargen` erzeugt Buchstaben, während `echo` sich damit begnügt, die empfangenen Daten wieder zurückzuschicken. Der Pirat braucht also nur UDP−Pakete zum Port 19 (`chargen`) eines seiner Opfer zu schicken und seine IP−Adresse und Quellport zu fälschen. In diesem Fall ist der Quellport der UDP−Port 7 (`echo`). Die UDP−Überschwemmung führt zur Auslastung der Bandbreite zwischen den beiden Maschinen. Ein ganzes Netzwerk kann also Opfer einer UDP−Überschwemmung

werden.

Packet Fragment

Denial−of−Services vom Typ „Packet Fragment“ nutzen die Schwäche einiger Implementationen des TCP/IP−Stacks auf der Ebene der IP−Defragmentation (Wiederzusammensetzung der IP−Fragmente) aus.

Eine bekannte darauf basierende Attacke ist Teardrop. Der Fragmentierungsabstand des zweiten Fragments ist kleiner als die Größe des ersten, ebenso der Abstand plus die Größe des zweiten, d.h. das zweite Fragment ist im ersten enthalten (overlapping). Bei der Defragmentation behandeln manche Systeme diese Ausnahmesituation nicht, und das führt zum Denial−of−Service. Es gibt Varianten des Angriffs: bonk, boink und newtear zum Beispiel. Der Denial−of−Service namens „The Ping of Death“ nutzt ein schlechtes Verhalten der Defragmentation auf der ICMP−Ebene aus, indem er eine Datenmenge schickt, die größer ist als die Maximalgröße eines IP−Pakets. Diese verschiedenen Denial−of−Service−Attacken enden in einem Absturz des Zielrechners.

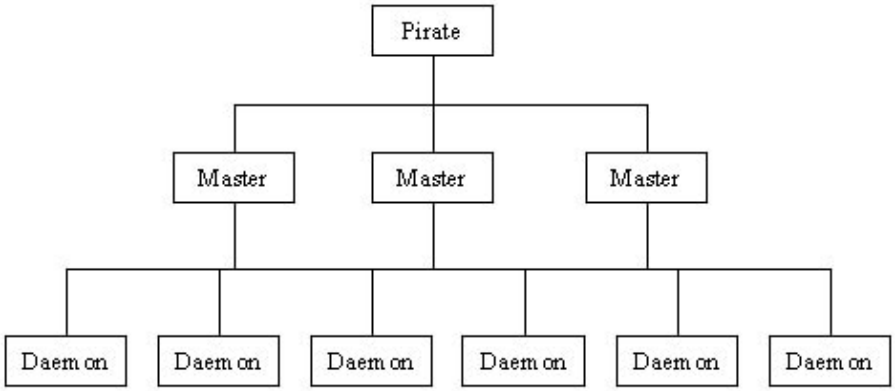
Smurfing

Dieser Angriff verwendet das Protokoll ICMP. Sobald ein Ping (eine ICMP ECHO−Botschaft) an eine Rundsendungs−Adresse (z.B. 10.255.255.255) geschickt wird, wird sie übersetzt und an jede Maschine des Netzwerks geschickt. Das Wesen der Attacke ist die Fälschung der gesendeten ICMP ECHO REQUESTs, indem als IP−Absenderadresse diejenige des Zielrechners benutzt wird. Der Pirat schickt einen kontinuierlichen Ping−Strom zur Rundsendungs−Adresse eines Netzes, und sämtliche Rechner antworten mit einem ICMP ECHO REPLY an den Zielrechner. Der Strom wird also von der Zahl der Rechner im Netz vervielfacht. Hier erleidet das ganze Netz die Denial−of−Service−Attacke, weil die enorme Quantität des von ihr erzeugten Datenverkehrs das Netz verstopft.

Verteilter Denial−of−Service

Verteilte Denial−of−Services überfüllen das Opfernetzwerk. Das Prinzip ist, verschiedene Quellen (Daemons) für die Attacken und höhere Instanzen (Master), die jene kontrollieren, zu benutzen. Die bekanntesten Werkzeuge des DDoS (*Distributed Denial of Service*) sind Tribal Flood Network (TFN), TFN2K, Trinoo und Stacheldraht. Die Abbildung 13 zeigt ein typisches DDoS−Netz:

Abb. 13: DDoS−Netz



Der Pirat benutzt „Dienstherren“, höhere Instanzen, um leichter die Quellen kontrollieren zu können. Denn er muß sich mit den Dienstherren (mit TCP) verbinden, um sie zu konfigurieren und die Attacke vorzubereiten. Die Dienstherren begnügen sich damit, Kommandos per UDP zu den Quellen zu senden. Hätte er die Dienstherren nicht, müßte sich der Pirat zu jeder Quelle verbinden. Der Ursprung der Attacke würde viel leichter entdeckt und ihre Umsetzung dauerte viel länger.

Jeder Daemon kommuniziert mit seinem Master, indem er spezifische Botschaften austauscht, je nach dem verwendeten Werkzeug. Diese Kommunikation kann sogar verschlüsselt oder authentifiziert erfolgen. Um die Daemons und Master aufzustellen, benutzt der Pirat bekannte Lücken (Buffer Overflow in den Diensten RPC, FTP oder anderen). Der Angriff selbst ist eine SYN−Überschwemmung, UDP−Überschwemmung oder sonstige Smurf−Attacke. Das Ergebnis eines Denial−of−Service ist es somit, das Netzwerk unzugänglich zu machen.

Fazit

Heute verstärkt sich die Sicherheit gegen entfernte Attacken mehr und mehr, im Gegensatz zur internen Sicherheit. Dieser „arme Verwandte“ der Schutzvorkehrungen gegen Piraten läßt immer noch lokalen Attacken wie dem TCP Session Hijacking, der ARP−Fälschung und der DNS−Fälschung gute Chancen. Zudem tauchen die Vorhersage von Sequenznummern (das Herzstück der IP−Fälschung) und die Varianten der Fragment Attacks einzig und allein wegen Fehlern auf Betriebssystemebene der Netzrechner auf. Was die Angriffe durch Applikationen betrifft, so haben jene noch gute Zukunftsaussichten dank der unaufhörlich wachsenden Komplexität von Web−Applikationen und dank der immer kürzer werdenden Fristen für Entwickler und Administratoren. Die Denial−of−Service−Attacken werden in ihrer verteilten Form ebenfalls sehr gefährlich bleiben, solange nicht alle auf den Schutz ihrer eigenen Rechner achten.

Links

- RFC 1858 − Security Considerations for IP Fragment Filtering : sunsite.dk/RFC/rfc/rfc1858.html
- IP Spoofing Demystified − Phrack 48 : www.phrack.org/
- Simple Active Attack Against TCP − Laurent Joncheray : www.insecure.org/stf/iphijack.txt
- DNS ID Hacking − ADM Crew : packetstorm.securify.com/groups/ADM/ADM−DNS−SPOOF/ADMID.txt
- The DoS Project's "trinoo" − David Dittrich : staff.washington.edu/dittrich/misc/trinoo.analysis
- The Strange Tale of the DENIAL OF SERVICE Attacks against 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

---

2005−01−11, generated by lfparser_pdf version 2.51