# Angriffsszenarien in Netzwerken

von Christian Daniel und Markus Müssig



# 1.Einführung

Im Jahre 2002 hat das Institut cert.org 5500 Vulnerabilities veröffentlicht. Das sind 5500 Stellen in Programmen, die ein Programm sicherheitsbezogen unsicher oder sogar gefährlich machen.
Angenommen, Sie arbeiten in einer Firma als Administrator und finden 10% der Veröffentlichungen passend auf ihre Client-/Serverlandschaft! Das bedeutet in verantwortungsvoller Konsequenz, dass sie 550 Papers zu diesen Programm-schwachstellen lesen und bewerten müssen und nach Bedarf ihre Software zu aktualisieren haben.
Es ist leicht nachvollziehbar, dass das den arbeitszeitlichen Rahmen eines einzelnen Administrators sprengt.
Hinzu kommen:

- schlecht geschulte Mitarbeiter
- unvorsichtige Mitarbeiter
- überforderte/schlecht ausgebildete Administratoren
- unsichere Technologien (von der Chefetage gewünscht)
- fehlende Geldmittel
- ...

All diese Punkte bilden einen Nährboden für theoretische Angriffsszenarien. Machen Sie sich klar, dass Sie ein Netzwerk niemals 100%ig sicher administrieren können. Tun Sie das, was in angemessenem Zeitaufwand und überschaubaren Kosten getan werden kann und die Kompromisse hinsichtlich der Verwendbarkeit eines Netzes/Rechners nicht überproportional benachteiligt.
Glücklicherweise ist gut die Hälfte der potenziellen Gefahren mit einfachen Mitteln auzuschließen.

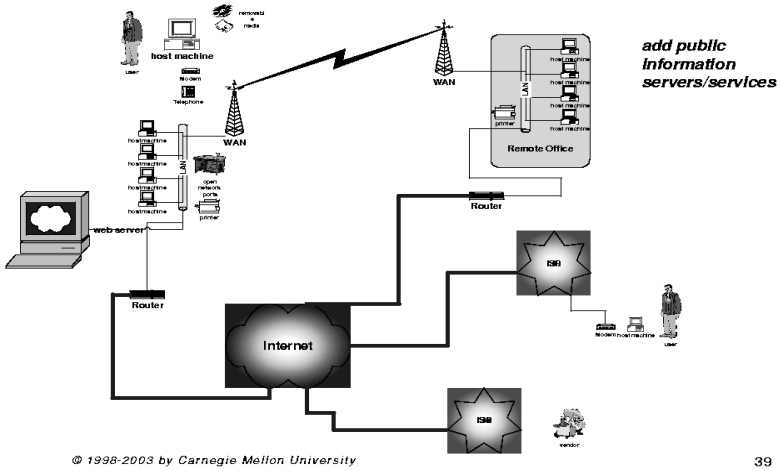
Dazu zählen:
- Virenscanner und aktuelle Virendefinitionen
- Sicherere Browserkonfiguration
- Klare Anweisungen und Richtlinen zur Netzbenutzung für alle Mitarbeiter
- Regelmäßige Sicherheitsupdates
- Keine Mails mit aktiven Inhalten
- Regelmäßige Backups
- Kein Zugang für Fremde an Schlüsselstellen des Netzwerkes (Serverraum, Passwörter...)
- ...

# 2.Die Komplexität der Systemadministration

In den letzten 25 Jahren hat sich rund um Computer und deren Vernetzung viel getan. Damals waren Netzwerke wenig komplex, die Anzahl der zu administrierenden Benutzer in einem Administrationsumfeld gering oder gar gleich EINS (single-user-workstation) und die Vermaschung mit anderen Netzwerken und Rechnern klein. Das hat sich grundlegend geändert.

Vom Einzelplatzrechner, an dem ein Benutzer arbeitete, welcher nicht vernetzt war kamen wir zu komplexen Netzen, mit:
- Mehrbenutzerbetrieb
- Internetzugang
- Remonte-Resourcen
- Öffentlichen Servern
- VPN für mehrere Standorte
- Offene "Verteilnetze" wie WLAN
- Wechseldatenträger


Von überschaubaren Einzelmaschinen zu unübersichtlichen, komplexen Netzwerk-strukturen:



Die sich daraus ergebenden Folgen sind:
- Höherer Administrationsaufwand
- Größere Lücken in der Administration (das Wichtigste erledigen)
- Höhere Anforderungen an:
  - Know-How des Administrators
  - Flexibilität des Administrators (bei Vorfällen zählen Arbeitszeiten nicht...)
  - Geldmittel
- Immens steigende Kosten
- ...

# 3.Der Ablauf eines Angriffes

1. **Scannen eines Zielrechners oder Netzwerkes (nmap)**
   Nmap ist ein weit verbreitetes Scannertool in der Unix/Linuxwelt. Es bietet eine reichhaltige Palette an Scanoptionen. Mitunter solche, die vom Opfer mit hoher Wahrscheinlichtkeit nicht einmal bemerkt werden.
   Der Aufruf nmap -sS -O -p 10-500 192.168.1.0/24 scannt zum Beispiel das komplette Subnetz 192.168.1.0/24 auf den TCP-Ports 10 bis 500. Für einen schnellen Überblick reicht dieser eingeschränkte Scan aus.
   Die Option -O versucht einen sog. OS-Fingerprint zu erstellen und das Betriebssystem zu erraten, das ein Host verwendet.
   Die Option -sS führt einen sog. Syn-Scan durch. Der Scanner schickt ein SYN-Paket an die Ziel-IP und wartet auf ein SYN-ACK. Erhält er dieses, sendet er ein RST-Paket. Der Three-Way-Handshake ist nicht abgeschlossen, die Verbindung wird im halboffenen Zustand zurückgesetzt und der Scan wird nicht geloggt, da nur abgeschlossene Verbindsaufbauten angezeigt werden.

2. **Feststellen der Dienste und ihrer Versionen hinter den offenen Ports**
   Eine Diensteversion lässt sich einfach feststellen indem man mit dem Programm **telnet** auf den entsprechenden Socket verbindet. Entweder verrät der Server gleich, welche Version installiert ist (z.B. ssh, Version...), oder es muss erst ein gewisser minimaler Protokoll-Handshake durchgeführt werden um an die Versionsinformation heran zu kommen (z.B. HTTP-Server).
   Alternativ kann das Tool amap (application-mapper) im Gegensatz zu nmap (network-mapper) die Dienste und ihre Versionen direkt in einem Schritt ermitteln.

3. **Suchen von Exploits**
   Eine gute Quelle ist immer www.securityfocus.com. Dort gibt es eine Rubrik Vulnerabilities, welche nach Herstellern und Versionsnummern bekannte Schwachstellen auflistet und unter Umständen sogar einen fertigen Exploit zur Verfügung stellt. Fertige Exploits werden in den meisten Fällen als kleine C-Programme veröffentlicht, deren Funktion allerdings häufig durch eine einfache Änderung eingeschränkt wurde um sog. Script-Kiddies fern zu halten.

4. **evtl. Exploit selber schreiben**
   Im Zweiten Teil des Vortrags wird das Erstellen eines Explots auf Basis der relativ gängigen Buffer-Overflow-Vulnerability genau erklärt.

5. **Ausführen von Exploits**
   Das Ausführen des Exploits beschränkt sich meistens auf das einfache Aufrufen eines Programms unter Angabe von IP-Adresse und Port des Opfers. Danach

versuchen Angreifer typischer Weise, ihre Spuren zu verwischen und leichter zugängliche Hintertüren auf einem System zu installieren.

# 4.Demonstrationen

1. **Netzwerkattacken**
   1. **Man-In-The-Middle**
      Die ARP-Tabellen der Teilnehmer an einem Ethernet-Netzwerk werden verfälscht, so dass Pakete nicht mehr direkt an den Zielhost zugestellt werden, sondern die IP-Adresse des Zielrechners mit der MAC-Adresse des Angreifers verknüpft wird. Der Angreifer erscheint dann z.B. im traceroute als zusätzlicher Hop.

   2. **Host-Isolation**
      An einen Opferrechner wird für jede mögliche IP-Adresse im LAN-Segment eine ungültige Zuordnung zu einer MAC-Adresse gesendet. Da der ARP-Cache beim Opfer unkontrolliert vom Betriebssystem aktualisiert wird, wird das Opfer nicht mehr selber nachfragen, sondern die falschen Angaben in seiner ARP-Tabelle ohne weiteres Hinterfragen verwenden. Da die MAC typischer Weise nicht existiert, ist der Rechner von jeder IP-Kommunikation ausgeschlossen.

   3. **TCP-Kill**
      Senden von gefälschten RST-Paketen an einen der Kommunkiationspartner. Der Empfänger löscht die Verbindung aus seiner Verbindungstabelle und sendet seinerseits RST-Pakete als Antwort auf die weiterhin eintreffenden Pakete seines Partners.
      Hierzu müssen gültige Sequenznummern in die RST-Pakete eingesetzt werden, die man nur durch das Mitlesen von TCP-Paketen erfahren kann.
      --> Man-In-The-Middle um gültige Seq.Nr. zu sniffen.

   4. **HTTPS-Attacke**
      Der Angreifer erstellt ein falsches Serverzertifikat, bringt sich in die Mitte zwischen Client (Opfer) und Server und lauscht auf "eingehende" HTTPS-Verbindungen.
      Bei Eingang einer Verbindung schickt er das gefälschte Serverzertifikat an den Client und übernimmt seinerseits die Kommunikation zum eigentlichen Server. Beim Akzeptieren desselben kann der Angreifer die Pakete des Opfers entschlüsseln, da sie mit dem public-key des gefälschten Zertifikates verschlüsselt wurden.
      Es handelt sich hierbei nicht direkt um einen Angriff auf eine Software-Schwachstelle, sondern es wird die Tatsache ausgenutzt, dass User dazu neigen, "störende" Pop-Up-Fenster beim Surfen im Internet einfach mit "Ok" zu bestätigen.

2. **lokale Attacken**
   1. **strace (keylogger)**
      Abfangen von Funktionsaufrufen an der Übergangsschicht zwischen Anwendung und Betriebssystem-Kernel (bei strace Linux). Hier ist es möglich, die gedrückten Tasten in dem Moment abzufangen, in dem sie vom Kernel (Tastaturdevice aus dem Kernelinneren) zurück zur Anwendung geliefert werden. An dieser Stelle ist es dann z.B. möglich, das Passwort oder den Key für eine SSH-Verbindung abzufangen oder die Passphrase für einen

RSA-Schlüssel.

2. **JohnTheRipper**
   Brute-Force oder Wörterbuchattacken auf Passwortdateien, die normalerweise Passworte nur als Hash-Werte enthalten, d.h. es ist nicht möglich, aus den Einträgen in auf das Passwort zu schließen.

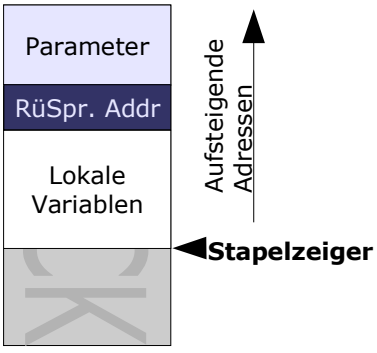
# 5.Buffer-Overflow

1. **Das Wesen von Buffer-Overflow Attacken**
   Buffer-Overflow Attacken basieren, dass Programme beim Einlesen von Daten nicht darauf achten, den Füllgrad des Empfangspuffers zu prüfen. Durch geschicktes Füllen/Überflüllen von Eingabepuffern ist es zu schaffen, eigenen Code in die verletzbare Anwendung einzuschleusen und diesen zur Ausführung bringen. Dieser Code läuft dann mit den Rechten der Anwendung - bei Netzwerkdiensten also typischer Weise mit root- bzw. Administrator-Rechten.

2. **Die Opfer von Buffer-Overflow Attacken**
   Typische und lohnende Opfer sind neben Netzwerkdiensten die sogenannten suid-root Programme. Veranlasst man sie dazu, durch einen eingeschleusten Angriffscode eine Shell bereitzustellen, läuft diese natürlich mit Root-Rechten.

3. **Warum Buffer-Overflow Attacken möglich sind**
   Der Speicher, den eine Anwendung vom Betriebssystem zur Verfügung gestellt bekommt, ist in drei Teile aufgeteilt:
   1. Das Codesegment
      Hier liegt der zur Anwendung gehörige Maschinen-Code. Auf dem Speicherbereich des Codesegments dürfen Befehle ausgeführt werden.
   2. Das Datensegment
      Hier liegen die zur Anwendung gehörigen Daten. In diesem Bereich sollte eine Ausführung von Code nicht erlaubt sein.
   3. Das Stacksegment
      Ein Teil des Datensegments. Erlaubt das Betriebssystem das Ausführen von Code in diesem Bereich des Anwendungsspeichers, sind Buffer-Overflow Attacken möglich, da in den Stackbereich der Angriffscode eingeschmuggelt wird.

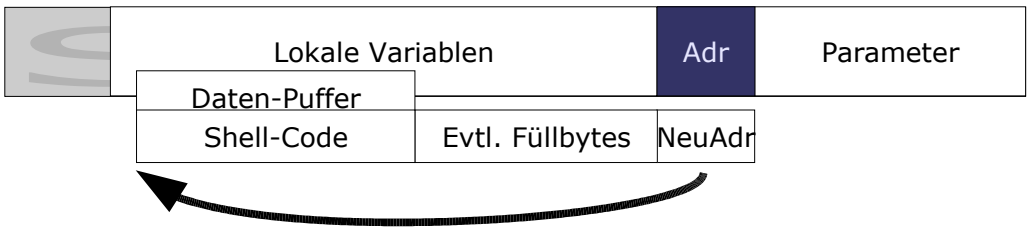
4. **Der Stack**
   Der Stack ist ein LIFO-Puffer und dient als Zwischenspeicher für lokale Variablen und enthält die Rücksprungadresse bei Funktionsaufrufen. Diese Adresse wird beim Überfüllen eines Puffers, der Teil der lokalen Variablen einer Funktion ist, überschrieben.



5. **Wie funktionieren Funktionsaufrufe?**
   Findet ein Funktionsaufruf statt, werden zuerst die Parameter der Funktion auf dem Stack abgelegt. Danach speichert die CPU die Rücksprungadresse auf dem Stack, wobei es sich hierbei um die Adresse des nächsten auszuführenden Befehls nach Abarbeitung des Unterprogramms handelt. Schließlich wird Platz für die lokalen Variablen geschaffen, wobei der Compiler beim Übersetzen des Programms berechnet hat, wie viele Bytes hierfür reserviert werden müssen.
   Eingaben, z.B. die von einem Webserver gewünschte URL, werden in einem Puffer zwischen den lokalen Variablen platziert.
   Es ist also möglich, Eingaben an ein Programm direkt auf dem Stack der Maschine zu platzieren. Entsprechend ist es auch möglich, bei Verletzung der Puffergrenze, den Stack zu überschreiben, so lange man dabei bestimmte Regeln nicht verletzt (keine ungültigen Zeichen in der Eingabe). Diese Regeln sind davon abhängig, welches Programm und welches Protokoll die Eingabe verarbeitet. Im Falle eines Webservers dürfen nur Zeichen vorkommen, die auch in URLs erlaubt sind.

6. **Wie funktioniert ein Buffer-Overflow**
   Wenn man die Daten im Eingabestrom nun sorgsam wählt, kann man den für lokale Variablen vorgesehenen Speicher mit Anweisungen überschreiben. Diese Anweisungen stellen normalerweise den Shellcode dar, ein kleines Assembler-Programm, das die Aufgabe hat, auf dem angegriffenen Rechner eine Hintertüre zu öffnen. Dazu wird häufig einfach eine Shell (z.B. /bin/sh) gestartet. Ist der Shellcode (also die präparierte Eingabe) nicht lang genug für den Speicherbereich der lokalen Variablen, kann man ihn mit Zeichen auffüllen, die keinerlei Auswirkung auf haben (z.B. NOP-Zeichen 0x90).
   Ist der Bereich für lokale Variablen aufgefüllt, beginnt auf dem Stack der Speicherbereich, in dem die Rücksprungadresse für das Hauptprogramm liegt. Diese muss nun so manipuliert werden, dass das Unterprogramm nicht zurück ins Hauptprogramm springt, sondern an den Anfang des übermittelten Shellcodes. Die Daten, die also an den Platz der Rücksprungadresse geschrieben werden, werden ebenfalls über die Eingabe des Programms (im Beispiel der Webserver) an das verletzbare Programm gesandt. Die Berechnung des Wertes der Adresse, an die gesprungen werden muss, ist auf den Folien exemplarisch für ein selbstgeschriebenes, für Buffer-Overflow anfälliges Programm dargestellt und stellt den wohl schwierigsten Teil des Angriffs dar, da der Angreifer genaue Kenntnis über Aufbau des angegriffenen Dienstes haben muss.

   Darstellung des Stacks nach Beschreiben mit manipulierten Eingaben:



# 6.Begriffe

**Social Engineering**
Angriffsart, bei der der Angreifer psychologische oder soziale Schwächen, Leichtgläubigkeit oder Unwissenheit der Opfer ausnutzt und unter Vorgabe einer falschen Identität Informationen erhält.

**Spoofing**
[engl. veralbern, hereinlegen]
Der Angreifer täuscht eine
- falsche MAC
- falsche IP-Adresse
- falsche DNS-Infos

vor und erhält so Kommunikationsmöglichkeiten mit Partnern, die den Angreifer für ein vertrauenswürdiges Gegenüber halten.

**Connection Hijacking**
Das Entführen einer Verbindung. Vorbereitende Man-In-The-Middle Attacke führt dazu, dass der Angreifer durch Verwendung korrekter TCP-Sequenznummern eine laufende Kommunikation übernimmt, und den alten Kommunikationspartner via DoS-Attacke ausschaltet.

**SUID-Root**
Programme, die root gehören und das SUID-Bit gesetzt haben. Werden solche Programme mittels Buffer-Overflow dazu gebracht, eine Shell zu starten, ist dies eine Rootshell.
Finden von SUID-root-Programmen im Filesystem als Maßnahme zum "härten" der Betriebssystemsicherheit:
find / -type f -perm +4000 -user root (finde alle Files, die das SUID-bit gesetzt haben und root gehören)

**Buffer-Overflow**
Das "Überfüllen" von Eingabepuffern mit präparierten Daten (diese enthalten im Normalfall Code zum Starten einer Shell). Nach einem Methodenaufruf, der den Eingabepuffer eines Programms füllt, kehrt das Programm nicht zur urspünglichen Aufrufstelle zurück um normal fortzufahren. Durch das "Überfüllen" des Eingabepuffers wurde die Rücksprungadresse auf dem Programmstack überschrieben und zeigt jetzt an eine Stelle des selbst eingefügten Codes. Im Falle von SUID-root-Programmen ist das Ergebnis eine root-Shell. C ermöglicht diese Art der Attacke, weil bestimmte Kopierfunktionen keine Bereichs- oder Längenprüfung für den Zielpuffer vornehmen.

**Privilege Escalation**
Angriffsart, bei der es um das Erlangen von höheren Rechten im System geht. Bsp. Buffer-Overflow auf SUID-root Programme. Dieser Begriff wird bei der Klassifizierung von Angriffsarten verwendet.

**DoS - Denial of Service**
Angriffsart, bei der der Angreifer einen Computer mit Anfragen überhäuft, so dass dieser alle Rechenleistung für deren Beantwortung aufbraucht, und so für seine normalen Dienstangebote nicht mehr zur Verfügung steht. Durch gefälschte Absenderadressen seitens des Angreifers verstärkt sich die Effektivität eines solchen Angriffes, weil das Opfer vergeblich auf Quittungen für die gefälschten Verbindungsanfragen wartet.
- SYN-Flooding zum Überlaufen der Connection-Table
- UDP-Flooding zum Aufbrauchen der Bandbreite
- ...

**dDoS**
Ziel wie bei DoS-Attacke, jedoch unter Zuhilfenahme sog. Slaves. Bei den Slaves handelt es sich um Rechner, die der Angreifer unter seine Kontrolle gebracht hat. Ein z.B. über einen Virus eingeschleustes Skript attackiert zusammen mit den anderen Slaves den Opferrechner.
Häufig sind mehrere Tausend Slaves beteiligt. Der W32.Blaster Wurm sollte windowsupate.microsoft.com 'DoSen'.
Der Code war aber nicht sehr clever geschrieben:
Die Webseite wurde im Code des Wurms durch einen DNS-Namen identifiziert. Als Abwehr reichte es, den Namen einfach aus dem DNS zu löschen, wodurch keine IP-Adresse als Angriffsziel mehr ermittelt werden konnte.

**Man-In-The-Middle**
Der Angreifer bringt sich z.B. durch Verfälschen der ARP-Tabellen der beiden anderen Kommunikationspartner in die Mitte der Kommunikation und leitet die Pakete von beiden Seiten unbemerkt nach dem Lesen oder Verändern weiter.

**Eavesdropper**
[engl. Horcher, Lauscher] Der Eavesdropper ist der Mithörer einer Verbindung. Pakete, die eigentlich verschlüsselt sein oder gar überhaupt nicht beim Angreifer vorbeilaufen sollten, können gelesen und evtl. auch verändert werden.

**Replay-Angriff**
Angriffsart nach einer Man-In-The-Middle-Attacke, bei der Datenpakete mitgelesen und verändert wieder in den Datenstrom eingefügt werden

**Exploit**
Kleines Programm zum Ausnutzen eines ganz bestimmten Implementierungs-fehlers in einer Software.

**Vulnerability**
Die Problembeschreibung eines fehlerhaft implementierten Programmes.

**Socket**
Die exakte Definition eines Kommunikationspartners. <IP-Adresse>:<port> definiert einen Socket. Eine TCP/IP-Kommunikation definiert üblicherweise zwei Sockets:
Source: 192.168.1.1:26736
Destination: 192.168.1.10:80

**BruteForce-Attacke**
Attacke, bei der eine Eingabeaufforderung mit nach bestimmten Algorithmen erstellten Testeingaben automatisiert gefüttert wird. Die automatisiert

eingegebenen Werte können auch aus Wörterbüchern stammen. --> Wörterbuch- oder Dictionary-Attacke.
PRINZIP: Erraten von korrekten Eingaben durch Ausprobieren aller denkbaren Möglichkeiten

**Shadow Passwörter**
Mechanismus in Unix/Linux Betriebssystemen, bei der die Passwörter aus der Passwort-Datei (z.B. /etc/passwd) ausgelagert sind und in der Datei /etc/shadow verschlüsselt abgelegt sind. In der Datei /etc/passwd steht dann an Stelle der Passwörter ein x.
Da für die Datei /etc/shadow strengere Rechtevergabe möglich ist, ist darin ein sinnvoller Schutz vor BruteForce-Attacken zu sehen.

# 7.Quellen und Links

**INTERNET**

www.insecure.org
www.cert.org
www.securityfocus.com
www.linuxsecurity.com
www.heise.de
www.sans.org
www.freshmeat.net
www.honeynet.org
www.packetstormsecurity.org

**LITERATUR**

Hacking Linux Exposed
Hacking Exposed (Network Security – Secrets and Solutions)
Linux-Hackers-GuideLinux-Security
Smashin' the stack for fun and profit

Fragen:
- Mit welchen Arten von Angriffen muss ein Administrator rechnen?
- Mit welchen unerlaubten Vorgängen muss man im Besonderen in einem Ethernet-Netzwerk rechnen?
- Welche besonderen Schwachstellen ergeben sich aus der Architektur der bekannten Rechnersysteme und wie lassen sie sich ausnutzen?