WIDZ - The Wireless Intrusion detection system

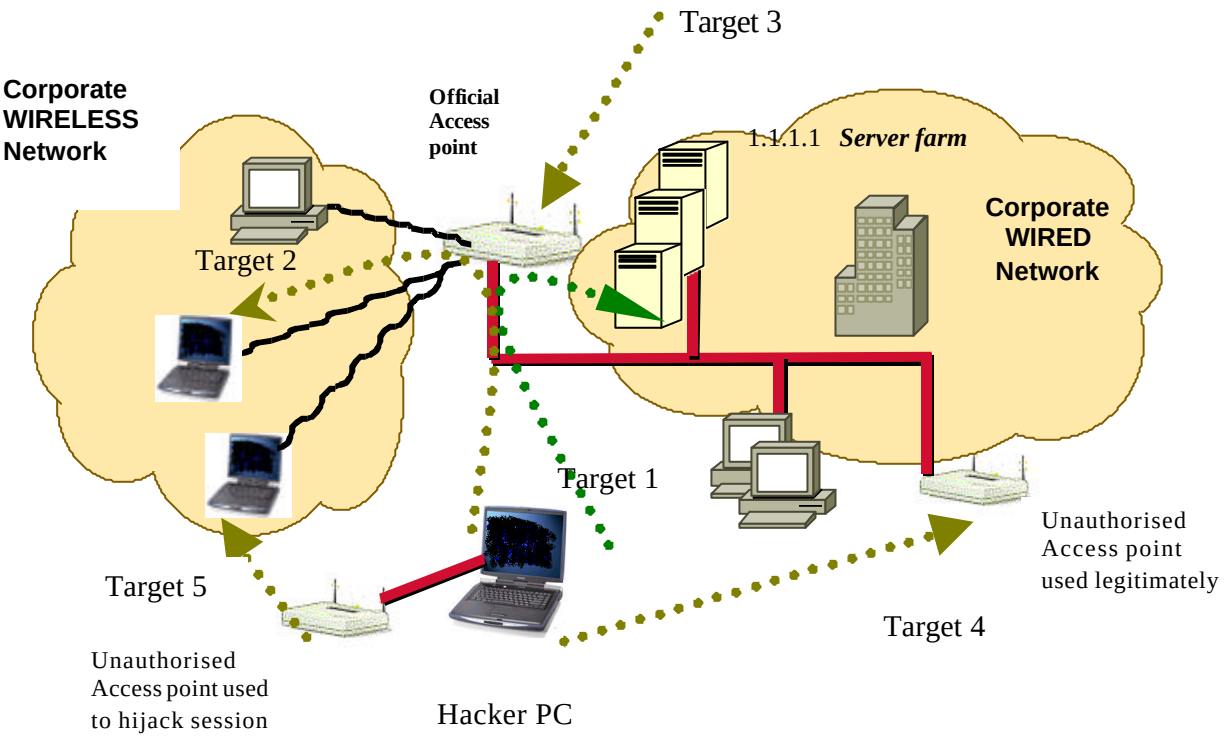
The design and implementation of Version One

1 Introduction (and excuses)

WIDZ version 1 is a proof of concept - It is not up to the standards of (and was never intended to be) great software packages like Snort etc that you might find on FreshMeat and sourceforge. Even given this, the code was very late – as its development has been bogged down by a series of trials some technical, some managerial – For this I apologise but its understandably hard to make time for "home time" development projects when the department that it took you ten years to build up has become a pawn in your boss' power games (well boo hoo, pull yourself together man)

None-the-less WIDZ had some very fresh ideas when it was conceived. We started by intentorising the network – then we studied our own wireless pentests and forensic reviews to establish what is attacked, and how.

1.1 The wireless network –what is attacked



www.loud-fat-bloke.co.uk

The diagram above shows the various elements of the wireless network and how these can be attacked:

- The corporate network and servers

This is what most people generally recognised as a 802.11 hacking target. An attempted penetration through the official access points(target 1) into the corporate network. Very simple and very basic.

Here only 802.11 scans and DOS attacks should remain un-detected by the existing infrastructure. If any more gets by, there is a more general network security problem.

- The wireless clients

The wireless clients(Target 2) is generally not recognised as a target. Here the Access point behaves as a hub connecting the authorised wireless clients to those of the bad buys – inevitably this will expose a connecting pc to a huge array of IP based attack.

- The unauthorised Access point

Unofficial access points installed by user departments (target 4) represent a huge risk as the security configuration is often questionable, providing an effective yet unmonitored back-door to the network.

Bogus Access points (Target 5) represent a different threat as these can be used to hijack sessions at the data link layer and steal valuable information.

- Target 3 – The legitimate Access point

Services like SNMP and web-based configuration tools on the Access point are often targeted by attackers. Again protecting the internal interface is a typical network security challenge – the same as any router or switch.

I decided that Target 3 could be adequately provided for by normal wired security mechanisms. I also decide that target 1 and target 2 functions could be combined into one module so we came up with the WIDZ two module design.

2 WIDZ two module design.

2.1 - Unauthorised AP monitor (widz_apmon.c)

This module covers two threats:-

- Bogus APS are designed to steal the association . Once this is achieved login credentials can be retrieved or a man in the middle attacks can be performed

- Unauthorised AP are the ones that are installed by, say, the marketing department after they have visited the local PC superstore. They usually allow all and sundry access to the corporate lan without a password.

Addressing this solution was easy - any reasonable administrator in a reasonably size organisation could identify an official Access point and record their details in a file. These details would include Mac Address and ESSID. All this module had to do was to do an AP scan and compare the results with our file – any that where found and did not appear are either bogus or unauthorised. Consideration was given to detecting changes in signal strength and frequency but this was considered over-kill.

2.2 802.11b Traffic monitor(widz_probemon.c)

A file containing Mac Addresses and ESSIDs is obviously not going to provide a real world solution to drive this modules detection capabilities. This is because:

- Any self-respecting hacker or war-driver knows how to change a MAC-Address – making the identification process flimsy at best.
- Many organisation would have thousands of network cards some of which will need to be replaced as they become faulty – this would create an administrative nightmare.

But frankly this left the first design *wanting* functionally – I decided to develop the code and see whether it (and the concurrent Wireless honey pot project) would shed any light on the required processing – it has see ***implementing 802.11 IDS attack signatures***. The module has two functions:

- **probe monitoring** - Picks up probe requests which don't have the ESSId field set in the probe.
- **Flood detection** - Picks up attempts to flood the AP with associations.
- **Wireless Client attack** – (picking up of wireless attacks on associated wireless client) although considered was never implemented

2.3 Alert management

Each new security product seems to need its own management console – that cannot be right! For both these modules it was planned to report in a way that could be processed by Snort.

3 Implementation

We used a DWL650 with Wlan-NG drivers. We also used a 2 box approach with 2 Compaq M700 laptops - Apmon running on one box and probemon running on another.

3.1 widz_apmon.c

This little program monitors an area for Access Points. If finds an ap it compares it to a list of Authorised APs in a configuration file called *widz_apmon.conf*. If the detected AP isn't listed in the configuration file, the program raises an Alert with an appropriate message.

`Command syntax`

`widz_apmon` ***`sleep_time Interface generate|monitor`***

Sleep_time is the time between scans in seconds

Interface is the WLAN network interface usually wlan0 or eth1

generate|monitor are mutually exclusive command

generate produces an automatic copy of *the widz_apmon.conf* file in the current directory
monitor – runs the program IDS mode

`example`

```
$ widz_apmon 1 wlan0 generate


$ widz_apmon 1 wlan0 monitor
```

`function`

widz_apmon.c assumes your driver is in auto associate mode. The network interface should be *up* otherwise the program returns odd results. It does the appropriate IOCTL call to read the ASSOCIATED ESSID from the interface then get the MAC.

It then performs the processing already described above to establish the validity of the detected AP, then it resets ESSID to force the card to search for another AP. When the program has processed all the Aps available it waits a couple seconds and restarts the process.

3.2 widz_probemon.c

This program reads the output of a program like prismdump, tethereal or our custom version.

It then takes two actions

alert1 - Alert if the essid is empty. It the calls the Alert script and the logs the next 100 packets from that source

alert2 - alert if more then MAXASSO occur in less than Maxasso time. It the calls the Alert script .

Usage

```
prismdump wlan0 |  widz_probemon > logfile
```

3.3 Alert

A program named Alert will be executed each time an Alert is raised. The WiDZ package provides an example script which shows how to

- send a syslog message
- write to the console or current terminal
- send an snmp trap
- send an email

4 WIDZ IMPROVEMENTS

A whole article is in progress describing the design of wireless IDS and WIZ V2 see ***implementing 802.11 IDS attack signatures***, but here are some initial ideas.

4.1 widz_apmon.c improvements

- AP scanning technique. It would be better to use a the sxxAPLIST ioct call or the scan function.
- Duplicate Alerts - It should also not raise an alert for the same AP more than once.
- Signal strength changes - Significant changes in signal strength and frequency should raise an alert.

4.2 widz_probemon.c improvements

- BPF processing – the module should be able to obtain RX frame standalone using pcap or wirtap.

- Bad Macs – Obviously detecting malevolence by a static Mac file would be a retrograde step but our work with the wireless honeypots show that many hackers have the strings EA7 or BAD in their Mac addresses – these good clues should not be over looked. The same give away detection could be applied to ESSID and Nickname.
- OS finger printing – Most corporate don't use Linux, yet most hackers do. I am considering implementing OS detection routines written by hobbie into the product.
- Good Authenticate Traffic – traffic that has been authenticated at OSI layers should be considered as good. We should have an exit point to establish this using finger, whoami and other existing services.
- Wireless Client attacks Wireless client – we never did implement this.

4.3 Alert improvements

- Snort processing - Snort processing was poor because we alert without an IP address. We plan to trap layer 3 traffic to obtain this and then notify snort with spoofed (source address) UDP packet. This will then trigger an *activate* or *tag* rule in snort.
- Shunning – We can use a disassociate packet to shun the malevolent probe. The following code segment could be used:

```
frame.hdr.mh_type = FC_TYPE_MGT;
frame.hdr.mh_subtype = MGT_DEAUTH;
send(socket, &frame, sizeof(frame), 0);
```