DA-WIN

Distributed Audit & Wireless Intrusion Notification

WIDZ- FG Configuration Guide

Contents

Distributed Audit & Wireless Intrusion Notification

1 Introduction

1.1 DA_WIN

DA WIN – Distributed Audit & Wireless Intrusion Notification is an evolution of a previous open source software Project. It includes a re-write of the WIDZ software, new drivers and a new miniature low cost ARM platform.

The project has been rejuvenated because Wireless has a re-emergence of importance because:

- Wireless IDS or Wireless scanning being mandated by PCI-DSS
- having a full section in GPG-13.
- Wireless Security having a bearing on BYOD

WIDZ 1.5, a proof of concept, was the first release of WIDZ and arguably one of the first purpose designed Wireless IDS. WIDZ 1.8 was the last release of open source release of WIDZ; it was still a proof of concept but had a degree of technical complexity. It was written for speed using PF_PACKET, modified PRISM2 drivers and specific PRISM2 IOCTLS to acquire the raw packets

1.2 WIDZ FG - Future generation

WIDZ FG - Future generation is not open Source and has been re-written. Notable features include:

- This new version was developed using the PCAP library. Previous versions of WIDZ received some criticism on Martin Roesches SNORT forums because it did not use Libpcap.
- Full channel AP table acquisition from the drivers
- A new miniature ARM platform and up-to-date drivers
- IWLIB and AIRMON driver compatibility
- Alerting via syslog to facilitate SIEM integration in line with PCI-DSS and GPG13 as referenced above.
- Daemonised components with full rc script integration and service scripts
- Improved parameterisation and configuration
- Designed to be compliant with much of NSA's "Guidelines for the Development and Evaluation of IEEE 802.11 Intrusion Detection Systems (IDS) (ref: I332-005R-2005)".

2 New Platform

The simple platform is based on a ARM processor. It has a built in 10/100 Mb/s nic and 2 USB2 ports. This doesn't make it unique these days – but what is interesting is the diminutive price tag. Even with the add-on costs, this puts multi-sensor deployments in the reach of even the most cash strapped cash organisations.

Mounting in an IP44 case it becomes suitable for unobtrusive deployment in an office or commercial environment for less than 50GBP per unit.

With multi-sensor deployment, we gain increased coverage and geo-location capability as required by the NSA standard above.

2.1 Distrubition

The PI runs on Debian based Linux. This is stored and boots on 8GB SD card. This has the advantage of having no moving plans and easy deployment/fabrication.

The build can be upgraded with a simple

$ dd bs=4m if=**dawinv2.2prod01012014.img** of=**/dev/sd[disk**#]

Where

dawinv2.2prod01012014.img is dd image you have downloaded from www.loud-fat-bloke....

dev/sd[disk#] is the disk name of your sd card on linux

when booting for first time root password is dawin

3 Multi-sensor deployments

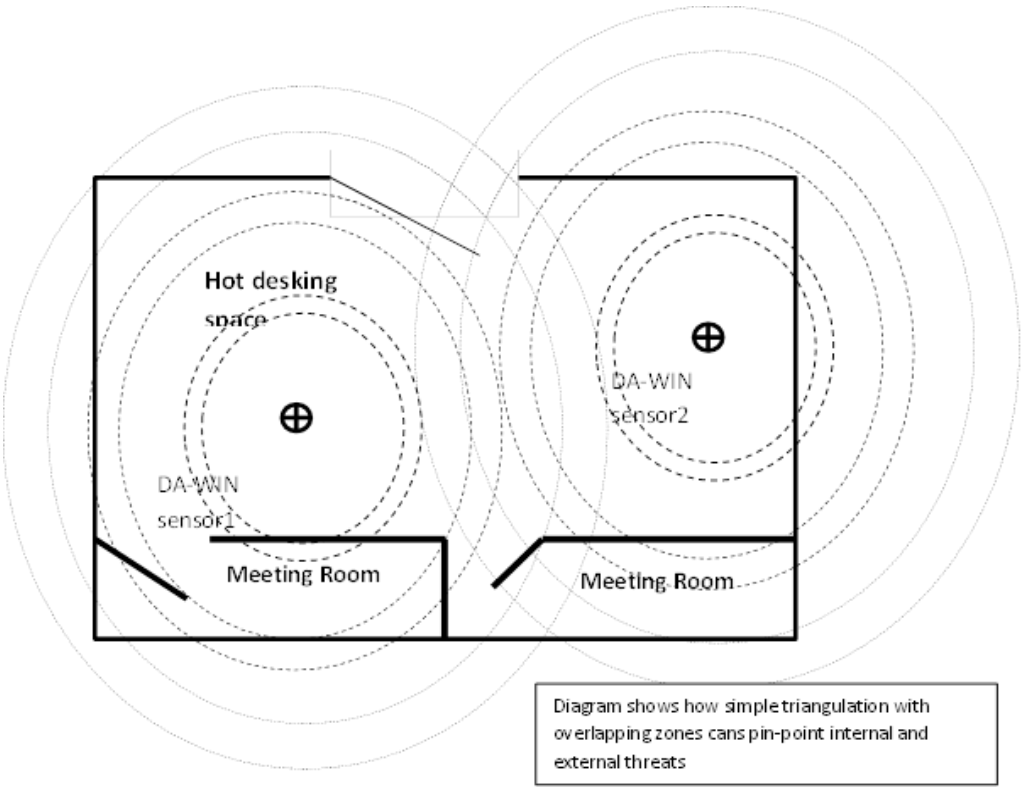

Diagram shows how simple triangulation with overlapping zones cans pin-point internal and external threats

4 Two module design

WIDZ was designed with two modules, the AP monitor and the Probe (or attack monitor). This are deployed separately to match the particular environment.

4.1 Unauthorised AP monitor (widz_apmon)

This module covers two types of attack:-

o Bogus APS which are designed to steal the associations.

o Unauthorised APs are the ones that are installed by, say, the marketing department after they have visited the local PC superstore.

4.2 Attack Traffic monitor(widz_probemon)

This modules detection capabilities includes

- Auth fail floods
- Asso floods
- Deauth floods
- General volume floods
- Fatajack
- General failures – most AP's do not provide any error recording, this detects non zero return code on authentication, association and disassociation and reports on them

Configuration of each of these modules is described in the following chapters.

5 widz_apmon

This monitors an area for Access Points(ap). If finds an ap it compares it to a list of Authorised APs in the config file.

If the AP isnt in the list it produces an Alert with an appropriate message. This will send a syslog message but it could sent an snmp trap.

5.1 Syntax

The command has the following syntax:

$ widz_apmon sleep_time Interface [generate|monitor] [-D]

- Sleep_time is the time between scans in seconds
- Interface is the WLAN interface say wlan0
- Generate produces the widz_apmon.conf file in the current directory
- Monitor - puts it into IDS mode
- -D -- runs in daemon mode; not for generate

5.2 Example

$ widz_apmon 1 wlan0 generate

Produces the widz_apmon.conf file in the current directory. This is intended to produce a baseline of the area for an easy initial install.

$ widz_apmon 1 wlan0 monitor -D

Run as an IDS daemon

Debugging mode can be enabled by exporting the DEBUG variable. This will cause the program to print debug info to stdout. It should not be used in daemon mode

$ export DEBUG=Y

$ widz_apmon 1 wlan0 monitor

5.3 The widz_apmon.conf file

5.3.1 Example

```
essid=virgin ap_mac=00:22:3f:ce:9f:36 channel=0 signal=174
```

```
essid=wireless-lab ap_mac=00:30:bd:f6:09:da channel=0 signal=196
```

5.3.2 Format

Each AP specification is contained on one line:

```
essid=BSID-Name ap_mac=AP_MAC_Address ↵
channel=Channel-no signal=signal-strength
```

5.3.3 essid=BSID-Name

The primary key is the access-point name.

5.3.4 ap_mac=AP_MAC_Address

This is the normal hardware address of the AP in hex colon format eg 00:22:3f:ce:9f:36.

If an essid name appears with a different MAC it will assume that there is rogue AP and alert

If ap_mac=NOCHECK - the NOCHECK directive prevents mac checking of that essid

5.3.5 channel=Channel-no

This is the normal RF channel of the AP.

If an essid name appears with the same MAC but a different channel it will assume that there is rogue AP and alert

if it is set to 666, all checking on channel is disabled

5.3.6 signal=signal-strength

This is the normal RF of the AP. By setting it to 999 this will effectively disable this check.

If an essid name appears with the same MAC but with a signal strength of 1.2 recorded strength it will assume that there is rogue AP and alert

5.4 Alerts

An example alert is produced syslog. The daemon adds date and IP address:

Jan 28 09:52:29 192.168.0.199 widz-sensor1 widz_apmon_wlan2: Alert unknown AP essid=wireless-lab ap_mac=00:30:bd:f6:09:db stored mac = 00:30:bd:f6:09:da

5.4.1 Alert 1

If widz detects an essid name that isn't in the baseline config file:

This alert will be produced.

```
widz_apmon_wlan2: Alert unknown AP essid=NETGEAR ↵
```

ap_mac=e0:46:9a:82:63:29 stored ap = unknown ESSid no stored mac

5.4.2 Alert 2

If the detected `essid` name appears in the baseline but the recorded mac in the baseline is not = Detected current mac - This alert will be produced:

widz_apmon_wlan2: Alert unknown AP essid=wireless-lab

ap_mac=00:30:bd:f6:09:db stored mac = 00:30:bd:f6:09:da ↵

5.4.3 Alert 3

If detected AP Name and MAC match those in the baseline file but actual signal > 120 percent of Recorded signal - This alert will be produced:

widz_apmon_wlan2: Alert AP essid=virgin ap_mac=00:22:3f:ce:9f:36 ↵

`recorded signal=` 10

5.4.4 Alert 4

If detected AP Name and MAC match those in the baseline file but Recored channel != actual detected channel - This alert will be produced:

widz_apmon_wlan2: Alert AP essid=virgin ap_

mac=00:22:3f:ce:9f:36 now on channel = 10 ↵

5.4.5 Alert 5

A baselined AP was not been detected for 4 * Sleep time - This alert will be produced:

widz_apmon_wlan2: Alert AP No beacon detected > 4 periods essid=virgin ap_

mac=00:22:3f:ce:9f:36 Is it down ↵

6 Configuring widz_probemon

Like ap_mon, widz_probemon. has a simple text control file *called* `probemon.conf`.

An example is shown below:

```
$cat probemon.conf

#this has the following stanzas
usebadmacs=y|n
#   use a blacklist  file
usebadssids=y|n
#    use a blacklist  file called badssids
usescripts=y|n
#comments start with # in col 0 just like the old days
#
# thresholds
#
#  type a = atomic
# s=summary i.e based on counts calculated at the end of assion
#       n = disabled
# lo:  = lo threshold -  alert after so many hits
# hi:  = hi threshold -  alert until so many hits
#  all counts resets at channel channel or after a session
#
alert:5 ,type:a ,lo:5 ,hi:10 ,test=extra alert description
#  scan channels
channel= 1, 2 , 3  ,10
```

It contains the following Stanzas each that take the value y or n:

- usebadmacs If set to n, this feature is NOT activated.

  If set to y, the probe monitor will report and call the alert subroutine for any packet that contains a mac address defined as a “bad mac” in any of the (up to) four address fields in an 802.11 packet.

  Bad macs are defined in a text file named in the constant (i.e. #define) BADMACFILE. Typically, it has the value "./badmacs"

- usebadssids If set to n, this feature is NOT activated.

If set to y, the probe monitor will report and call the alert subroutine for any probe or probe-response packet that contains a SID defined as a “bad sid”.

Bad SIDS are defined in a text file named in the constant (i.e. #define) BADSSIDFILE. Typically, it has the value "./badsids".

- usegoodmacs If set to n, this feature is NOT activated.

  If set to y, the probe monitor will IGNORE any packet that contains a mac address defined as a “good mac” in any of the (up to) four address fields in an 802.11 packet.

  Bad macs are defined in a text file named in the constant (i.e. #define) GOODMACFILE. Typically, it has the value "./goodmacs".

- usescripts If set to n, this feature is NOT activated.
  This feature has been deprecated.

  If set to y, the probe monitor will call a number of custom script for every record intercepted. The scripts must be stored in a directory defined by the expression stored in the constant WIDZSCRIPTS. An example is provide in the WIDZ 1.6 distribution.

  calls all scripts in directory ./scripts after setting the following environment variable

```
$WIDZSSID
$WIDZPACKETTYPE
$WIDZMAC1
$WIDZMAC2
$WIDZMAC3
```

6.1 Thresholds

The volume of alerts can be set with various thresholds. The config line is in the following format

alert: *alert-no* ,type:[a|s|n] ,lo:*low-threshold*,hi: *Hi-threshold*,test=extra alert description

alert: ***alert-no***

- alert: alert-no is the alert number

type :[**a**|**s**|**n**]

- a = atomic – will alert when a packet match occurs
- s=summary when a packet match occurs a counter is incremented. At the end of a timed session an alert is raised if the count falls between the hi and lo
- n = disabled

lo: ***lo-threshold***

- *lo threshold* - alert is ignored until the alert-count is above *lo threshold.* Then syslog the alert

hi: ***Hi-threshold***

- hi threshold - alert until alert-count is above this

All counts reset at channel change or after a session. A session is defined as 9999 packets or 200 seconds which ever least.

6.2 Channels

channel= 1, 2 , 3 ,4

- Allows a sequence of channel hoping of up to 4 channels. Each session starts on a different channel

6.3 Running Probemon widz_probemon.

To run widz_probemon from the command line requires the following:

$**widz_probemon wlan0**

Debugging mode can be enabled by exporting the DEBUG variable. This will cause the program to print debug info to stderr. It can not be used in daemon mode

$ export DEBUG=1

$**widz_probemon wlan0**

To test widz_probemon on a pre-stored pcap trace replace the interface parameter with a file ending in ” .pcap “ .It cannot be used in daemon mode. For example:

$ export DEBUG=1

$**widz_probemon /tmp/wirless_trace.pcap**

To run as a daemon

$**widz_probemon wlan0 -D**

6.4 alert1

alert1 will be triggered and go into alert function if the essid is empty (len < 2 as widz_hdr.essid always has min len of 1).

Previous versions would then log to stdout the next 100 packets from that source – this has been disabled. However, we could do a a pcapdump call for 100 packets

Packet fields	
Field	Alert Condition
Packet type	0
Packet_subtype	4
ESSID	len < 2
macaddress_1 (hex format)	N/A
macaddress_2 (hex format)	N/A
macaddress_3 (hex format)	N/A
macaddress_4 (hex optional depending on packet type)	N/A
Resulting action	
Alert =	Alert Null Probe ", Followed by SSID, macaddress_1, macaddress_2, macaddress_3, macaddress_4

6.5 alert2

Add one to a counter if an association packet, alert at end of session based on thresholds.

Packet	
Field	Alert Condition
Packet type	0
Packet_subtype	0
ESSID	N/A
macaddress_1 (hex format)	N/A
macaddress_2 (hex format)	N/A
macaddress_3 (hex format)	N/A
macaddress_4 (hex optional depending on packet type)	N/A
Resulting action	
Alert-text	Alert-2 association flood Followed by SSID, macaddress_1, macaddress_2, macaddress_3, macaddress_4

6.6 alert3

alert3 is triggered if any mac is in badmac list drawn from the badmac file

Field	Alert Condition
Packet type	0
Packet_subtype	
ESSID	N/A
macaddress_1 (hex format)	In Badmaclist or
macaddress_2 (hex format)	In Badmaclist or
macaddress_3 (hex format)	In Badmaclist or
macaddress_4 (hex optional depending on packet type)	In Badmaclist
Resulting action	
	"Alert-3 Blacklisted mac detected " Followed by SSID, macaddress_1, macaddress_2, macaddress_3, macaddress_4

6.7 alert4

Alert if ssid is in the badsids list loaded from badsids file

Field	Alert Condition
Packet type	0
Packet_subtype	
ESSID	In Badssids list
macaddress_1 (hex format)	N/A
macaddress_2 (hex format)	N/A
macaddress_3 (hex format)	N/A
macaddress_4 (hex optional depending on packet type)	N/A
Resulting action	
Alert	Alert-4 Blacklisted SSID detected Followed by SSID, macaddress_1, macaddress_2, macaddress_3, macaddress_4

6.8 alert5

alert5 is fatajack

Field	Alert Condition
Packet_type	0
Packet_subtype	11
ESSID	n/a
macaddress_1 (hex format)	n/a
macaddress_2 (hex format)	n/a
macaddress_3 (hex format)	n/a
macaddress_4 (hex optional depending on packet type)	n/a
auth_type	== "custom"
Resulting action	
	Alert-5 Possible FataJack-custom auth method detected Followed by SSID, macaddress_1, macaddress_2, macaddress_3, macaddress_4

6.9 alert6

Dummy

6.10 alert7

alert-7 tracks Deassscation Response, DeAuthentication Response, assscation Response & Authentication Command failled

Field	Alert Condition
Packet type	0
Packet_subtype	= "*ation Response"
ESSID	N/A
macaddress_1 (hex format)	N/A
macaddress_2 (hex format)	N/A
macaddress_3 (hex format)	N/A
macaddress_4 (hex optional depending on packet type)	N/A
Resulting action	
	Alert-7 Auth Asso Command failled Followed by SSID, macaddress_1, macaddress_2, macaddress_3, macaddress_4

6.11 Alert-8

Alert-8 you are being WLANJACKED if you see more DEAUTH requests in a session than the threshold.

Field	Alert Condition
Packet_type	0
Packet_subtype	12
ESSID	n/a
macaddress_1 (hex format)	n/a
macaddress_2 (hex format)	n/a
macaddress_3 (hex format)	n/a
macaddress_4 (hex optional depending on packet type)	n/a
Resulting action	
	Alert-8 DEAUTH/WLANjack flood SSID, macaddress_1, macaddress_2, macaddress_3, macaddress_4

6.12 Alert9

Alert-9 IS ISSUED WHEN A Single broadcast Deauth IS DETECTED

Field	Alert Condition
Packet type	0
Packet_subtype	Deauthentication
BSID	
macaddress_1 (hex format)	ffffffffffff
macaddress_2 (hex format)	N/A
macaddress_3 (hex format)	N/A
macaddress_4 (hex optional depending on packet type)	N/A
Resulting action	
	Alert-9 DeAuth broadcast SSID, macaddress_1, macaddress_2, macaddress_3, macaddress_4

6.13 Alert10

Alert-10 is triggered when the DeAuth counter exceeds the counter for a session. This is a duplicate – needs to detect an Auth flood

Field	Alert Condition
Packet type	0
Packet_subtype	Deauthentication
BSID	N/A
macaddress_1 (hex format)	N/A
macaddress_2 (hex format)	N/A
macaddress_3 (hex format)	N/A
macaddress_4 (hex optional depending on packet type)	N/A
Resulting action	
	Alert-10 DeAuth counter SSID, macaddress_1, macaddress_2, macaddress_3, macaddress_4

Configuring WIDZ-FG

Widz Future Generation has a very simple configuration script. It deals with a standard probe deployment with a Single NIC. Obviously Widz FG supports multiple NICs and numerous protocols but this has to be done manual.

At the linux prompt and entry the responses as bold and underline.

$Configure_Sensor.sh

Sensor Hostname is sim

do you want to change it

Please enter y or n

Y ↵

enter new hostname

customer1_sensor1↵

You have enter customer1_sensor1 as a hostname

do you want to update /etc/hostname

Please enter y or n

Y ↵

customer1_sensor1 as a hostname

This sets the host name and makes it persistent. It should identify customer name and probe name in the convention used.

Configure_Sensor1.sh

do you want to configure Interface eth0 - Y/N

CTRL-c to end or n to do next configuration stage

Please enter y or n

Y ↵

do you want to use DHCP Y or N ?

Please enter y or n

N ↵

do you want to configure a static ip y / n ?

Please enter y or n

Y ↵

Enter a dot IP address

192.168.0.126 ↵

Enter a dot IP NETMASK

255.255.255.0

Enter a dot IP gateway

192.168.0.1 ↵

The script and the platform can support DHCP but this will not support remote admin access once deployed and so is not recommended

Configure_Sensor2.sh

Configuration stage 2

do you want to configure WIDZ_FG - Y/N

CTRL-c to end or n to do next configuration stage

Please enter y or n

Y ↵

Do you want to (re) create the Access point Baseline

ensure wireless interface wlan2 is installed

Please enter y or n

Y ↵

Do you want to install the config in the current directory

in the production directory y / n ?

you can stop the script, edit the file and rerun the script

to install in the production directory

Please enter y or n

Y ↵

This creates the AP baseline file and stores it in the production directory

Configure_Sensor3.sh

Configuration stage 3

do you want to configure WIDZ_FG boot config - Y/N

CTRL-c to end or n to do next configuration stage

Please enter y or n

Y ↵

Do you want to start at boottime Access point monitoring -widz_apmon

Please enter y or n

Y ↵

Do you want to start at boot attack monitoring -widz_probemon

Please enter y or n

Y ↵

rc update success

This make Widz_apmon and Widz_probemon start at boot-time.

Configure_Sensor4.sh

```
_____________________________________

_                       _
_ CONFIGURATION COMPLETE      _
_                       _
_____________________________________
```

This signifies the end of the guided install.

The only thing needs to be updated is the rsyslog.conf to the DA syslog server

7 Advanced deploy

DA_WIN has an advanced deploy mode which includes multiple NICS and the PI being bowered from the Powered hub.