Localização de Vulnerabilidades de Segurança em Gestores de Dispositivos Wi-Fi com Técnicas de Fuzzing[1]

Manuel Mendonça[1], Nuno Neves[2]

Faculdade de Ciências da Universidade de Lisboa,
Bloco C6, Campo Grande 1749-016 Lisboa - Portugal
[1]manuelmendonca@msn.com, [2]nuno@di.fc.ul.pt

Resumo. As redes sem fios estão a tornar-se banais à medida que cada vez mais equipamentos as suportam. Esta situação cria novas preocupações de segurança às organizações uma vez que os atacantes deixam de necessitar de ter acesso físico à rede que liga os equipamentos, mas apenas precisam de estar na sua proximidade para enviar dados maliciosos para explorar alguma vulnerabilidade. Este trabalho apresenta o Wdev-Fuzzer, uma ferramenta que pode ser utilizada para localizar vulnerabilidades de segurança em gestores de dispositivos Wi-Fi. As experiências realizadas com um dispositivo equipado com o Windows Mobile 5 indicam que o Wdev-Fuzzer pode ser bastante eficaz em descobrir defeitos de concretização, confirmar vulnerabilidades conhecidas e descobrir novos problemas no protocolo Wi-Fi.

Palavras-chave: Vulnerabilidade, segurança, fuzzer, device driver, wi-fi, windows.

1 Introdução

As redes sem fios (WLANs) foram desenvolvidas para ligar elementos de rede, oferecendo aos seus utilizadores a possibilidade de se deslocaram mantendo a conectividade. Estas podem ser utilizadas para aumentar a extensão duma infra-estrutura de rede com fios já existente, ou substitui-la. A redução nos custos que oferecem deve-se não só à contínua redução no preço dos componentes, mas principalmente na poupança com a instalação de cabos de comunicações e de alimentação.

A evolução em termos de velocidade de transmissão nas redes com fios, está agora a ocorrer nas WLANs, à medida que os consumidores exigem maior largura de banda,

[1] Este trabalho foi parcialmente suportado pela UE através dos projectos IST-4-027513-STP (CRUTIAL) e NoE IST-4- 026764-NOE (RESIST), e pela FCT através dos projectos POSC/EIA/61643/2004 (AJECT) e pelo Laboratório de Sistemas de Grande Escala (LASIGE).

maiores distâncias de comunicação, menor consumo de energia e maior interoperabilidade.

Hoje em dia esta tecnologia está a tornar-se banal, e quase todos os computadores portáteis, PDAs, telefones celulares e outros dispositivos estão equipados com suporte para WLANs, invadindo os lares e as empresas.

As WLANs no entanto enfraquecem o perímetro de segurança duma infra-estrutura de rede. Em muitos lugares, tais como aeroportos e centros comerciais, existem dezenas de pontos de rede maliciosos à espera de serem acedidos pelos viajantes. Cada vez que alguém se liga e se autentica, está a transmitir pelo ar o nome e senha do utilizador, e quando acede à Internet possivelmente o número do cartão bancário. As redes de comunicação públicas oferecem oportunidades para que um atacante capture enormes quantidades desse tipo de dados [1].

As redes domésticas sem fios podem ser atractivas para um vizinho malicioso que quer roubar largura de banda, ou então tentar aceder ao disco rígido. As redes das empresas despertam um interesse acrescido na medida em que alguém pode conseguir roubar segredos de negócio, transacções bancárias, dados pessoais ou registos médicos.

Esta situação sucede porque a maioria das WLANs, tanto públicas como privadas, utilizam um tipo de segurança fraca ou muitas vezes até nem sequer utilizam segurança. Isto significa que alguém com um computador portátil e uma placa de rede pode interceptar as comunicações e ler os pacotes enviados ou recebidos pelos legítimos utilizadores.

Apesar de muitas falhas de segurança se deverem a má configuração, algumas são o resultado de concretizações com erros, ou vulnerabilidades, que são exploradas por um ataque de alguém malicioso. Em particular, estes ataques podem ser direccionados contra os gestores de dispositivos (*device drivers* - DD). Os DD virtualizam os detalhes físicos dos equipamentos, estando a tornar-se rapidamente nos componentes mais dinâmicos e de maior dimensão nos sistemas operativos (SO). O seu desenho e desenvolvimento envolvem o conhecimento de diferentes áreas que muitas vezes não são simultaneamente dominadas pelos analistas e programadores. Apesar de hoje em dia serem desenvolvidos com linguagens de alto nível, como "C" ou "C++", continuam a ser difíceis de concretizar e manter. Actualmente, apesar de existirem diversos programas [6][7][7][9][10] que ajudam os programadores a aumentar a qualidade das concretizações, muitos DD apresentam erros que só mais tarde são descobertos.

Por estas razões, é importante o desenvolvimento de técnicas que possibilitem a descoberta de vulnerabilidades em DD. A metodologia a empregar depende se o código é fechado ou aberto (*close/open source*). Nos DD de código aberto, o acesso ao código fonte pode levar a resultados mais rápidos na medida em que se pode ler e procurar falhas de concretização sistematicamente, embora às vezes de forma manual. Em ambos casos pode ser usada uma técnica de caixa negra. Neste tipo de testes o comportamento funcional da unidade em teste (UT) é verificado (resultado de saída) em função do tipo de entrada que lhe é apresentado. A utilização de *reverse engineering* é uma das outras formas de descobrir vulnerabilidades, mas é dispendiosa, demorada, aborrecida e requer conhecimentos profundos da arquitectura do sistema e código máquina.

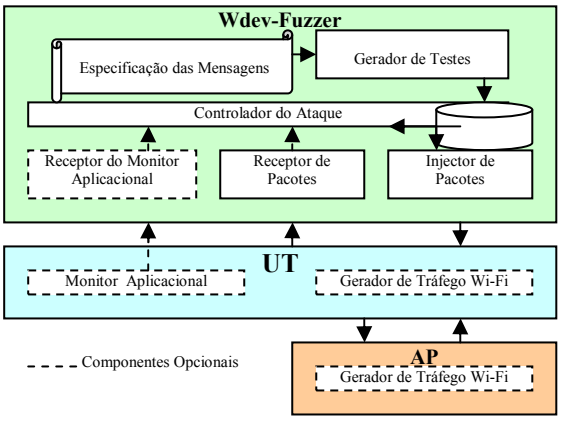

Figura 1. Diagrama de blocos do Wdev-Fuzzer.

Os atacantes utilizam muitas vezes uma metodologia de teste de caixa negra, denominada de *fuzzing* [2]. Esta consiste na apresentação de dados mal formados na interface do sistema e em se observar os seus resultados. Esta técnica é muito eficaz na descoberta dos erros mais óbvios, requerendo alguma sofisticação nas ferramentas que pretendem encontrar erros mais complexos.

Este artigo apresenta o Wdev-Fuzzer, um fuzzer que pode ser usado para a construção de mensagens mal formadas, e para executar ataques contra um sistema alvo independentemente do seu tipo de comunicação. A concretização actual da ferramenta apenas suporta o Wi-Fi, no entanto esta pode ser facilmente estendida para suportar outros tipos de protocolos de comunicação, tais como a Ethernet, Ird e Bluetooth.

O artigo descreve ainda uma utilização da ferramenta na avaliação do DD Wi-Fi no SO Windows Mobile 5. Os cenários de teste simulam um ataque a um dispositivo que esteja à procura de ligação a um ponto de acesso (*Access Point* – AP) ou que já esteja ligado. Pretendia-se por exemplo compreender se com este tipo de ataques era possível provocar a paragem do SO ou se este pode ser comprometido remotamente.

Este tipo de estudo é importante devido ao crescente interesse neste tipo de tecnologia e pela forma como se consegue facilmente enviar dados maliciosos para uma WLAN. Á medida que o standard Wi-Fi evolui, este tipo de ferramentas ajudam a compreender quão confiáveis os sistemas se tornam e que soluções são necessárias construir e aplicar para melhorar a segurança.

2 Wdev-Fuzzer

2.1 Arquitectura do Wdev-Fuzzer

O Wdev-Fuzzer pode ser dividido em 8 componentes (ver Figura 1). A Especificação das Mensagens é um ficheiro de texto que define as mensagens do protocolo como uma junção de componentes. Cada componente da mensagem é definido no mesmo ficheiro, utilizando tipos de dados intrínsecos ao Gerador de Testes.

Para cada tipo de dados, existe um operador de fuzzer que atribui valores específicos aos componentes de acordo com um conjunto de regras. Durante a geração dos pacotes Wi-Fi, a Especificação das Mensagens é utilizada pelo Gerador de Testes, e esses operadores são utilizados para preencher o valor de cada componente da mensagem. Estendendo os tipos básicos e os operadores é possível construir outros tipos de dados e novos valores, e ir de encontro à especificidade de outros protocolos.

O Controlador do Ataque controla a actividade do Injector de Pacotes. Decide qual o próximo ataque a executar com base na informação fornecida pelo Receptor do Monitor Aplicacional e Receptor de Pacotes, segundo um critério predeterminado.

O Monitor Aplicacional e o correspondente Receptor são componentes opcionais usados para troca de informação sobre o estado da UT, para determinar o sucesso do ataque e decidir o próximo ataque a realizar.

O Gerador de Tráfego é usado para forçar a existência de comunicação entre a UT e o AP. Desta forma podemos observar qual o efeito dum ataque nas comunicações em curso.

O Receptor de Pacotes captura quaisquer mensagens que o UT envie para o meio de transmissão, para ajudar a determinar o sucesso do ataque e decidir o próximo a ser executado.

Finalmente, o Injector de Pacotes é usado para enviar os ataques à UT.

Para que o Wdev-Fuzzer possa ser usado em vários protocolos, é necessário fazer-se algumas adaptações, como definir o ficheiro Especificação das Mensagens e usar funções específicas que permitam concretizar o Injector e Receptor de Pacotes.

2.2 Wdev-Fuzzer e o Protocolo 802.11

A arquitectura do protocolo IEEE 802.11 é composta por vários elementos que interagem entre si, com o objectivo de disponibilizar uma WLAN que suporta a mobilidade dos dispositivos (ou estações) de uma forma transparente para as camadas superiores de software.

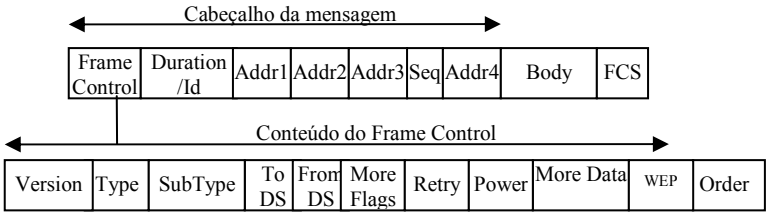

Figura 2. Estrutura duma mensagem Wi-Fi.

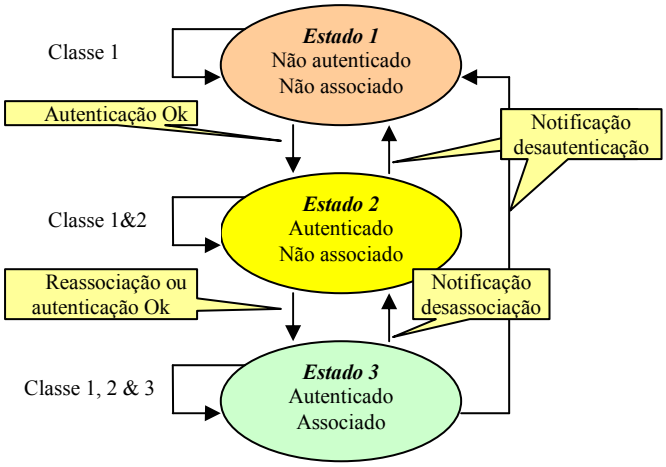

Figura 3. Relação entre o estado do funcionamento do protocolo Wi-Fi e as classes de mensagens.

O bloco constituinte mais simples de uma rede IEEE 802.11 é chamado de *conjunto de serviços básico* (*Basic Service Set* - BSS). A *área de cobertura* da BSS consiste na zona onde as estações membro (STA) podem comunicar directamente umas com as outras.

A forma mais elementar duma rede Wi-Fi (denominada IBSS) é constituída apenas por duas STA, que conseguem comunicar directamente uma com a outra. Esta rede é também referida como rede ad-hoc por ser muitas vezes formada sem planeamento.

Em vez de existir de forma independente, a BSS pode estar interligada por um sistema de distribuição (*distribution system* – DS) a outras BSS. Um AP é uma estação que para além de agir como um componente da rede, disponibiliza também acessos ao DS.

As mensagens do protocolo IEEE 802.11 (ver Figura 2) são compostas por elementos de dimensão fixa e de dimensão variável (TLV). Os elementos de dimensão fixa surgem sempre no início da mensagem. Um campo TLV é composto por três elementos, um elemento que univocamente identifica o tipo de dados transportado (T-TAG), um elemento que transporta a dimensão dos dados (L-Length) e finalmente os dados em si (V-Value). O tipo de mensagem determina a presença de cada elemento TLV.

Os tipos de mensagens do protocolo IEEE 802.11 que podem ser trocadas dependem do estado dos STAs (ver Figura 3) e são agrupadas em classes que correspondem ao estado da estação. No estado 1, apenas é permitido o envio de mensagens de Classe 1. No estado 2 é permitido o envio de mensagens de Classe 1 e 2, no estado 3 é permitido o envio de todas as classes de mensagens (ver Tabela 1).

O standard 802.11 é bastante vasto e as revisões efectuadas pelo comité que o regula podem ser bastante extensas. Para ultrapassar este problema, assim como conseguir concretizar subgrupos do standard que ainda não tenham sido regulamentados, quase todos os fabricantes de equipamentos sem fios juntaram-se na Wi-Fi Alliance.

Este grupo está dedicado a gerir a especificação Wi-Fi (um subgrupo do standard 802.11) e define o que se deve fazer em caso de ambiguidades. Este comité garante a interoperabilidade entre fabricantes e assegura que todos os produtos certificados com o logótipo "Wi-Fi" trabalham em conjunto.

Nos cenários de teste, usou-se o Wdev-Fuzzer para simular um AP malicioso e avaliar um dispositivo configurado como um STA. O módulo do dispositivo que foi testado foi o DD que concretiza o Wi-Fi. O sistema operativo que foi considerado nas experiências foi o Windows Mobile 5.

A avaliação do comportamento do equipamento a funcionar como AP está fora do âmbito deste estudo, e foi deixado para trabalho futuro.

2.3 Valores Testados

Como foi explicado anteriormente, um fuzzer envia dados maliciosos quando está a testar um equipamento. A Tabela 2 apresenta os valores que são aplicados a cada um dos campos das mensagens utilizadas nas experiências. O carácter 'X' indica a aplicação do operador sobre o elemento da mensagem, o carácter '-' indica o oposto.

O operador "Omisso" omite um elemento da mensagem. O operador "Repetir" reproduz várias vezes o mesmo elemento na mensagem. Os operadores "Todos os bits a Zero" e "Todos os bits a um" são auto explicativos. Os operadores "MIN" e "MAX" produzem respectivamente o valor mínimo e máximo que um elemento pode ter, tal como especificado no standard. Muitas vezes os operadores "Todos os bits a zero" e o "MIN" produzem o mesmo valor assim como os operadores "MAX" e "Todos os bits a um". Nestes casos não faz sentido aplicar os operadores "MIN-1" e "MAX+1", uma vez que levam ao mesmo valor produzido pelos operadores "Todos os bits a um" e "Todos os bits a zero" (respectivamente).

O operador "Aleatório" produz valores aleatórios entre os valores produzidos pelos operadores "MIN" e "MAX". Por fim, o operador "Valor específico" produz sempre o mesmo valor. É usado por exemplo para forçar um determinado endereço de MAC.

2.4 Cenários Testados

No início considerou-se testar o dispositivo em todos os seus diferentes estados, tal como representado na Figura 3. No entanto, em situações reais o estado 2 é alcançado apenas por breves períodos, daí terem sido apenas considerados os estados 1 e 3.

Os testes foram aplicados em 3 cenários diferentes. No cenário A, o dispositivo encontra-se no estado 1, não estando associado ou autenticado no AP. No cenário B, o dispositivo encontra-se no estado 3, associado e autenticado ao AP, mas sem que as mensagens tenham autenticação. O cenário C é semelhante ao cenário B, mas as mensagens são autenticadas.

Tabela 1. Mensagens Wi-Fi testadas.

Mensagem	Tipo	Sub Tipo	Para AP	Da AP	Classe
Association Request	Mgt	0	X	-	2
Association Response	Mgt	1	-	X	2
Reassociation Request	Mgt	2	X	-	2
Reassociation Response	Mgt	3	-	X	2
Probe Request	Mgt	4	X	-	1
Probe Response	Mgt	5	-	X	1
Beacon	Mgt	8	-	X	1
Disassociation	Mgt	10	X	X	2
Authentication	Mgt	11	X	X	1
Deauthentication	Mgt	12	X	X	1,3
Power Save	Ctrl	10	X	-	3
Request To Send	Ctrl	11	X	-	1
Clear to Send	Ctrl	12	-	X	1
Acknowledgment (Ack)	Ctrl	13	X	X	1
Contention Free (CF) End	Ctrl	14	-	X	1
CF-End + CF-Ack	Ctrl	15	-	X	1
Data	Data	0	X	X	1,3

(Mgt – Management, Ctrl – Control)

Tabela 2. Valores testados.

Operador	Campo de dimensão fixa	Campo de dimensão variável (TLV)
Omisso	-	X
Repetir	-	X
Todos os bits a zero	X	X
MIN-1	X	X
MIN	X	X
MIN+1	X	X
Aleatório	X	X
Valor específico	X	X
MAX-1	X	X
MAX	X	X
MAX+1	X	X
Todos os bits a um	X	X

Tabela 3. Resultados esperados.

ID	Descrição
F1	Não foram detectados problemas na execução.
F2	Foi detectada uma mensagem incorrecta.
F3	UT deixou de estar associado.
F4	UT deixou de estar autenticado.
F5	O Monitor Aplicacional sofreu um bloqueio.
F6	O SO sofreu um bloqueio.
F7	O sistema sofreu um crash e reiniciou.

Tabela 4. Valores de F1 detalhados.

ID	Descrição
F1A	Os pontos de acesso detectados estão correctos.
F1B	O ponto de acesso não é detectado mas deveria ser.
F1C	O ponto de acesso é detectado mas não deveria ser.

2.5 Resultados Esperados

O Gerador de Testes usa a Especificação das Mensagens e os operadores de fuzz para construir mensagens Wi-Fi. Dependendo dos valores produzidos, o dispositivo ao receber mensagens pode ter um erro ou pode acabar por as conseguir processar de alguma maneira.

A Tabela 3 resume os resultados esperados do sistema quando sujeito às experiências. Esta tabela foi elaborada com base nos resultados dos testes preliminares, na literatura disponível e trabalho relacionado.

É de notar que apesar do modo F1 representar o caso em que o sistema aparentemente continua a funcionar sem se terem verificado problemas, tal não significa que o pacote tenha sido bem processado. Sempre que o teste é realizado com base em pacotes do tipo "Beacon" ou "Probe Response", o Monitor Aplicacional fornece informação sobre os pontos de acesso detectados. Nestes casos pode-se estender o valor F1 em três outras categorias, como representado na Tabela 4.

O resultado F1A representa o caso em que o Monitor Aplicacional reporta correctamente os AP detectados. O valor F1B aplica-se aos casos em que o Monitor não detecta o AP mas deveria, e o valor F1C aplica-se aos casos em que o Monitor detecta o AP mas não deveria.

O valor F2 significa que o Receptor de Pacotes detectou o envio de um pacote incorrecto por parte do dispositivo.

O valor F3 indica que o dispositivo ficou desassociado do AP em consequência dum ataque. O valor F4 significa que o ataque fez com que o dispositivo deixe de estar autenticado perante o AP. Os valores F3 e F4 só poderão ocorrer quando o dispositivo estiver no estado 3.

Caso o Monitor Aplicacional fique bloqueado, esse facto é assinalado com o valor F5. Isto indica que um problema no gestor de dispositivos foi propagado para a aplicação. Se o sistema operativo sofrer um bloqueio, o resultado do teste é assinalado com o valor F6. No caso de ocorrer um crash e consequente reboot o resultado do teste é F7.

3 A Infra-estrutura de Teste

O principal problema no desenvolvimento duma infra-estrutura de teste para redes Wi-Fi consiste em reunir uma placa de comunicações e correspondente gestor de periférico que permita a injecção e recepção de todas as mensagens independentemente do destino.

Na família de sistemas operativos Windows a especificação de gestor de periférico da interface de rede (*Network Driver Interface Specification* - NDIS) define uma

interface aplicacional (API) para as placas de comunicações (*Network Interface Cards* - NIC's). Os detalhes de concretização do hardware do NIC são encapsulados pelo gestor do periférico de tal forma que todos os NICs do mesmo tipo comunicação (e.g., Ethernet) possam ser acedidos usando a mesma API. As aplicações agem sobre os NIC através duma pilha de gestores de dispositivos. Cada gestor adiciona funcionalidades à infra-estrutura de comunicações.

No princípio desenvolveu-se um DD que foi colocado na pilha de gestores de periféricos na esperança de interceptar todos os pacotes enviados e recebidos por cada NIC (assim como instruções dadas pelo SO para o NIC). Verificou-se que o protocolo Wi-Fi é concretizado no DD do NIC, emulando para as camadas superiores da pilha o protocolo Ethernet. Isto significa que o gestor de periféricos desenvolvido apenas capturava mensagens Ethernet, em vez de capturar as mensagens Wi-Fi conforme se desejava.

Existe contudo a possibilidade de se capturar mensagens Wi-Fi na família de SO Windows, embora nenhuma delas seja muito fácil de conseguir. Uma das formas consiste em utilizar uma interface disponibilizada pelo gestor do dispositivo do NIC, mas infelizmente tanto quanto nos foi possível averiguar nenhum fabricante assim a disponibiliza. Outra forma consiste em desenvolver um gestor de periférico que faça directamente a interface com o hardware (e nesse caso é necessário conhecer os detalhes de concretização da placa de comunicações). Airpcap [5] é um produto e interface aplicacional que pode ser utilizada para capturar mensagens Wi-Fi, mas é baseado num hardware e o gestor de periférico proprietário. Logo existe um custo associado ao seu uso.

A solução que acabou por ser escolhida baseia-se em se usar uma plataforma Linux. Existem diversas placas de comunicação e DD para o SO Linux que suportam a injecção e captura de mensagens Wi-Fi. A melhor forma de as encontrar é fazer uma pesquisa na Internet, e procurar por sniffers Wi-Fi e ver os NICs compatíveis. A infra-estrutura de teste que foi utilizada (representada na Figura 4) é composta por 4 componentes que são descritos nas próximas secções.

3.1 Controlador

O Wdev-Fuzzer usa uma máquina instalada com uma distribuição Linux, que emprega o gestor de periférico MadWi-Fi [3] para placas de rede baseadas nos integrados da Atheros. O Injector de Pacotes recorre a uma biblioteca de funções genérica para injectar e capturar mensagens Wi-Fi, resultado do projecto Lorcon [4], que foi modificada para alcançar os nossos objectivos.

O Receptor do Monitor recebe todas as mensagens enviadas pelo Monitor Aplicacional instalado na UT e envia esta informação para o Controlador do Ataque para sincronizar o próximo teste. O Receptor de Pacotes informa o Controlador do Ataque de cada mensagem enviada pelo UT, com o objectivo de detectar qualquer comportamento inesperado.

3.2 Unidade em Teste

A UT é o dispositivo Wi-Fi alvo das experiências. Executa o Monitor Aplicacional que regularmente se liga ao Receptor do Monitor informando-o da lista de AP detectadas. Esse comportamento é útil principalmente quando se está a testar mensagens "Beacon" ou "Probe Response", uma vez que a lista de AP detectados é crucial para determinar o correcto funcionamento dos mecanismos de processamento de erros. A actual versão tem bastantes limitações, como ser incapaz de detectar bloqueios quer do DD como do SO, para além de ser incapaz de realizar reinícios automáticos da UT. Casa ocorra um bloqueio o dispositivo tem que ser reiniciado à mão.

3.3 PC Hospedeiro

A UT está fisicamente ligada a um PC através duma porta USB. Através do software instalado (Windows XP e o ActiveSync da Microsoft), o Monitor Aplicacional consegue comunicar com o Receptor do Monitor Aplicacional através dum canal fora de banda, deixando o meio de comunicações Wi-Fi livre para as experiências.

3.4 Ponto de Acesso

Para evitar a introdução de código específico e complexo no Wdev-Fuzzer, é utilizado um AP para mudar o estado da UT do estado 1 para o estado 3.

4 Resultados Experimentais

O Wdev-Fuzzer foi concretizado num computador Dell Optiplex 170L Pentium IV, instalado com o SO Fedora Core 6. Utiliza como meios de comunicação uma placa NetGear WPN311 e o controlador Ethernet integrado na placa mãe.

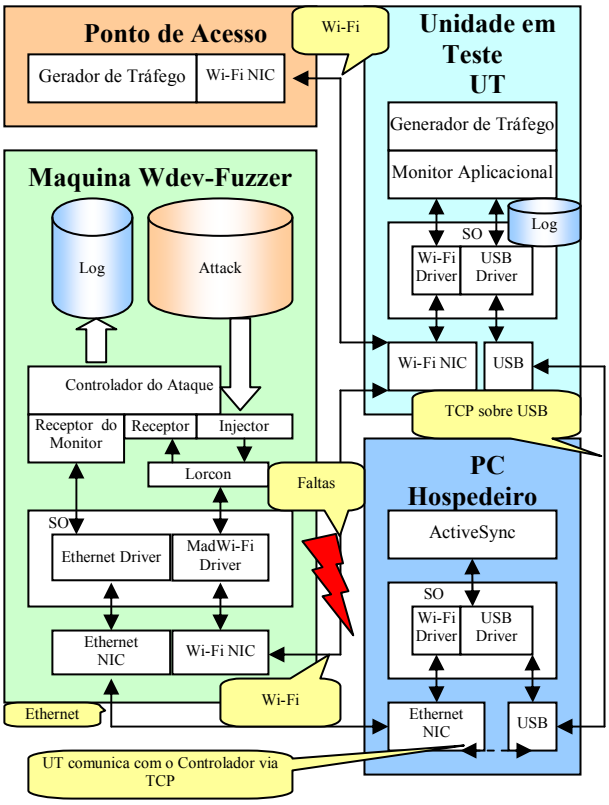

Figura 4. A infra-estrutura de teste.

O computador hospedeiro da UT foi concretizado numa máquina HighScreen Pentium IV, com Windows XP Professional e o Microsoft ActiveSync 4.1 build 4841. Esta máquina tem instalado uma placa Ethernet que liga ao Controlador através duma rede Ethernet a 100Mbits. Foi-lhe adicionada uma placa GigaByte AirCruiser GN-WP01GS, que em conjunto com o software fornecido, implementa o AP.

A UT consiste no HP iPAQ hw6915 PDA, instalado com Windows Mobile 5, e equipado com um circuito Wi-Fi Texas Instruments. Foi colocado a uma distância aproximada de 2m do Controlador e do AP, e está ligada a uma porta USB do computador hospedeiro.

5 Resultados Observados

Os resultados obtidos nos testes estão exibidos nas Tabela 5 e Tabela 6. Estas tabelas só mostram os resultados das mensagens que assumem a UT como um STA (ver Tabela 1). Os restantes tipos de mensagens que foram enviados deram sempre F1, o que demonstra uma boa capacidade do Windows Mobile em ignorar mensagens que não deveria receber, ainda que tenham como destino o seu endereço MAC.

A primeira coluna dessas tabelas mostra o campo da mensagem que foi testado. A segunda coluna mostra quantos valores diferentes foram testados.

As colunas que se seguem mostram os resultados obtidos para cada uma das mensagens identificadas na primeira linha.

Para aumentar a legibilidade da tabela apenas quando o resultado é diferente de F1 se colocam na mesma célula o número de resultados obtidos.

Quando existe diferença entre o número de testes realizados e o número de resultados obtidos deve-se assumir que a diferença corresponde ao resultado F1.

5.1 Resultados observados no cenário A

Os testes do cenário A realizam-se com a UT no estado 1. Este estado é alcançado depois de se ligar o dispositivo Wi-Fi, garantindo que este não se associa ou autentica no AP.

A Tabela 5 mostra que com a excepção das mensagens "Beacon" e "Probe Response", o dispositivo não apresentou problemas, mostrando-se bastante imune às tentativas de ataque.

Contudo, foram observados alguns resultados interessantes. Por exemplo, o DD Wi-Fi do Windows Mobile inclui na lista de AP detectados todas as APs que são anunciadas em mensagens "Beacon" independentemente do endereço de destino (ver linha "DA" da tabela). Esta situação não é correcta uma vez que as AP devem anunciar-se apenas com o endereço MAC de broadcast. Por outro lado, quando o campo "SSID" inclui o código do caracter ASCII '0x00', o AP não é incluído nessa lista, o que constitui uma forma efectiva de esconder uma rede deste SO. Todos os restantes caracteres são exibidos.

Quando existem vários campos "SSID", o DD assume como válida a identificação do último campo.

Sempre que o dispositivo recebe uma mensagem "Beacon" que inclua o elemento TLV com TAG=5, dimensão 255 e valor hexadecimal 0xFF, o SO sofre um bloqueio (ver linha "TIM" e coluna "Beacon"). Até ao momento não foi possível concluir se é possível explorar esta vulnerabilidade com o objectivo de tomar conta do sistema, de qualquer forma, constitui uma forma de provocar um ataque de negação de serviço (*Denial of Service* - DoS).

Os resultados observados para a mensagem "Probe Response" são idênticos aos observados para a mensagem "Beacon", no entanto não foi observado qualquer bloqueio do SO ao enviar o elemento TIM nas condições enunciadas.

5.2 Resultados Observados no Cenário B

No cenário B, o UT foi associado ao AP utilizando o protocolo de autenticação aberto (*open*). Neste cenário o Wdev-Fuzzer realiza exactamente os mesmos testes que realizou no cenário anterior.

Tabela 5. Resultados do cenário A.

Campo da Mensagem	#Testes e Resultados	CTS	Ack	CF-End	CF-End CF-Ack	Data	Assoc. Resp	Reass. Resp	Prob. Resp	Beacon	Disass.	Auth.	Deauth.
Protocol* Version	4	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	F1	F1	F1
To/From* DS	4	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	F1	F1	F1
More Flags*	11	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	F1	F1	F1
Retry*	11	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	F1	F1	F1
Power* Management	11	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	F1	F1	F1
More*Data	11	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	F1	F1	F1
WEP*	11	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	F1	F1	F1
Order*	11	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	F1	F1	F1
Duration	3500	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	F1	F1	F1
RA/Addr1	8	F1	F1	F1	F1	F1	-	-	-	-	-	-	-
TA/Addr2	8	-	-	-	-	F1	-	-	-	-	-	-	-
DA	8	-	-	-	-	-	F1	F1	7x F1C	7x F1C	F1	F1	F1
SA	8	-	-	-	-	-	F1	F1	F1A	F1A	F1	F1	F1
AID	15	-	-	-	-	-	F1	F1	-	-	-	-	-
BSS ID	8	-	-	F1	F1	-	F1	F1	F1	F1	F1	F1	F1
Addr3	8	-	-	-	-	F1	-	-	-	-	-	-	-
Sequence Control	10	-	-	-	-	F1	F1	F1	F1	F1	F1	F1	F1
Addr4	7	-	-	-	-	F1	-	-	-	-	-	-	-
Payload	7	-	-	-	-	F1	-	-	-	-	-	-	-
TimeStamp	6	-	-	-	-	-	-	-	F1A	F1A	-	F1	-
Beacon** Interval	2700	-	-	-	-	-	-	-	F1A	F1A	-	F1	-
Capabilities**	2050	-	-	-	-	-	F1	F1	F1A	F1A	-	F1	-
SSID**	1275	-	-	-	-	-	F1	F1	32x F1B	32x F1B	F1	F1	F1
Supported** Rates	256	-	-	-	-	-	F1	F1	F1A	F1A	F1	F1	F1
FH** Parameter	256	-	-	-	-	-	F1	F1	F1A	F1A	F1	F1	F1
DS** Parameter	256	-	-	-	-	-	F1	F1	F1A	F1A	F1	F1	F1
CF** Parameter	256	-	-	-	-	-	F1	F1	F1A	F1A	F1	F1	F1
IBSS** Parameter	256	-	-	-	-	-	F1	F1	F1A	F1A	F1	F1	F1
TIM**	256	-	-	-	-	-	F1	F1	F1	1x F6	F1	F1	F1
Reason Code	15	-	-	-	-	-	-	-	-	-	F1	-	F1
Status Code	5	-	-	-	-	-	F1	F1	-	-	-	F1	-
Auth. Algorithm Nbr	5	-	-	-	-	-	-	-	-	-	-	F1	-
Auth. Transaction Nbr	5	-	-	-	-	-	-	-	-	-	-	F1	-
Other TLV**	1255	-	-	-	-	-	F1	F1	F1	F1	F1	F1	F1

*Frame Control; **TLV

A Tabela 6 exibe os resultados observados neste cenário. Os resultados dos testes realizados às mensagens "Beacon" e "Probe Response" foram iguais ao do cenário A, o que não constitui surpresa, uma vez que o mecanismo de detecção de AP não deve ser alterado por se encontrar no estado 3.

As experiências com as mensagens "Disassociation" e "Deauthentication" confirmaram o sucesso na execução de ataques de DoS quando os endereço MAC de destino são iguais ao da UT ou ao endereço de broadcast. Ainda assim, existem algumas verificações que são feitas às flags da secção "Frame Control" das mensagens (ver Figura 4). Verifica-se que nem todas as combinações dessas flags ("To/From DS", "More Flags", "Retry", "Power Management", "More Data", "WEP" e "Order") resultam num ataque com sucesso.

Durante essas experiências, a comunicação entre os Geradores de Tráfego na UT e no Wdev-Fuzzer é interrompida em consequência do dispositivo ficar desassociado do AP. A comunicação é sempre restabelecida no final dos ataques no caso dos testes com mensagens do tipo "Deautentication", mas o mesmo não acontece com as mensagens "Disassociation". Este comportamento revela por um lado que os ataques realizados com este último tipo de mensagens podem ser mais prejudiciais do que o realizado com as mensagens "Deautentication", por outro que a pilha do protocolo TCP/IP pode ter algum problema.

Não foram encontrados problemas nos ataques realizados aos outros tipos de mensagens.

5.3 Resultados Observados no Cenário C

Os testes realizados no cenário C envolveram a associação da UT ao AP utilizando o protocolo de autenticação de chave partilhada WEP. Embora não seja o protocolo mais seguro, o objectivo consistia em verificar em que medida isso alteraria os resultados em relação aos testes realizados no cenário B. As experiências demonstraram não existir diferenças.

6 Trabalho Relacionado

As ferramentas e métodos de injecção de faltas permitem injectar faltas no software e hardware do sistema alvo em teste (ver por exemplo, [12][13][19][25][27][30]). Ao provocar e reproduzir a ocorrência de eventos irregulares e invulgares, permitem por exemplo avaliar a capacidade do sistema em as suportar. As falhas provocadas são na maior parte dos casos relativamente simples, tais como as faltas provocadas por alterações de bits em memórias, registos, ou instruções. Estas técnicas têm sido utilizadas em diversas actividades, principalmente na validação do hardware ou verificação dos mecanismos de tolerância a falhas e não para a descoberta de vulnerabilidades.

Os testes de robustez podem ser aplicados para caracterizar o comportamento dos componentes de software quando expostos as entradas excepcionais ou condições de operação extremas.

Tabela 6. Resultados dos cenários B e C.

Campo da Mensagem	#Testes e Resultados	CTS	Ack	CF-End	CF-End CF-Ack	Data	Assoc. Resp	Reass. Resp	Prob. Resp	Beacon	Disass.	Auth.	Deauth.
Protocol* Version	4	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	1x F3	F1	1x F4
To/From* DS	4	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	3x F3	F1	3x F4
More Flags*	2	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	1x F3	F1	1x F4
Retry*	2	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	1x F3	F1	1x F4
Power* Management	2	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	1x F3	F1	1x F4
More* Data	2	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	1x F3	F1	1x F4
WEP*	2	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	1x F3	F1	F1
Order*	2	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	1x F3	F1	1x F4
Duration	3500	F1	F1	F1	F1	F1	F1	F1	F1A	F1A	3500x F3	F1	3500x F4
RA/Addr1	8	F1	F1	F1	F1	F1	-	-	-	-	-	-	-
TA/Addr2	8	-	-	-	-	F1	-	-	-	-	-	-	-
DA	8	-	-	-	-	-	F1	F1	7x F1C	7x F1C	2x F3	F1	2x F4
SA	8	-	-	-	-	-	F1	F1	F1A	F1A	1x F3	F1	1x F4
AID	15	-	-	-	-	-	F1	F1	-	-	-	-	-
BSS ID	8	-	-	F1	F1	-	F1	F1	F1	F1	1x F3	F1	1x F4
Addr3	8	-	-	-	-	F1	-	-	-	-	-	-	-
Sequence Control	10	-	-	-	-	F1	F1	F1	F1	F1	F3	F1	F4
Addr4	7	-	-	-	-	F1	-	-	-	-	-	-	-
Frame Body	7	-	-	-	-	F1	-	-	-	-	-	-	-
TimeStamp	6	-	-	-	-	-	-	-	F1A	F1A	-	F1	-
Beacon Interval	2700	-	-	-	-	-	-	-	F1A	F1A	-	F1	-
Capabilities**	2050	-	-	-	-	-	F1	F1	F1A	F1A	-	F1	-
SSID**	1275	-	-	-	-	-	F1	F1	32x F1B	32x F1B	1275x F3	F1	1275x F4
Supported** Rates	256	-	-	-	-	-	F1	F1	F1A	F1A	256x F3	F1	256x F4
FH** Parameter	256	-	-	-	-	-	F1	F1	F1A	F1A	256x F3	F1	256x F4
DS** Parameter	256	-	-	-	-	-	F1	F1	F1A	F1A	256x F3	F1	256x F4
CF** Parameter	256	-	-	-	-	-	F1	F1	F1A	F1A	256x F3	F1	256x F4
IBSS** Parameter	256	-	-	-	-	-	F1	F1	F1A	F1A	256x F3	F1	256x F4
TIM**	256	-	-	-	-	-	F1	F1	F1A	1x F6	256x F3	F1	256x F4
Reason Code	15	-	-	-	-	-	-	-	-	-	256x F3	-	256x F4
Status Code	5	-	-	-	-	-	F1	F1	-	-	-	F1	-
Auth. Algorithm Nbr	5	-	-	-	-	-	-	-	-	-	-	F1	-
Auth. Transaction Nbr	5	-	-	-	-	-	-	-	-	-	-	F1	-
Other TLV**	1255	-	-	-	-	-	F1	F1	F1	F1	1255x F3	F1	1255x F4

*Frame Control; **TLV

A maior parte das ferramentas de teste de robustez têm tido como alvo os sistemas operativos de uso genérico, fornecendo entradas inválidas às funções que fazem parte da interface aplicacional. Por exemplo o Ballista foi aplicado a diversos SO que implementam o standard POSIX standard [20]. Shelton et al. fizeram um estudo comparativo entre seis variantes do Windows [28]. Outros estudos com estas ferramentas incluem microkernels de tempo real [14] e sistemas middleware de suporte como CORBA [22][24]. Recentemente esta técnica foi aplicada às interfaces dos DD dos SO [15][16][17][23]. Contudo, os testes de robustez têm sido principalmente aplicados às interfaces internas dos sistemas, que não podem ser exploradas directamente por um adversário externo. Por essa razão, a descoberta de problemas pode não corresponder a descobertas de vulnerabilidades.

Fuzzing é uma técnica que gera dados inválidos para os passar a aplicações alvo para processamento e verificar se a aplicação falha enquanto consome esses dados [11]. Uma falha indica a presença de algum tipo de vulnerabilidade, e que potencialmente pode ser explorada por algum adversário. Fuzz foi um dos primeiros projectos a explorar estas ideias, e foi desenhado para testar comandos UNIX (e foi mais tarde aplicado a outros SO) [26]. O objectivo consistia em gerar grandes quantidades de sequências aleatórias de caracteres que eram usadas como argumentos da linha de comando dos programas. Muitos programas falharam no processamento de argumentos ilegais provocando o crash da aplicação. Nos anos mais recentes, os fuzzers têm sido incluídos em ferramentas mais inteligentes e de carácter menos aleatório, capazes de testar diferentes tipos de componentes de software (ver por exemplo, [18][21][29][31]).

7 Conclusões e Trabalho Futuro

Neste artigo apresentou-se o desenho da ferramenta Wdev-Fuzzer, que tem como objectivo encontrar vulnerabilidades na concretização dos DD de dispositivos Wi-Fi. A ferramenta funciona duma forma automática e é capaz de testar os elementos de diversas mensagens durante os diversos estados do protocolo.

Foi realizada uma avaliação experimental com um dispositivo instalado com Windows Mobile que tem sido extensivamente testado ao longo dos anos. Não obstante, descobriu-se uma vulnerabilidade ainda não conhecida. Foi possível reproduzir as situações já conhecidas de DoS relacionadas com mensagens de controlo específicas. Conseguiu-se perceber alguns detalhes de concretização que podem comprometer a interoperabilidade entre sistemas Wi-Fi e que afectam o funcionamento das aplicações.

Estão identificadas várias áreas de evolução deste trabalho, como a verificação do comportamento dos DD Wi-Fi noutros SO e a aplicação da ferramenta noutros protocolos.

As falhas na concepção e implementação das WLANs podem colocar em causa a segurança nas organizações que as utilizam, devido à facilidade com que se concretiza com sucesso um ataque conhecido e a descoberta de novas vulnerabilidades. Por outro lado dado o reduzido custo dos seus componentes e a sua ampla disponibilidade nos

novos equipamentos informáticos apelam à sua utilização que deve ser criteriosamente justificada para não comprometer a segurança dos seus utilizadores.

Referencias

[1] Ferramentas de sniffing, Jun. 2007, http://insecure.org/

[2] Fuzzers, Jun. 2007, http://www.infosecinstitute.com/blog/2005/12/fuzzers-ultimate-list.html

[3] Madwifi driver, Jun. 2007, www.madwifi.org

[4] Lorcon, Jun. 2007, http://802.11ninja.net/lorcon

[5] Airpcap, Jun. 2007, http://www.cacetech.com/products/airpcap.htm

[6] Microsoft Corporation, "Introducing Static Driver Verifier", Mai. 2006.

[7] A. Chou, J. Yang, B. Chelf, S. Hallem, and D. Engler, On u-kernel construction, Proceedings of the Symposium on Operating Systems Principles, Dez. 1995, pp. 237–250.

[8] B. Ford, G. Back, G. Benson, J. Lepreau, A. Lin, and O. Shivers, The Flux OSKit: a substrate for OS language and resource management, Proceedings of the Symposium on Operating Systems Principles, Out. 1997, pp. 38–51.

[9] M. Swift, B. Bershad, and H. Levy, Improving the reliability of commodity operating systems, Proc. of the Symp. on Operating Systems Principles, Out. 2003, pp. 207–222.

[10] M. Young, M. Accetta, R. Baron, W. Bolosky, D. Golub, R. Rashid, and A. Tevanian, Mach: A new kernel foundation for UNIX development, Proceedings of the Summer USENIX Conference, Jun. 1986, pp. 93–113.

[11] P. Oehlert, Violating Assumptions with Fuzzing, IEEE Security & Privacy, pages 58-62, Mar./Abr. 2005.

[12] J. Aidemark, J. Vinter, P. Folkesson, and J. Karlsson, GOOFI: Generic Object-oriented Fault Injection Tool, Proceedings of the Int. Conference on Dependable Systems and Networks, pp. 83–88, Jul. 2001.

[13] J. Arlat, A. Costes, Y. Crouzet, J.-C. Laprie, and D. Powell, Fault Injection and Dependability Evaluation of Fault-tolerant Systems. IEEE Transactions on Computers, 42(8):913.923, Ago. 1993.

[14] J. Arlat, J.-C. Fabre, M. Rodríguez and F. Salles, Dependability of COTS Microkernel-Based Systems, IEEE Trans. on Computers, vol. 51, no. 2, pp. 138-163, 2002.

[15] A. Albinet, J. Arlat, and J.-C. Fabre, Characterization of the Impact of Faulty Drivers on the Robustness of the Linux Kernel, Proceedings of the Int. Conf. on Dependable Systems and Networks, Jun. 2004.

[16] J. Durães and H. Madeira, Characterization of Operating Systems Behavior in the Presence of Faulty Drivers through Software Fault Emulation, Proc. of the Pacific Rim Int. Symposium On Dependable Computing, pp. 201-209, Dez. 2002.

[17] A. Johansson, and N. Suri, Error Propagation Profiling of Operating Systems, Proc. of the Int. Conference on Dependable Systems and Networks, Jun 2005.

[18] T. Biege, Radius Fuzzer, Set. 2005, http://www.suse.de/~thomas/index.html.

[19] J. Carreira, H. Madeira, and J. G. Silva, Xception: Software Fault Injection and Monitoring in Processor Functional Units, Proc. of the Int. Working Conference on Dependable Computing for Critical Applications, pp. 135–149, Jan. 1995.

[20] P. Koopman, J. DeVale, The Exception Handling Effectiveness of POSIX Operating Systems, IEEE Tran. on Software Engineering, vol. 26, no. 9, pp. 837-848, Set. 2000.

[21] A. Greene, SPIKEfile, September 2005. http://labs.idefense.com/labs-software.php?show=14.

[22] E. Marsden, J.-C. Fabre and J. Arlat, Dependability of CORBA Systems: Service Characterization by Fault Injection, Proceedings of the 21st Int. Symposium on Reliable Distributed Systems, pp. 276-285, Jun. 2002.

[23] M. Mendonça, N. Neves, Robustness Testing of the Windows DDK, Proceedings of the Int. Conference on Dependable Systems and Networks, Jun. 2007

[24] J. Pan, P. J. Koopman, D. P. Siewiorek, Y. Huang, R. Gruber and M. L. Jiang, Robustness Testing and Hardening of CORBA ORB Implementations, Proceedings of the International Conference on Dependable Systems and Networks, pp. 141-150, Jun. 2001.

[25] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer, Fault Injection Techniques and Tools, IEEE Computer, 30(4):75.82, Abr. 1997.

[26] B. P. Miller, L. Fredriksen, and B. So, An empirical study of the reliability of UNIX utilities, Communications of the ACM, 33(12):32–44, 1990.

[27] G. A. Kanawati, N. A. Kanawati, and J. A. Abraham. Ferrari: A tool for the Validation of System Dependability Properties, Proceedings of the Int. Symp. on Fault-Tolerant Computing, pp. 336.344, Jun. 1992.

[28] C. Shelton, P. Koopman and K. D. Vale, Robustness Testing of the Microsoft Win32 API, Proceedings of the Int. Conference on Dependable Systems and Networks, pages 261-270, Jun. 2000.

[29] M. Sutton, FileFuzz, Set. 2005. http://labs.idefense.com/labs-software.php? show=3.

[30] T. K. Tsai and R. K. Iyer, Measuring Fault Tolerance with the FTAPE Fault Injection Tool, Int. Conference on Modeling Techniques and Tools for Computer Performance Evaluation, volume 977 of Lecture Notes in Computer Science, pp. 26–40. Set. 1995.

[31] J. Röning, et al., PROTOS – Security Testing of Protocol Implementations, Computer Engineering Laboratory, University of Oulu, 1999–2003. http://www.ee.oulu.fi/research/ouspg/protos/.