UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

FERNANDO DUTRA FAGUNDES MACEDO

WINFIRMAMENT: Adaptação do Injetor de Falhas de Comunicação FIRMAMENT para Ambientes Windows

Trabalho de Graduação

Prof. Dr. Sérgio Luis Cechin
Orientador

Porto Alegre, julho de 2012

AGRADECIMENTOS

Agradeço ao Prof. Doutor Sérgio Luis Cechin a orientação deste trabalho, suas ideias, sua paciência e seu esforço dedicados a mim durante a realização deste trabalho.

Agradeço aos meus pais, Gregório Bohrer Macedo e Daniela Dutra Fagundes, os quais me deram toda a oportunidade para a realização do curso.

Dedico, em especial, este trabalho a minha esposa, Letícia Dilélio Araújo, a qual me acompanha desde o início desta jornada, todo o incentivo dado, amor e paciência nos momentos de ausência.

Agradeço aos meus irmãos, Eduardo Dutra Fagundes Macedo e Henrique Dutra Fagundes Macedo, a colaboração em diversas etapas do curso.

Agradeço aos meus ex-colegas Jorel Settin, Maurício dos Santos Condessa e Rodrigo Jaureguy Dobler todo o apoio dado durante o curso e momentos especiais que tornaram a realização do curso menos árdua.

Finalmente, gostaria de agradecer aos colegas e ex-colegas de trabalho que também deram as suas contribuições ao longo do curso.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
IDS	Intrusion Detection System
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPS	Intrusion Prevention System
JVM	Java Virtual Machine
LAN	Local Area Network
LLC	Logical Link Control
LSP	Layered Service Provider
MAC	Medium Access Control
NDIS	Network Driver Interface Specification
RFC	Request for Comments
RMI	Remote Method Invocation
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
TDI	Transport Driver Interface
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WAN	Wide Area Network
WFP	Windows Filtering Platform
WDK	Windows Driver Kit

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A execução de testes que cubram o maior número de casos de uso é essencial para a validação de uma aplicação. Atualmente, muitas delas dependem da comunicação entre dispositivos, os quais podem estar localizados nas mais diversas localidades e sujeitos a diferentes tipos de falhas. A ocorrência de falhas de comunicação deve ser, também, simulada nesses casos para uma maior cobertura dos casos de uso por meio de injeção de falhas.

Este trabalho analisa a ocorrência de falhas de comunicação e suas implicações. Nele, é proposta a adaptação da ferramenta de injeção de falhas de comunicação FIRMAMENT - voltada para ambientes GNU/Linux e já adaptada para plataformas Android - para os ambientes Windows. Para isso, é escolhida a plataforma Windows Filtering Platform devido às suas capacidades de filtragem e manipulação de mensagens de redes. São descritos os principais métodos de filtragem de mensagens nesses ambientes, suas capacidades e limitações. Além disso, são executados testes de validação da ferramenta WINFIRMAMENT.

Palavras-chave: falhas de comunicação, injeção de falhas, FIRMAMENT, Windows, Windows Filtering Platform.

WINFIRMAMENT: Adaptação do Injetor de Falhas de Comunicação FIRMAMENT para Ambientes Windows

ABSTRACT

Testing perfoming that covers a bigger number of use cases are essencial to validate an application. Nowadays, many of them depends on communication between devices, that can be located in any place and subject to different kinds of faults. The fault communication occurrence must be simulated in this cases to have a bigger covering to use cases by fault injection.

This work analyses the communication fault occurrence and its implications. It proposes an adaptation of the communication fault injection tool FIRMAMENT - native of GNU/LINUX environments, ported to Android platform - to the Windows environments. For this, the Windows Filtering Platform is chosen due to its network messages filtering and manipulation ability. The main messages filtering methods are described in this environments, its capacities and limitations. Besides, validation tests of WINFIRMAMENT tool are perfomed.

Keywords: communication faults, fault injection, FIRMAMENT, Windows, Windows Filtering Platform.

1 INTRODUÇÃO

Atualmente, a maior parte das aplicações depende de dados e serviços localizados em outros dispositivos. A separação de dados é necessária para que seja provido um maior isolamento, já que um dispositivo indisponível não implica um serviço final indisponível. Além disso, com a separação, é possível prover mecanismos de redundância através da replicação de um mesmo serviço em diversos dispositivos distribuídos.

Assim como em qualquer sistema computacional, os ambientes que utilizam a comunicação entre os dispositivos estão sujeitos à ocorrência de falhas. Tais falhas podem ser ocasionadas tanto pelos dispositivos finais, bem como por aqueles intermediários envolvidos na comunicação. Por isso, faz-se necessário realizar a injeção de falhas a fim de alcançar a dependabilidade.

Segundo levantamento da empresa Net Applications (NET APPLICATIONS 2012), acima de 80% das computadores utilizam sistemas operacionais Windows (dados de junho de 2011 a abril de 2012). Tal levantamento mostra a importância no desenvolvimento de ferramentas para o aprimoramento de aplicações que são utilizadas nesses sistemas.

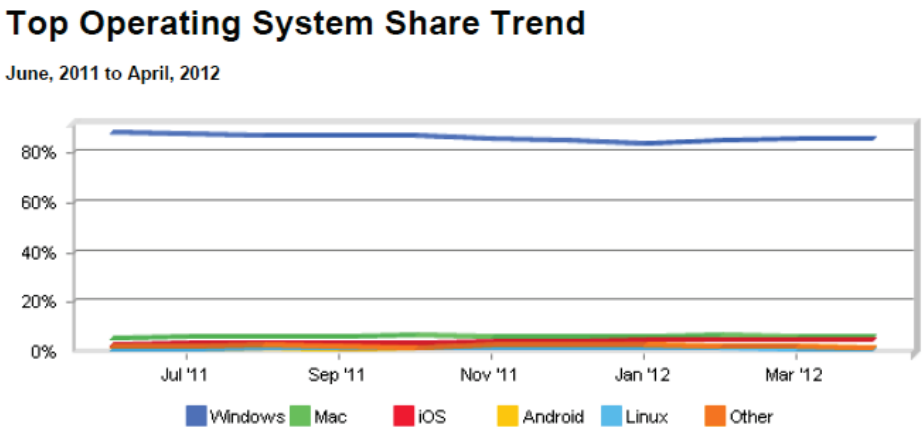

Figura 1.1: Sistemas operacionais mais utilizados no período de abr/11 a jun/12.

Neste trabalho, são analisados os mecanismos de filtragem presente nos ambientes Windows e avaliados quanto às suas capacidades. A ferramenta de injeção de falhas de mensagens FIRMAMENT é adaptada para esses ambientes com o uso da plataforma Windows Filering Platform. Além disso, são realizados testes para validar o seu funcionamento, bem como exemplificar falhas presentes na comunicação entre dispositivos.

São objetivos deste trabalho: avaliar o funcionamento do Windows Filtering Platform e adaptar o FIRMAMENT aos ambientes Windows.

Através do estudo do Windows Filtering Platform, é possível verificar quais são as funcionalidades para a implementação da ferramenta FIRMAMENT nos ambientes Windows.

A adaptação do FIRMAMENT permitirá que aplicações desenvolvidas nas plataformas Windows sejam validadas quanto ao funcionamento na presença de falhas de comunicação.

Além disso, testes são apresentados para verificar o correto funcionamento do WINFIRMAMENT.

Através deste trabalho, pôde-se avaliar os mecanismos presentes nos ambientes Windows para a filtragem de mensagens, quais as capacidades e limitações de cada um deles. A plataforma Windows Filtering Platform foi escolhida para implementação do WINFIRMAMENT devido às recomendações do desenvolvedores de não utilizar os outros mecanismos, uma vez que estão em desuso e, em versões futuras, não estarão presentes nesses ambientes.

Além disso, foi possível realizar testes do Windows Filtering Platform, os quais estão descritos no final deste trabalhos.

Neste trabalhos são apresentadas as principais falhas de comunicação presentes em sistemas computacionais bem como algumas das suas causas. Os objetivos de realizar a injeção de falhas de comunicação indicam a necessidade de utilizar uma ferramenta desse tipo. Cada um dos mecanismos de filtragem de mensagens presentes nos ambientes Windows é, brevemente, explicado a fim de compará-los e justificar a utilização da plataforma Windows Filtering Platform na construção do WINFIRMAMENT.

As ferramentas necessárias nos desenvolvimento de uma ferramenta de injeção de falhas de comunicação são exploradas e mostradas as suas funcionalidades e o papel que cada uma desempenha. Por fim, são apresentados os testes realizados bem como mostradas as limitações da ferramenta desenvolvida, as quais devem ser exploradas em trabalhos futuros.

2 ESPECIFICAÇÃO DO PROJETO

Neste capítulo, são apresentados os principais conceitos utilizados neste trabalho, bem como são apresentadas as ferramentas utilizadas na construção e na realização de testes do WINFIRMAMENT.

2.1 Injeção de Falhas

O principal objetivo da injeção de falhas é a validação de sistemas através da inserção de falhas artificiais para verificar a ocorrência de erros e defeitos. Busca-se, com isso, alcançar a dependabilidade, a capacidade em oferecer um serviço confiável (AVIZIENIS; LAPRIE; RANDELL, 2001). Neste capítulo, são descritos conceitos básicos relacionados a injeção de falhas, bem como descritas as suas técnicas e mostrados alguns de seus exemplos.

Alguns termos da área de tolerância a falhas - na qual a injeção de falhas se insere - são, com frequência, utilizados de maneira inadequada. Isso se deve a similaridade que alguns deles possuem. Além disso, a falta de correspondência com termos da língua portuguesa colabora com os seus usos inadequados. Um exemplo disso são os termos falha, erro e defeito, cujos significados são mostrados a seguir.

Abaixo são mostrados alguns dos principais termos utilizados em injeção de falhas (AVIZIENIS; LAPRIE;RANDELL, 2001).

Confiabilidade - atendimento aos requisitos estabelecidos.
Confidencialidade - resguardo das informações.
Defeito - desvio de especificação.
Dependabilidade - qualidade de um serviço provido.
Disponibilidade - período de funcionamento de um sistema.
Integridade - consistência de dados.
Erro - propriedade que pode provocar um defeito em um sistema.
Falha - causa de um erro.
Latência de falha - tempo entre a ocorrência de uma falha e a ocorrência de um erro.
Mantenabilidade - capacidade de reparo de um sistema.
Segurança (safety) - proteção contra danos.
Segurança (security) - proteção contra falhas.

Um produto, após produzido, deve ser submetido a um conjunto de testes para avaliar o seu funcionamento dentro de suas especificações e submetido a testes que incluem a injeção de falhas. Os mecanismos de injeção de falhas podem ser utilizados em qualquer fase do desenvolvimento de um produto.

A antecipação de testes é vantajosa, uma vez que detectado um problema no desenvolvimento, o tempo dispendido é menor quanto mais próximo do início do desenvolvimento

do produto.

A injeção de falhas pode antecipar a ocorrência de erros e defeitos. A visualização de tais erros e defeitos podem permitir:

- Avaliar o impacto, a importância e os danos ocorridos na presença de um defeito.
- Determinar limites seguros e outros parâmetros de funcionamento de um sistema.
- Detectar problemas no projeto e desenvolvimento.

A injeção de falhas pode ser realizada tanto através de softwares como através de hardware (Hsueh, 1997). A injeção de falhas por hardware pode ser classificada em dois tipos: com ou sem contato. Já a injeção de falhas por software pode ser realizada em tempo de compilação ou em tempo de execução.

A injeção de falha através de hardware com contato é realizada com inserção de pinos ou soquetes no sistema alvo. Uma vez conectados ao sistema, eles podem inserir corrente ou tensão adicionais ao sistema.

A injeção de falha através de hardware sem contato é realizada através de um fenômeno físico no qual o gerador não entra em contato com o sistema alvo. Alguns exemplos de fenômenos utilizados para isso: radiação iônica e interferência eletromagnética.

A injeção de falha através de software em tempo de compilação é aquele em o código alvo da injeção de falhas é modificado a fim de verificar a ocorrência de erros.

A injeção de falha através de software em tempo de execução é executada com auxílio de software ou hardware e não necessita modificação no código alvo. Nesse método, são utilizados os seguintes mecanismos: esgotamento de tempo, geração de exceções e inserção de código.

2.2 Falhas de Comunicação

Toda a comunicação entre dispositivos está sujeita a falhas que podem ocorrer por diversos motivos. Muitas vezes, aquele que realiza a comunicação entre dois dispositivos desconhece os detalhes da topologia de conexão entre os dispositivos, a qual pode dinâmica, bem como os índices de qualidade de tal conexão. Podemos classificar essas falhas, basicamente, como atraso e alteração das mensagens. Além disso, podemos estendê-las para uma melhor compreensão. São elas: perda, alteração, atraso e duplicação de mensagens.

A perda de uma mensagem é caracterizada pelo não recebimento desta por um dispositivo que troca mensagens ou o recebimento em um tempo inválido, superior àquele que determinada aplicação pode aguardar. Suas causas estão ligadas a falhas dos dispositivos finais, dos dispositivos intermediários e dos enlaces envolvidos na comunicação.

O transbordamento de dados, que pode ocorrer tanto nos dispositivos finais como nos intermediários, é a perda de dados devido à incapacidade de um dispositivo manipulá-los devido a falta de recurso para tal. Essa incapacidade é, normalmente, ocasionada pela taxa de recebimento maior que o dispositivo pode manipular.

As falhas em enlaces também podem provocar a perda de dados e estão relacionados à sua natureza. Um enlace ótico pode, por exemplo, ter perdas devido à presença de sujidade em seus conectores. Já um enlace do tipo ethernet pode, por exemplo, sofrer interferências eletromagnéticas de modo que seus dados sejam perdidos.

As perdas, no entanto, podem ser induzidas pelos dispositivos. Um dispositivo pode induzir o descarte devido a políticas de descarte, que podem estar relacionadas a uma política de qualidade de serviço, ao mal comportamento na comunicação ou um determinado conjunto de regras que as causem.

A alteração de mensagens é a troca de qualquer parte da mensagem de tal forma que a mensagem recebida é diferente daquela enviada, ou seja, há a violação da integridade. A alteração de mensagens pode ser induzida ou não. As alterações induzidas são aquelas provocadas por equipamentos que visam adequar a mensagem para devido fim, tal como adequação a uma RFC. Já as alterações não induzidas decorrem de defeitos nos equipamentos e enlaces envolvidos na comunicação por motivos diversos.

O atraso de uma mensagem é o seu recebimento em um tempo maior do que aquele tempo estimado. Esse atraso pode ter causas em diversos fatores como, por exemplo, aumento do caminho a ser tomado pelas mensagens, sobrecarga de dispositivos envolvidos na comunicação, políticas de qualidade de serviço, nas quais determinadas mensagens são priorizadas frente a outras. Até mesmo a perda de uma mensagem pode ser devido ao atraso desta e ser descartada por não ter mais validade.

A duplicação de mensagem ocorre quando uma mesma mensagem é recebida por um mesmo dispositivo mais de uma vez. Essa duplicação pode ocorrer devido a mecanismos de redundância, nos quais uma mesma mensagem é enviada diversas vezes a fim de aumentar a probabilidade de que ela chegue ao seu destino, devido a mecanismos de controle, nos quais uma mensagem não confirmada é reenviada e, até mesmo em comutadores, os quais enviam as mensagens a todos os seus vizinhos quando desconhecem a localização do próximo dispositivo responsável por encaminhar ou receber a mensagem.

2.3 Drivers

Nesta seção são descritas as funcionalidades de um driver, como interage com o sistema operacional, com as aplicações e o com dispositivo por ele controlado. Além disso, são apresentados alguns exemplos de drivers.

Um driver (MICROSOFT, 2012) é, basicamente, um software que permite a comunicação entre um dispositivo qualquer e o sistema operacional. Um driver (CORBET; RUBINI; KROAH-HARTMAN, 2005) implementa todas as funcionalidades que o dispositivo pode oferecer a uma aplicação através do sistema operacional. A implementação de tais funcionalidades é específica de cada dispositivo, pois depende da construção do seu hardware. O sistema operacional comunica-se com o driver através de chamadas bem conhecidas, comuns a cada classe de drivers. Como exemplo de chamada, temos a leitura de um caractere de um teclado.

Alguns drivers, por respeitar determinados padrões, são genéricos e atendem a uma variedade de dispositivos semelhantes de fabricantes diferentes. Tal padronização visa uma maior compatibilidade do sistema operacional com os dispositivos. Teclados são alguns dos dispositivos que podem funcionar com drivers genéricos devido a sua simplicidade e, também, devido à necessidade para o funcionamento da maioria dos sistemas computacionais pessoais.

Os drivers podem, também, ser encadeados a fim de definir diferentes camadas de abstração e diminuir a complexidade de programação. Além disso, tal encadeamento permite que um mesmo driver de uma camada superior possa ser usado em diferentes dispositivos.

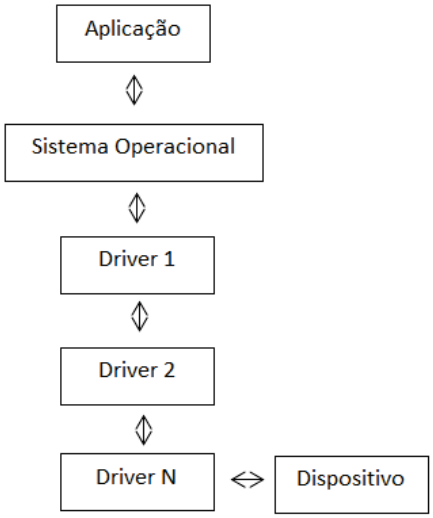

Figura 2.1: Diagrama de interação entre aplicação, sistema operacional e drivers.

Uma vez que os drivers são os softwares que controlam os dispositivos, qualquer dispositivo tem associado a ele um driver. Exemplos de drivers: drivers de impressoras, drivers de vídeo, drivers de áudio, drivers de rede, entre outros.

2.4 Filtragem de Mensagens

A construção de um injetor de falhas de comunicação depende, basicamente, de mecanismos de filtragem de mensagens. Através da filtragem, é possível tomar uma ação de acordo com a mensagem recebida. A seguir, são mostrados alguns dos mecanismos disponíveis para as plataformas Windows.

2.4.1 Winsock Layered Service Providers

A fim de facilitar a criação de serviços e funções de rede, a Microsoft disponibiliza os chamados provedores de serviço em camada para Winsock (MICROSOFT, 2012). Cada provedor de serviço em camada (LSP) implementa atividades de intercepção e modificação de mensagens de acordo com a sua construção. Tal modelo permite uma maior flexibilidade nas atividades de filtragem de mensagens, pois um provedor de serviço pode ser adicionado entre quaisquer dois outros.

Assim como em outros modelos em camadas, uma mensagem deve passar pela primeira camada até a última. Em qualquer parte do processo, as mensagens podem ser modificas ou alteradas. Um exemplo de uso de um LSP é a filtragem de URL não permitidas, serviço do chamado controle dos pais que está presente em algumas versões do sistema operacional Windows.

2.4.2 Filter Hook Drivers

Os drivers de filtragem (Filter-Hook Drivers) são drivers do modo kernel (MICROSOFT, 2012), utilizados apenas na filtragem de mensagens e não na modificação destas. Como eles estendem as funcionalidades do driver de filtragem da camada IP, não podem ser utilizados, por exemplo, para a filtragem de quadros Ethernet.

Esses drivers sinalizam ao driver da camada IP a ação a ser tomada: repassar a men-

sagem às outras camadas, eliminar a mensagem ou deixar a decisão da ação a ser tomada para a camada IP. São compatíveis com as versões de Windows 2000 e superiores. Apesar disso, seu uso não é recomendado a partir do Windows Vista, já que, a partir dessa versão, é disponibilizada a plataforma Windows Filtering Platform.

Uma importante restrição desse tipo de driver é que apenas um deles pode ser utilizado por vez, não permitindo uma construção mais completa, tal como o encadeamento de drivers. Além disso, não permitem a visualização da área de dados de um pacote IP.

2.4.3 Firewall Hook Drivers

Os drivers de firewall (Firewall Hook Drivers) também são drivers do modo kernel e são semelhantes aos drivers de filtragens. Possuem, no entanto, duas vantagens na sua utilização. A primeira é a possibilidade de fazer chamadas a mais de uma função de filtragem, executadas de acordo com as suas prioridades. Já a segunda, é a possibilidade de verificar o conteúdo da área de dados das mensagens.

A mesma recomendação da não utilização desse tipo de driver em sistemas Windows Vista e superiores é indicada, uma vez que as suas funcionalidades são cobertas pela plataforma Windows Filtering Platform. Apesar de seu nome, não é recomendada a construção de uma firewall com esse tipo de drivers, uma vez que não pode atuar nas camadas inferiores.

2.4.4 NDIS

A especificação de interface de driver de rede, Network Driver Interface Specification (NDIS), criada pela Microsoft juntamente com a 3Com, é uma interface de programação de aplicações, application programming interface (API), para interfaces de rede. A NDIS abstrai o hardware de rede dos seus drivers e atua na subcamada LLC (Logical Link Control) da segunda camada do modelo de referência MR-OSI, a camada de enlace.

A NDIS é suportada desde o MS-DOS, NDIS 2.0, até a versão mais recente da família Windows, o Windows 8, NDIS 6.30.

2.4.5 TDI Filter Driver

A interface TDI (Transport Driver Interface) 2010) fornece recursos para facilitar o uso de protocolos da camada de transporte. Um driver de filtragem TDI, está localizado logo acima da camada de transporte e intercepta a comunicação entre esta camada e a camada de sessão. Existe pouca documentação sobre esta interface, no entanto suas funcionalidade são cobertas pelo Windows Filtering Platform.

2.4.6 Comparação dos Mecanismos de Filtragem de Mensagens

Abaixo é mostrada uma tabela comparativa entre os mecanismos de filtragem existentes para os ambientes Windows. A plataforma Windows Filtering Platform (WFP) foi escolhida no desenvolvimento do WINFIRMAMENT por conter os requisitos para isso, além de ser o mecanismo recomendado a ser utilizado frente aos outros, uma vez que eles deixarão de existir em futuras versões desses ambientes.

Mecanismo de Filtragem	Múltiplos Filtros	Modificação de Mensagens	Camada de Atuação
Winsock LSP	Sim	Sim	Transporte
Filter Hook Drivers	Não	Não	Rede
Firewall Hook Drivers	Sim	Sim	Rede
NDIS	Sim	Sim	Enlace
TDI Filter Driver	Sim	Sim	Transporte
WFP	Sim	Sim	Rede

Tabela 2.1: Mecanismos de Filtragem de Mensagens

2.5 Ferramentas de Desenvolvimento e Apoio

A construção de uma ferramenta de injeção de falhas de comunicação necessita de diversos aplicativos para a sua construção, bem como são necessárias ferramentas que permitam avaliar o funcionamento daquilo que é desenvolvido.

Nesta seção, são apresentadas as ferramentas utilizadas durante o desenvolvimento do WINFIRMAMENT.

2.5.1 Windows Driver Kit

O Windows Driver Kit (WDK) (MICROSOFT, 2012) é um conjunto de ferramentas fornecido pela Microsoft para a construção de drivers para os seus sistemas operacionais. Esse conjunto é integrado ao Microsoft Visual Studio, conjunto de ferramentas de desenvolvimento de softwares, e permite ao usuário construir e testar os drivers desenvolvidos, bem como depurar os drivers para verificar os detalhes de sua implementação.

Para manter compatibilidade com versões dos drivers desenvolvidos, a ferramenta permite gerar código compatível com versões anteriores do sistema operacional. Na sua versão 8, por exemplo, pode-se desenvolver drivers para os sistemas operacionais Windows 8, Windows 7 e Windows Vista.

A ferramenta é acompanhada de documentação para diversos tipos de drivers e um conjunto de exemplos funcionais na forma de código para que o desenvolvedor possa, através deles, construir drivers semelhantes derivados dos primeiros. Alguns tipos de drivers cobertos pela documentação e com exemplos: áudio, rede, impressão, sensores, armazenamento, entre outros.

2.5.2 Microsoft Visual Studio

O Microsoft Visual Studio (MICROSOFT, 2012) é um conjunto de ferramentas voltado ao desenvolvimento de aplicações em diversas linguagens de programação. As ferramentas inclusas nesse conjunto permitem o desenvolvimento completo de aplicações durante todas as suas fases, desde a especificação até os testes de homologação. Além disso, são disponibilizadas ferramentas colaborativas para um desenvolvimento em grupo.

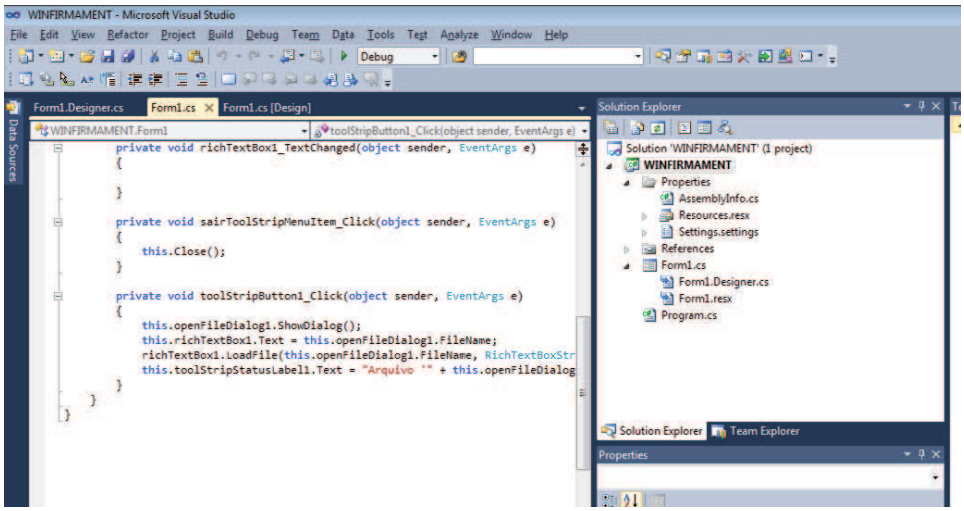

Figura 2.2: Microsoft Visual Studio 2010.

2.5.3 DEV C++

O DEV-C++ (BLOODSHEDSOFTWARE, 2012) é uma ferramenta de código aberto utilizada para o desenvolvimento de aplicações nas linguagens C e C++. Uma vez que o DEV-C++ utiliza o GCC, compilador nativo de ambientes GNU/LINUX, a adaptação de códigos é facilitada quando comparada com a adaptação por meio de outras ferramentas, tal como o Microsoft Visual Studio.

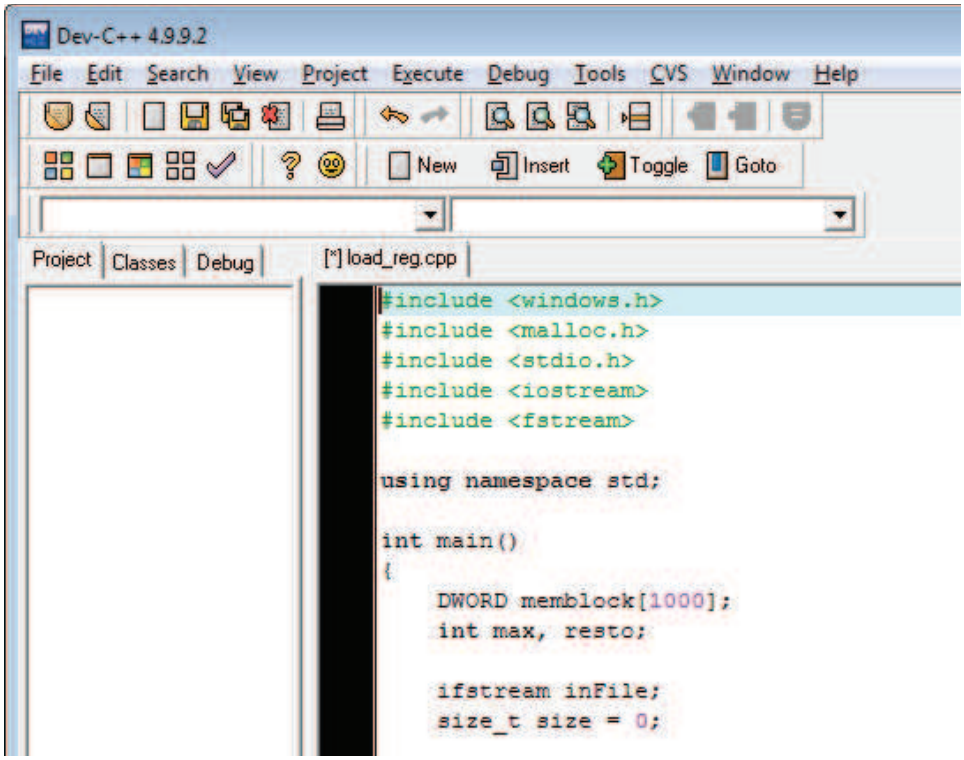

Figura 2.3: Dev-C++ 4.9.9.2.

2.5.4 IPERF

O Iperf é uma ferramenta multiplataforma de código aberto voltada à medida do desempenho de banda dos protocolos UDP e TCP (IPERF, 2012). Através de seu uso, é

possível obter dados da qualidade da rede, tais como a largura de banda disponível, o atraso, o jitter, a taxa de perdas entre dois dispositivos. Seu uso é indispensável no levantamento dos dados de qualidade de uma rede, pois, por exemplo, o uso de uma simples transferência de arquivos não caracteriza a qualidade de transmissão, bem como não é capaz de identificar alguns problemas nessa transmissão.

```
root@fernando-VirtualBox:~# iperf -c 192.168.0.150 -u
------------------------------------------------------------
Client connecting to 192.168.0.150, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  224 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.151 port 56465 connected with 192.168.0.150 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.309 ms    0/  893 (0%)
root@fernando-VirtualBox:~#
```

Figura 2.4: Instância cliente do Iperf.

2.5.5 OSR Driver Loader

O OSR Driver Loader é uma ferramenta voltada ao suporte de desenvolvimento de drivers nos ambientes Windows (OSR, 2012). Seu funcionamento consiste, basicamente, em carregar drivers, seus registros, inicialização e parada dos drivers. Essa ferramenta facilita o desenvolvimento de um driver, pois tais funções disponíveis devem ser implementadas pelo aplicativo de instalação do driver e aqueles outros que o utilizarem.

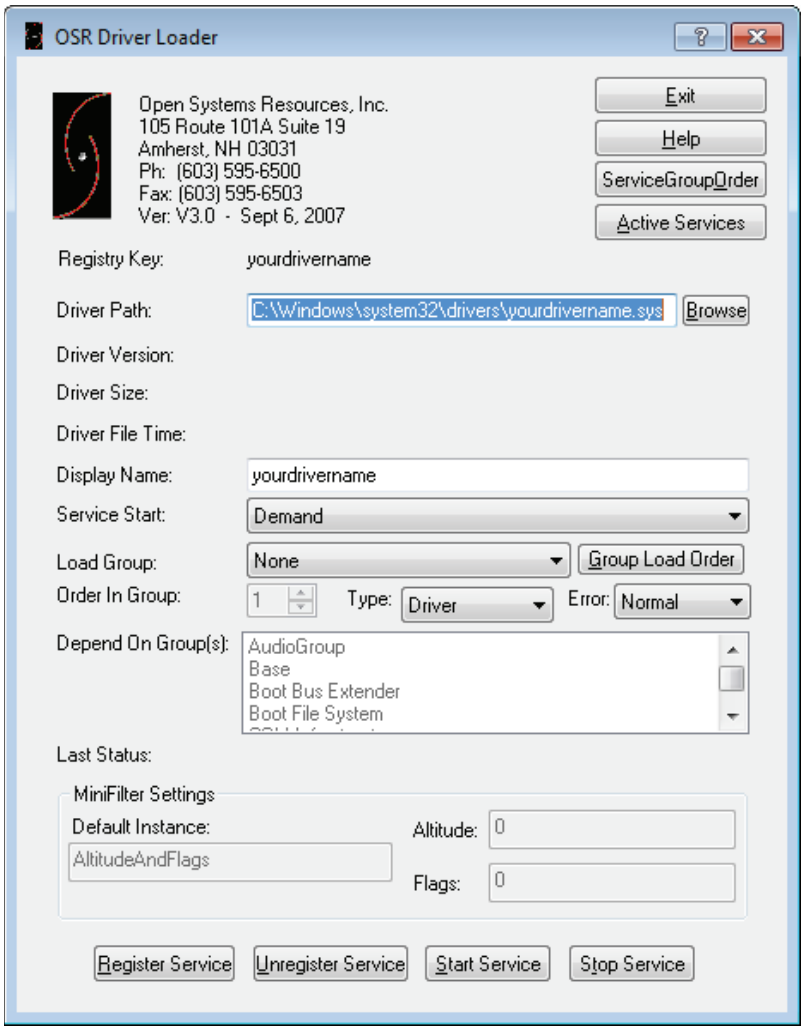

Figura 2.5: OSR Driver Loader.

2.5.6 DebugView

DebugView é uma ferramenta utilizada para monitorar as saídas de depuração em uma máquina qualquer que utilize o sistema operacional Windows (MICROSOFT, 2012). Possui capacidade para salvamento das saídas, para busca, para filtragem, entre outros. No modo kernel, através da chamada DbgPrint, um driver pode escrever dados, seja para diagnóstico ou simples demonstração de funcionamento, e obtê-los no DebugView.

A aplicação faz parte de um conjunto de ferramentas de sistemas e de informações técnicas chamada Sysinternals, criado em 1996 e hoje mantido pela Microsoft.

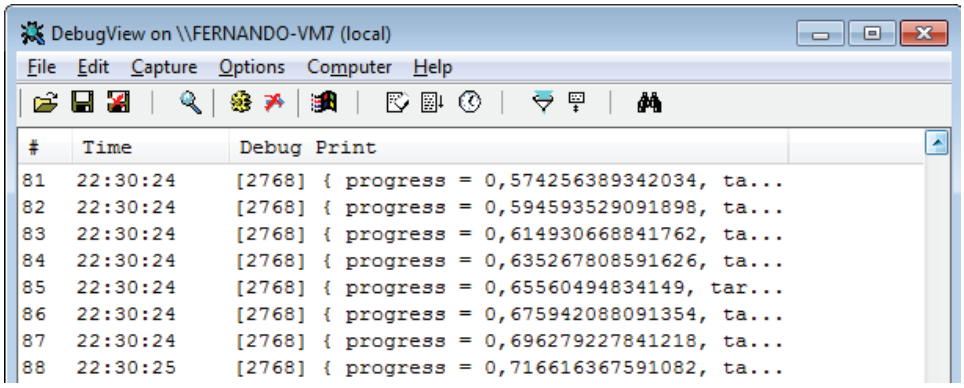

Figura 2.6: DebugView.

2.6 Windows Filtering Platform

A plataforma Windows Filtering Platform (WFP) (MICROSOFT, 2012) é um conjunto de ferramentas utilizadas para a criação de aplicações que utilizem a filtragem de mensagens. Tal filtragem pode ocorrer na camada de rede e camadas superiores. Os dados podem ser filtrados e modificados em qualquer um dos sentidos, entrada e saída.

Suas ações são baseadas em um conjunto de ganchos e um motor de filtragem que coordena as atividades de rede.

A plataforma surgiu para substituir antigas tecnologias de filtragem a partir do Windows XP. Com exceção das filtragens na camada MAC, é recomendado converter os componentes criados com outras tecnologias para utilizar a WFP. Isso se deve ao fato de que, com o surgimento da WFP, tais ferramentas serão descontinuadas gradativamente.

A plataforma Windows Filtering Platform foi escolhida para a construção do WINFIRMAMENT devido à recomendação de uso dos desenvolvedores frente aos outros mecanismos de filtragem. Ela possui, conforme a sua documentação, as capacidades necessárias no desenvolvimento do WINFIRMAMENT além de ser, atualmente, o mais novo mecanismo de filtragem de mensagens nos ambientes Windows.

A plataforma WFP é composta por cinco componentes: o motor de filtragem, o motor básico de filtragem, os Shims, as chamadas, e a API. A função de cada um deles é, brevemente, explicada abaixo.

Motor de Filtragem - núcleo da infraestrutura de filtragem multicamadas.

Motor Básico de Filtragem - controla a operação da plataforma WFP.

Shims - atuam verificando a ação que deve ser tomada de acordo com o motor de filtragem.

Chamadas - funções utilizadas pelos drivers e utilizadas na filtragem.

API - conjunto de dados e funções para o desenvolvimento de aplicações de filtragem.

O uso da plataforma apresenta diversas vantagens em relação a outras tecnologias. Algumas delas são descritas a seguir. São elas:

- Suporte a pilha dupla IPv4/IPv6;
- Maior estabilidade;
- Possui um grande conjunto de filtros;
- Maior capacidade de diagnóstico;

- Maior facilidade de programação.

A plataforma WFP pode ser utilizada na construção de qualquer tipo de aplicação que utilize a filtragem ou a modificação de mensagens de rede. Alguns exemplos são ilustrados a seguir.

- Antivírus: softwares que detectam a presença de softwares maliciosos. Algumas das suas atividades maliciosas podem ser atividades de rede.
- IDS: sistemas de detecção de intrusão. Detectam tentativas de acessos não autorizados.
- IPS: sistemas de prevenção de intrusão. Utilizam mecanismos para evitar acessos não autorizados.
- Firewall: software utilizado para permitir ou proibir a passagem de mensagens de acordo com uma política.
- Ferramentas de diagnóstico de rede: ferramentas de análise da qualidade e desempenhos de rede com monitoramento ativo ou passivo.
- Ferramentas de monitoramento e controle de conteúdo: ferramentas de controle de pais e outras que monitoram o conteúdo que trafega pela rede.
- Scanners: ferramentas utilizadas para a descoberta de serviços ativos em determinada máquina.
- Sniffers: ferramentas de análise do tráfego de mensagens que transita sob determinada interface de rede.

O funcionamento da plataforma Windows Filtering Platform pode ser exemplificado através de seis passsos que são descritos a seguir.

1. Um pacote ingressa na pilha de rede.
2. A pilha de rede chama um shim.
3. O shim invoca o processo de classificação na camada correspondente.
4. Durante a classificação, filtros são aplicados e suas ações correspondentes são tomadas.
5. Caso algum filtro de chamada se aplique, o driver de chamada correspondente é invocado.
6. O shim executa a ação final sobre o pacote.

2.7 Firmament

FIRMAMENT (DREBES, 2005) é uma ferramenta de injeção de falhas de comunicação. Foi construída para ambientes GNU/LINUX, na forma de um módulo para esses ambientes com auxílio do NETFILTER. A injeção de falhas é realizada na camada de rede, não permitindo portanto, alteração na camada de enlace. Possui suporte aos protocolos IPv4 e IPv6.

O código do FIRMAMENT é aberto e está disponível publicamente para que qualquer indivíduo possa obtê-lo e modificá-lo conforme sua necessidade. Foi construído com a linguagem de programação C e está modularizado para permitir a extensão de suas funcionalidades.

Os faultlets, descrição do comportamento das falhas, são construídos na forma de código de baixo nível, semelhante ao Assembly, denominada FIRMASM. A linguagem é composta de 31 instruções classificadas em sete grupos e recebe, cada uma delas, até três parâmetros. Além disso, é possível utilizar 16 registradores de 32 bits na construção dos faultlets. Os diferentes conjuntos de instruções da linguagem FIRMASM são apresentados abaixo.

Instrução	Argumento 1	Argumento 2	Argumento 3	Descrição
READB	Ry	Rx	-	Leitura de byte
READS	Ry	Rx	-	Leitura de dois bytes
READW	Ry	Rx	-	Leitura de quatro bytes
WRTB	Ry	Rx	-	Escrita de byte
WRTES	Ry	Rx	-	Escrita de dois bytes
WRTEW	Ry	Rx	-	Escrita de quatro bytes
SET	y	Rx	-	Escrita em registrador

Tabela 2.2: Instruções de entrada e saída

Instrução	Argumento 1	Argumento 2	Argumento 3	Descrição
ADD	Ry	Rx	-	Soma
SUB	Ry	Rx	-	Subtração
MUL	Ry	Rx	-	Multiplicação
DIV	Ry	Rx	-	Divisão
AND	Ry	Rx	-	E lógico
OR	Ry	Rx	-	OU lógico
NOT	Ry	Rx	-	NÃO lógico

Tabela 2.3: Instruções lógicas e aritméticas

Instrução	Argumento 1	Argumento 2	Argumento 3	Descrição
ACP	-	-	-	Aceitação da mensagem
DRP	-	-	-	Descarte da mensagem
DUP	-	-	-	Duplicação da mensagem
DLY	Rx	-	-	Atraso da mensagem

Tabela 2.4: Instruções de ação sobre as mensagens

Instrução	Argumento 1	Argumento 2	Argumento 3	Descrição
JMP	x	-	-	Desvio incondicional
JMPZ	Ry	x	-	Desvio condicional
JMPN	Ry	x	-	Desvio condicional

Tabela 2.5: Instruções de desvio de fluxo

Instrução	Argumento 1	Argumento 2	Argumento 3	Descrição
AION	Ry	Rx	-	Ativação do auto-incremento
AIOFF	Ry	-	-	Desativação do auto-incremento

Tabela 2.6: Instruções de manipulação de auto-incremento

Instrução	Argumento 1	Argumento 2	Argumento 3	Descrição
CSRT	Ry	Rx	"cadeia"	Comparação de cadeia
SSTR	Ry	"cadeia"	-	Escrita de cadeia

Tabela 2.7: Instruções de ação sobre as mensagens

Instrução	Argumento 1	Argumento 2	Argumento 3	Descrição
MOV	Ry	Rx	-	Cópia
RND	Ry	Rx	-	Núm. pseudo-aleatório
SEED	Ry	Rx	Rz	Semente para núm. pseudo-aleatório
DBG	Rx	"cadeia"	-	Depuração
DMP	Rx	-	-	Exibição de mensagem
VER	Rx	-	-	Versão da Máquina Virtual

Tabela 2.8: Instruções diversas

FIRMAMENT possui um mecanismo cão-de-guarda para evitar o processamento indefinido de um faultlet, o que pode deixar a ferramenta, a aplicação testada e o próprio sistema operacional irresponsíveis. Tal mecanismo é baseado em um tempo pré-definido de 20 milissegundos para o processamento de um faultlet. Uma vez esgotado o tempo definido, o processamento do faultlet é parado e a mensagem prossegue o seu curso, mesmo que o processamento parcial a tenha deixado inconsistente sob o ponto de vista do faultlet.

O tempo limite para o processamento de um faultlet pode ser modificado por meio de uma diretiva *setttimeout*. Além disso, o mecanismo cão-de-guarda pode ser desabilitado.

3 TRABALHOS RELACIONADOS

A injeção de falhas em comunicação é explorada, na maioria das vezes, através de trabalhos específicos para cada ambiente e para cada protocolo. Em decorrência disso, existem poucas ferramentas genéricas para a injeção de falhas de comunicação que atendam diferentes protocolos. Abaixo são apresentados alguns trabalhos relacionados à injeção de falhas de comunicação.

3.1 Injeção de Falhas em Aplicações Java

A injeção de falhas de comunicação é importante para assegurar o correto funcionamento de aplicações de rede. Para isso, pode-se utilizar o Comform (COMmunication Fault injector ORiented to Multi-protocol Java applications), um injetor de falhas desenvolvido para ser utilizado em aplicações Java que utilizem um ou mais protocolos de comunicação, como por exemplo, o TCP, UDP e o RMI (MENEGOTTO; WEBER, 2011). Na literatura, encontramos como exemplos de aplicações multi-protocolos e tolerantes a falhas a Zorilla e algumas aplicações do grupo de comunicação middleware JGroups. O injetor proposto, Comform, opera no nível da JVM (Java Virtual Machine), interceptando mensagens do protocolo e também no nível de sistema, usando regras do firewall para emular alguns tipos de falhas, como o descarte de mensagens, que não pode ser emulado no nível da JVM. O injetor também é útil para realizar testes de caixa branca e preta e possui características importantes como preservar o código-fonte da aplicação alvo.

3.2 Injeção de Falhas em Sistemas Orientados a Serviços

A tecnologia dos Web Services é de grande importância para aplicações e tecnologias orientadas a serviços e, por isso, é necessário avaliar o seu desempenho e Mecanismos de Tolerância a Falhas (FTMs) (FARJ; CHEN; SPEIRS, 2012). Para isso, utiliza-se uma ferramenta de injeção de falhas, o Network Fault Injector Service (NetFIS), a qual permite testar tanto o desempenho quanto a tolerância a falhas de Web Services.

A ferramenta pode emular uma WAN dentro de um ambiente de LAN entre componentes de serviços, oferecendo controle total sobre o ambiente emulado e a habilidade de injetar falhas relacionadas a redes sem a modificação desses componentes de serviço. Além disso, a ferramenta também pode gerar cargas de trabalho em segundo plano, o que permite emular de forma mais precisa as redes reais.

O injetor age como um Proxy de injeção de falhas e um emulador entre o serviço cliente e o serviço provedor, manipulando a invocação e a resposta de mensagens para emular comportamentos incorretos ou defeitos na rede e em serviços. A ferramenta pode

gerar as falhas de descarte e atraso de mensagens e, aleatoriamente, corromper mensagens SOAP. O fato de o injetor ser capaz de descartar uma mensagem SOAP completa provoca a propagação da falhas até o nível da aplicação, permitindo assim a verificação se aplicação pode tolerar tais falhas.

3.3 Injeção de Falhas de Comunicação em Dispositivos Móveis

O crescimento do uso de dispositivo móveis tem levado a criação de diversas aplicações nessas plataformas. Devido ao seu uso massivo, faz-se necessário validar as suas aplicações na presença de falhas de comunicação, uma vez que, nesses dispositivos, a maior parte das aplicações depende das redes de dados. Dispositivos móveis tem diferenciais quanto ao processamento e armazenamento e, por isso, prefere-se utilizar dados tratados e armazenados em equipamentos de maior porte para facilitar o uso de aplicações nos dispositivos móveis.

A ferramenta de injeção de falhas de comunicação FIRMAMENT (DREBES, 2005) foi portada para ambientes Android (ACKER, 2009), (ACKER; CECHIN; WEBER, 2010) (DOBLER, 2010), um sistema operacional para dispositivos móveis, cujo kernel é baseado no kernel Linux. Através de testes de injeção de falhas, é possível verificar o comportamento das aplicações na presença delas e definir ajustes que podem ser realizados para que suportem tais falhas.

4 DESENVOLVIMENTO

Neste capítulo, são apresentados os procedimentos realizados na construção da ferramenta WINFIRMAMENT. Além disso, são mostradas as ferramentas utilizadas durante o desenvolvimento, bem como ferramentas auxiliares de teste e depuração.

4.1 Compilador

O compilador do FIRMAMENT foi adaptado aos ambientes Windows com pequenas alterações na compilação e com a substituição de algumas bibliotecas, uma vez que algumas delas não estão presentes nos ambientes Windows. Seu funcionamento e sua operação são os mesmos de seu ambiente nativo (GNU/LINUX).

4.2 Faultlets

Os faultlets, descrição das falhas injetadas, após compilados, são carregados pelo WINFIRMAMENT na forma de registro do tipo binário (REG_BINARY). Uma vez carregados, podemos visualizá-los através do editor de registros do Windows no seguinte caminho: HKEY_LOCAL_MACHINE -> SYSTEM -> CurrentControlSet -> services -> WINFIRMAMENT.

4.3 Interface Gráfica

Para facilitar a carga e execução dos faultlets, foi criada uma interface gráfica simples. Nela é possível carregar um faultlet no seu formato textual, semelhante a um código Assembly, compilar um faultlet, gerar o código binário de um faultlet, iniciar e parar a injeção de falhas a partir do faultlet compilado.

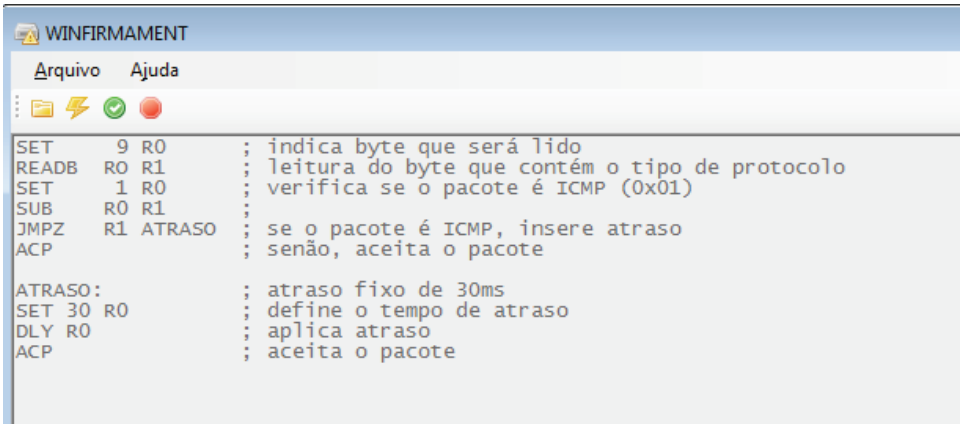

Figura 4.1: Interface Gráfica do WINFIRMAMENT.

4.4 Limitações

O WINFIRMAMENT possui algumas limitações quando comparado ao FIRMAMENT. Tais limitações devem-se à maneira como o WINFIRMAMENT foi construído. No entanto, as capacidades não implementadas nesta versão inicial devem ser adicionadas em uma próxima versão para permitir a total compatibilidade dos faultlets gerados bem como padronizar a operação da ferramenta nos diferentes ambientes. Aquelas instruções não suportadas são ignoradas pelo WINFIRMAMENT a fim de que não seja gerado um faultlet diferente daquele utilizado no FIRMAMENT.

O mecanismo cão-de-guarda não está presente na primeira versão do WINFIRMAMENT. Por este motivo, devem-se construir códigos de faultlets livres de laços infinitos para que o sistema não se torne irresponsível.

Não é possível, também nessa versão inicial, enviar os comandos suportado pelo FIRMAMENT. São eles: *startflow*, *stopflow*, *showregister*, *settimeout*, *reset*, *version* e *wdverbose*. No entanto, as capacidades de início e fim do processamento dos faultlets, bem como as suas cargas, são dados pela interface do WINFIRMAMENT, assim como a sua versão é exibida na própria interface.

4.5 Uso da Ferramenta

O WINFIRMAMENT tem um uso bastante simples e envolve cinco passos. São eles:

1. Construção do faultlet
2. Compilação do faultlet
3. Carga do faultlet
4. Inicialização do WINFIRMAMENT
5. Parada do WINFIRMAMENT

Com exceção da construção do faultlet, a qual pode ser realizada em um editor de texto qualquer, os passos seguintes podem ser realizados diretamente na interface do WINFIRMAMENT. Além disso, o faultlet utilizado é mostrado na sua interface no formato no qual foi construído e não aquele compilado, ou binário.

5 TESTES E RESULTADOS

Neste capítulo, são apresentados três testes para a validação do WINFIRMAMENT bem como são analisados os seus resultados.

5.1 Teste 1 - Atraso de Mensagens

Este teste consiste em inserir um atraso fixo de 30 milissegundos nos pacotes do tipo ICMP. Para a validação do faultlet, é utilizada a ferramenta PING, presente na maior parte dos sistemas operacionais. Na sua implementação presente no sistemas operacionais Windows 7, o seu funcionamento consiste em enviar uma mensagem ICMP do tipo 8 (echo request) e receber uma resposta do tipo 0 (echo reply). Normalmente, é utilizada para medir a latência entre dois dispositivos quaisquer, apesar de nem sempre fornecer resultados confiáveis, uma vez que os pacotes do tipo ICMP podem possuir menor prioridade frente a outros protocolos, tais como UDP e TCP.

Primeiro é verificada a latência entre os dispositivos sem a inserção do atraso e, após, com a inserção do atraso fixo de 30 milissegundos. Foram disparadas 30 requisições para cada um dos casos. Os resultados desse teste podem ser visualizados nas figuras a seguir e os seus faultlets encontram-se no apêndice A.

```
Resposta de 192.168.0.150: bytes=32 tempo<1ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo<1ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo<1ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo<1ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo<1ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo<1ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo<1ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo<1ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo<1ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo<1ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo<1ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo<1ms TTL=63

Estatísticas do Ping para 192.168.0.150:
    Pacotes: Enviados = 30, Recebidos = 30, Perdidos = 0 (0% de
             perda),
Aproximar um número redondo de vezes em milissegundos:
    Mínimo = 0ms, Máximo = 0ms, Média = 0ms
```

Figura 5.1: Execução da ferramenta PING sem a inserção de atrasos.

Além disso, testou-se o atraso para tempos maiores (acima de cinco segundos), o que ocasionou a indisponibilidade do sistema, levando-o, às vezes, à reinicialização e outras vezes ao travamento. Esses mesmos efeitos foram observados quando disparadas múltiplas instâncias do PING para aquele destino alvo do faultlet. Tais problemas devem-se ao fato de haver grande restrição de memória para um driver que executa em modo

```
Resposta de 192.168.0.150: bytes=32 tempo=32ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo=33ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo=32ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo=33ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo=31ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo=32ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo=33ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo=33ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo=32ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo=34ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo=32ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo=35ms TTL=63
Resposta de 192.168.0.150: bytes=32 tempo=32ms TTL=63

Estatísticas do Ping para 192.168.0.150:
    Pacotes: Enviados = 30, Recebidos = 30, Perdidos = 0 (0% de
             perda),
Aproximar um número redondo de vezes em milissegundos:
    Mínimo = 29ms, Máximo = 35ms, Média = 32ms
```

Figura 5.2: Execução da ferramenta PING com a inserção de atrasos fixo de 30ms.

kernel, diferentemente de uma aplicação que roda em modo usuário e dispõe de uma área de memória muito superior.

5.2 Outros Testes

Outros testes foram conduzidos sobre os protocolos UDP e TCP. Entretanto, o comportamento esperado não foi alcançado. Nesses testes, a função do WFP que indica a classificação do tráfego foi executada uma única vez, diferentemente do que ocorreu com os testes sobre ICMP. Nestes casos, a função de classificação executava a cada pacote ICMP recebido. O comportamento dos testes sobre UDP e TCP podem indicar que, sobre esses protocolos, o WFP carrega uma única vez as suas regras de filtragem e aplica para todo tráfego que fizer parte daquela regra, demonstrando que, para eles, o WFP os trata como fluxos e não individualmente.

6 CONCLUSÃO

Neste trabalho, foram apresentados mecanismos de filtragem de mensagens para ambientes Windows e avaliadas as suas capacidades. A ferramenta FIRMAMENT foi adaptada aos ambientes Windows, sob o nome WINFIRMAMENT, a fim de criar uma ferramenta para estes ambientes, dada a falta de ferramentas com tais capacidades.

A realização da injeção de falhas é essencial para a correta validação de uma aplicação que utilize a troca de mensagens entre os dispositivos, principalmente quando é desconhecido o trajeto das mensagens. Algumas falhas de comunicação podem não ser observadas quando são realizados testes de homologação de uma aplicação, pois, para isso, deve-se alterar o seu funcionamento e, muitas vezes, não é possível obter os mesmos comportamentos decorrentes de falhas em outros dispositivos envolvidos na comunicação.

A adaptação da ferramenta FIRMAMENT permitiu que os mesmos faultlets possam ser gerados nas duas diferentes plataformas, GNU/Linux e Windows, e, com isso, possibilitar a realização de injeção de falhas em ambas as plataformas sem qualquer modificação naqueles.

A descrição das falhas no formato de código semelhante ao Assembly permite que falhas complexas possam ser criadas com o uso de condições e repetições. Apesar disso, a construção de faultlets, para uma maior facilidade daquele que utiliza a ferramenta, poderia ser feita através de regras de mais alto nível, semelhantemente àquelas utilizadas em Firewalls, roteadores e outros dispositivos. A facilidade de uso, bem como uma documentação completa são fatores decisivos na escolha de uma ferramenta e, por isso, espera-se, em trabalhos futuros, adicionar funcionalidade e interfaces que visem a criação rápida de faultlets e que eles sejam facilmente compreendidos sem a necessidade de documentação do seu código.

Durante o desenvolvimento, os testes demonstraram as capacidades oferecidas pela ferramenta WINFIRMAMENT, através da simulação de falhas comuns presentes na comunicação entre dispositivos. A falha na execução de alguuns dos testes é decorrente da falta de documentação que não indica o comportamento dos filtros sobre fluxos e datagramas. No entanto, isso não demonstra que a plataforma WFP não tenha capacidade de simular as falhas propostas, uma vez que, sobre ICMP, os testes foram conduzidos com sucesso.

Enfim, pôde-se, através deste trabalho, construir mecanismos adicionais para um melhor desenvolvimento de aplicações que dependem de troca de mensagens através de redes nos ambientes Windows.

6.1 Trabalhos Futuros

O funcionamento incorreto do WINFIRMAMENT para os protocolos UDP e TCP indica que o tratamento deles deve ser diferenciado pelo WFP e, por isso, devem-se averiguar as causas para que a ferramenta funcione corretamente. Uma abordagem possível é deixar que uma aplicação externa ao driver realize tanto a classificação das mensagens como a alteração e as suas ações relacionadas. Essa mesma abordagem pode ser eficaz para manipular os atrasos, uma vez que uma aplicação no modo usuário não possui a mesma restrição de memória que um driver no modo kernel possui.

O carregamento dos faultlets pode ser modificado e, no lugar de registros, podem ser utilizados arquivos. Apesar de serem mais facilmente manipulados pelos usuários, os arquivos não apresentam tal facilidade quando manipulados por um driver no modo kernel.

Além disso, a descrição das falhas, através dos faultlets pode ser simplificada, de modo que aquele que a constrói bem como aquele que a verifica possam comprender tais descrições sem uma documentação adicional, necessária quando ela é complexa.

REFERÊNCIAS

NET APPLICATIONS. **Top Operating System Share Trend.** Disponível em: <http://www.netmarketshare.com/os-market-share.aspx?qprid=9>. Acesso em: abr. 2012.

CORBET, J.; RUBINI, A.; KROAH-HARTMAN, G. **Linux Device Drivers.** O'Reilly Media, 2005. 640p

MICROSOFT. **What is a driver?** Disponível em: <http://msdn.microsoft.com/en-us/library/windows/hardware/ff554678>. Acesso em: abr. 2012.

AVIZIENIS, A.; LAPRIE, J. C.; RANDELL, B. **Fundamental Concepts of Dependability.** 2000. In Proc. 3rd Information Survivability Workshop, 7–12.

HSUEH, M. C.; TSAI, T. K.; IYER, R. K. **Fault Injection Techniques and Tools.** Computer, v.30, n.4, p. 75-82, Abr. 1997.

MICROSOFT. **Windows Driver Kit (WDK).** Disponível em: <http://msdn.microsoft.com/en-us/library/windows/hardware/gg487428.aspx>. Acesso em: abr. 2012.

MICROSOFT. **Microsoft Visual Studio.** Disponível em: <http://www.microsoft.com/visualstudio/pt-br/home-produtos>. Acesso em: mar. 2012.

IPERF. **Iperf.** Disponível em: <http://iperf.sourceforge.net/>. Acesso em: fev. 2012.

OSR ONLINE. **Driver Loader.** Disponível em: <http://www.osronline.com/article.cfm?article=157>. Acesso em: fev. 2012.

MICROSOFT. **DebugView - Windows Sysinternals .** Disponível em: <http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx>. Acesso em: mar. 2012.

MICROSOFT. **Unraveling the Mysteries of Writing a Winsock 2 Layered Service Provider.** Disponível em: <http://www.microsoft.com/msj/0599/layeredservice/layeredservice.aspx>. Acesso em: abr. 2012.

MICROSOFT. **Filter-Hook Drivers.** Disponível em: <http://msdn.microsoft.com/en-us/library/windows/hardware/ff546489%28v=vs.85%29.aspx>. Acesso em: abr. 2012.

MICROSOFT. **Firewall-Hook Drivers.** Disponível em: <http://msdn.microsoft.com/en-us/library/windows/hardware/ff546499%28v=vs.85%29.aspx>. Acesso em: abr. 2012.

MICROSOFT. **Network Driver Interface Specification.** Disponível em: <http://technet.microsoft.com/en-us/library/cc958797.aspx>. Acesso em: mar. 2012.

ZWANGER, V.; **An Introduction To Writing TDI Filter Drivers.** Disponível em: <http://iseclab.org/papers/Writing_TDI_Drivers.pdf>. Acesso em: mar. 2012.

MICROSOFT. **Windows Filtering Platform.** Disponível em: <http://msdn.microsoft.com/en-us/library/windows/desktop/aa366510%28v=vs.85%29.aspx>. Acesso em: mar. 2012.

FARJ, K.; CHEN, Y.; SPEIRS, N. A. **A Fault Injection Method for Testing Dependable Web Service Systems.** 2012. In 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, 49-55.

MENEGOTTO, C. C.; WEBER, T. S. **Communication Fault Injection for Multi-Protocol Java Applications Testing.** 2011. In Test Workshop (LATW), 2011 12th Latin American, 1-6.

ACKER, E. V.; **Validação de Aplicações para Ambientes Móveis Utilizando Injeções de Falhas.** 2009. 43 f. Trabalho de Graduação (Bacharelado em Engenharia de Computação) – Instituto de Informática, UFRGS, Porto Alegre.

ACKER, E. V.; WEBER, T. S.; CECHIN, S. L. **Injeção de falhas para validar aplicações em ambientes móveis.** 2010. In Workshop de Testes e Tolerância a Falhas, 11:61-74.

DOBLER, R. J. **Injeções de Falhas de Comunicação para Validação de Aplicações no Ambiente Android.** 2010. 67 f. Trabalho de Graduação (Bacharelado em Engenharia de Computação) – Instituto de Informática, UFRGS, Porto Alegre.

DREBES, R. J. **FIRMAMENT: Um Módulo de Injeção de Falhas de Comunicação para Linux.** 2005. 87 f. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

ANEXO - ARTIGO DA PROPOSTA DESTE TRABALHO

Adaptação do Injetor de Falhas de Comunicação FIRMAMENT para Ambientes Windows

Fernando D. F. Macedo

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

fdfmacedo@inf.ufrgs.br

Abstract. *Computer networks can bring lots of faults that increase with their interconnection complexity [Kumar 1990]. This faults presence can cause system malfunctioning, such as cause services unavailability. Performing communication injection faults is important to the computer networks dependent applications and systems, because some faults aren't easily observable in a development environment. Using fault injectors can make applications more reliable through adjustments in their behavior to turn them tolerants to the common faults in computers networks. This paper approaches an adaptation of the communication fault injector FIRMAMENT to Windows environments.*

Resumo. *Redes de computadores podem apresentar diversos tipos de falhas que crescem com a complexidade de suas interligações [Kumar 1990]. A existência dessas falhas pode acarretar o mau funcionamento de sistemas, bem como causar indisponibilidade de serviços. Executar injeções de falhas de comunicação é importante para aplicações e sistemas que dependem de uma rede de computadores, pois algumas falhas não são facilmente observáveis em um ambiente de desenvolvimento. O uso de injetores de falha pode tornar aplicações mais confiáveis através de ajustes nos seus comportamentos para torná-las tolerantes às mais comuns falhas presentes nas redes de computadores. Este artigo aborda a adaptação do injetor de falhas de comunicação FIRMAMENT para ambientes Windows.*

1. Introdução

O ciclo de desenvolvimento de uma aplicação requer que ela seja validada em diversos aspectos. A validação implica a análise dos requisitos que a aplicação deve atender e se ela os atende corretamente. Frequentemente, testes de validação de uma aplicação não contemplam a injeção de falhas no ambiente no qual a aplicação executa.

A popularização dos computadores pessoais tem impulsionado a criação de novas aplicações para automatizar tarefas que antes eram executadas fora de um ambiente computacional ou, então, cujos procedimentos não envolviam a distribuição do seu conteúdo aos usuários finais. Esse cenário torna-se mais agudo ao perceber-se que, apesar de ser baixa, a taxa daqueles que têm acesso a um computador, no Brasil quinze porcento, aproximadamente, tem crescido de maneira acelerada.

Com a crescente demanda por aplicações que utilizam a Internet para a troca de mensagens, faz-se necessária a utilização de ferramentas que permitam verificar o comportamento de sistemas em um ambiente complexo e sujeito a diversos tipos de falhas.

Este trabalho propõe a adaptação da ferramenta de injeção de falhas de comunicação FIRMAMENT para ambientes Windows, incorporando as suas facilidades ao sistema operacional para permitir que aplicações sejam validadas na presença de falhas de comunicação sem a necessidade de alterações nos seus códigos fonte e sem a necessidade de qualquer dispositivo adicional.

2. Falhas de Comunicação

As falhas de comunicação podem se manifestar de três maneiras diferentes [Han 1995]: perda de mensagens, alteração de mensagens e atraso na recepção de mensagens. Além disso, podem ocorrer duplicações de mensagens [Dai 2005], principalmente em ambientes distribuídos, nos quais uma mensagem pode ser replicada em diversos nodos.

2.1. Perda de Mensagens

Perdas de mensagens são ocorrências do não recebimento de determinada mensagem por um equipamento receptor. Algumas de suas causas são: transbordamento de dados [Bolot 1993], descartes induzidos e falhas de enlances.

2.1.1. Transbordamento de Dados

O transbordamento de dados ocorre quando um dispositivo não possui recurso disponível para manipular um dado e o descarta. Isso ocorre porque os recursos em um dispositivo são finitos. Falhas desse tipo podem ser diminuídas com o aumento de recursos disponíveis, como o aumento de memória, mas esse aumento implica custo mais alto no preço final de um dispositivo e nem sempre é viável, dependendo daquilo que se projeta.

2.1.2. Descartes Induzidos

Descartes induzidos são aqueles que ocorrem em um equipamento prévia ou dinamicamente configurado para descartar mensagens de acordo com uma regra. Essas regras são chamadas filtros e podem estar presentes em equipamentos como firewalls e comutadores, entre outros. Em um ambiente com qualidade de serviço (QoS) configurada, por exemplo, temos os descartes de mensagens para garantir que uma regra de qualidade de serviço seja garantida.

2.1.3. Falhas de Enlaces

Ocorrem na falha de um meio físico pelo qual transitam os dados. Essas falhas podem ocorrer devido a interferências, interrupções ou defeitos físicos nos enlaces.

2.2. Alteração de Mensagens

Mensagens alteradas são aquelas que tiveram o seu cabeçalho ou o seu corpo modificados e são recebidas pelo receptores diferentemente daquelas enviadas. Esse tipo de falha pode ter qualquer uma das causas que provoca, também, o descarte de mensagens, conforme explicado anteriormente.

2.3. Atraso de Mensagens

A entrega de uma mensagem não ocorre instantaneamente, pois existem tempos de propagação das mensagens nos diferentes meios pelas quais passam e tempos de processamento nos dispositivos intermediários, bem como nos dispositivos finais. O atraso de uma mensagem decorre na entrega de uma mensagem com um tempo superior àquele esperado. O atraso pode ocorrer devido a fatores como a elevação no tempo de processamento em um dispositivo ou um tempo superior de propagação devido a um caminho maior tomado por uma mensagem.

2.4. Duplicação de Mensagens

Uma mensagem duplicada é aquela recebida por um dispositivo duas ou mais vezes. Esse tipo de evento pode ocorrer em um sistema distribuído no qual as mensagens são duplicadas para aumentar a probabilidade de que ela chegue ao seu destino, em sistemas com mecanismo de repetição por limite excedido no recebimento de uma mensagem - como o protocolo TCP, por exemplo. Também ocorre em comutadores que, ao não conhecerem a localização do destino ou intermediário, enviam a mensagem a todos os dispositivos diretamente conectados.

3. Injeção de Falhas em Redes de Computadores

A injeção de falhas pode ser realizada tanto em hardwares quanto em softwares [Hsueh 1997]. A injeção de falhas em hardware pode ser realizada com ou sem contato. A técnica que utiliza contato demanda a inserção de probes nos pinos do hardware ou a inserção de soquetes no hardware. A técnica que não envolve contato consiste em inserir radiação no ambiente de modo a provocar falhas no hardware. A injeção de falhas em software pode ser de dois tipos: em tempo de compilação ou em tempo de execução.

A injeção de falhas em tempo de execução possui diversas vantagens sobre as outras:

- Pode ser realizada sem a necessidade de hardware;
- Não requer alteração no código fonte da aplicação avaliada;
- Não possui risco de danos fìsicos, tal como podem ocorrer com probes;
- Possui alta controlabilidade;
- Possui alta repetitibilidade.

4. Outros Trabalhos

Diversos trabalhos foram propostos para a validação de sistemas que se utilizam da comunicação de dados [Han 1995], [Gupta 2003], [Dai 2005], [Drebes 2005], [Acker 2009] e [Dobler 2010]. Alguns deles propõem ferramentas para a validação completa do sistema com a injeção de falhas em diversos componentes dos sistemas, não se restringindo somente às falhas de comunicação. Abaixo são apresentados alguns dos trabalhos que usam a injeção de falhas para verificar o grau de cobertura dos mecanismos de tratamento de falhas em sistemas que utilizam troca de mensagens.

4.1. Injeção de Falhas em Ambientes Móveis

A ferramenta de injeção de falhas de comunicação FIRMAMENT foi adaptada para funcionamento em ambientes móveis com sistemas operacionais Android [Acker 2009] e [Dobler 2010]. Apesar de o Android possuir o seu núcleo baseado no núcleo Linux, foram necessárias algumas adaptações ao código do FIRMAMENT para o seu funcionamento no Android. Além disso, a arquitetura dos processadores de dispositivos móveis não é a mesma encontrada nos computadores pessoais, geralmente baseadas em x86.

A validação das aplicações em ambientes móveis tem aumentado rapidamente a sua importância, pois o número de celulares no mundo já era superior a cinquenta porcento do número de habitantes em 2007 [IBRE 2007]. No Brasil, o número de celulares em Junho de 2011 era superior ao número de habitantes, com uma densidade de 111 celulares para cada 100 habitantes. Um fator que tem acelerado esse crescimento é a queda dos preços para esse tipo de dispositivo. Essa queda também tem afetado os smartphones, celulares com funcionalidades avançadas que executam um sistema operacional específico para esse fim.

4.2. Tolerância a Falhas em Redes de Sensores sem Fio

A robustez de uma rede de sensores sem fio deve ser avaliada para o seu correto funcionamento [Gupta 2003]. Uma rede de sensores está limitada por diversos fatores que podem contribuir para a ocorrência de falhas no seu uso. O maior limitante de um sistema desse tipo é a baixa disponibilidade de energia dos dispositivos, pois, normalmente, estão inseridos em ambientes que não podem fornecer energia constantemente e são alimentados por baterias. Devido ao uso restrito de energia, os protocolos de comunicação em redes de sensores devem oferecer recuperação na presença de falhas sem o uso excessivo de energia. A validação de diferentes modelos de arquiteturas de sensores distribuídos deve ser executada antes da adoção de um modelo de acordo com o nível de confiabilidade que se deseja em uma rede de sensores.

4.3. Ambientes para Injeções de Falhas Diversas

Algumas ferramentas permitem a avaliação de um sistema de maneira mais completa, ao avaliar mais de um componente e permitir a construção de falhas mais complexas com a combinação de falhas e a diversidade de parâmetros de configuração das falhas.

A ferramenta DOCTOR [Han 1995] permite uma avaliação da dependabilidade de sistemas sob diferentes aspectos. Nele, podem ser simuladas falhas de memória, de processador e de comunicação. Suas opções de falhas para cada grupo de falhas são os seguintes:

- Memória: extensão, tipo, duração, intervalo e localização;
- Processador: tipo, duração, intervalo e localização;
- Comunicação: tipo, opções, duração e intervalo.

5. Windows Filtering Platform

A plataforma Windows Filtering Platform (WFP) é um conjunto de arcabouços e serviços de sistema que permitem a criação de aplicações de filtragem de rede. Com esse conjunto, podem-se criar ferramentas que analisem e modifiquem dados da rede nas suas camadas de rede e superiores.

O uso do Windows Filtering Platform se dá através da criação de drivers, escritos nas linguagens C ou C++. Os drivers podem ser desenvolvidos com o Windows Driver Kit (WDK), um conjunto de ferramentas para a construção de drivers quaisquer para ambientes Windows. Essa limitação na construção de drivers não impede que sejam utilizadas outras linguagens para uma aplicação de rede, pois elas podem fazer chamadas aos drivers através de bibliotecas próprias.

5.1. Histórico

O Windows Filtering Platform substitui antigos mecanismos de filtragem de pacotes, tais como os filtros TDI (interface para drivers de transporte), filtros NDIS (especificações de interface para drivers de rede) e filtros Winsock LSP (provedores de serviços em camadas).

Essa plataforma está disponível em versões do Windows 2008 Server, Windows Vista e superiores. Não há suporte ao Windows Filtering Platform para as versões anteriores de Windows, uma vez que esses produtos foram descontinuados.

5.2. Aplicações

O Windows Filtering Platform pode ser utilizado para qualquer aplicação que faça inspeção, filtragem e alteração de pacotes. Alguns exemplos de aplicações que podem ser construídas com o Windows Filtering Platform:

- Antivírus;
- Controle de Conteúdo de Pais (Parental Control);
- Firewalls;
- Monitores de Rede;
- Sistema de Detecção de Intrusão (IDS);
- Sistema de Prevenção de Intrusão (IPS);
- Scanners;
- Sniffers.

5.3. Arquitetura WFP

A arquitetura do Windows Filtering Platform é composta por três elementos: o motor de filtro, o motor de filtragem básico e os drivers de chamada.

O motor de filtro possui dois componentes, um para o modo kernel e o outro para o modo usuário, os quais são responsáveis por todas as ações relacionadas à filtragem de dados na rede. O componente de modo kernel é responsável por filtragens nas camadas de rede e de transporte do modelo TCP/IP, já o componente de modo usuário é responsável por filtragens de RPC (Chamada Remota de Procedimentos) e IPsec (protocolo de segurança IP). Cada um dos componentes possui as suas camadas de filtragem.

O motor de filtragem básico (BFE) é um serviço que coordena os componentes do Windows Filtering Platform. Sua principal função é adicionar e remover filtros do sistema, armazenar configurações e reforçar configurações de segurança do Windows Filtering Platform. As aplicações devem se comunicar com o BFE através das funções de gerenciamento do Windows Filtering Platform.

Os drivers de chamada estendem as capacidades de filtragem através de funções personalizadas em uma ou mais camadas de filtragem do modo kernel. Eles permitem tanto a inspeção quanto a modificação de mensagens.

5.4. Camadas

Cada camada define um conjunto de filtros que o motor de filtro pode aplicar. Uma camada pode ser composta de sub-camadas. Exemplos de camadas: camada de pacotes IPv4 entrantes e camada de quadros Ethernet.

5.5. Filtros

Um filtro consiste em uma regra com uma ação associada. Ele pode ser aplicado em qualquer sentido (entrada, saída ou ambas as direções). Exemplo de filtro: bloqueio de conexões saintes para a porta 53/UDP (DNS).

5.6. Shims

Shims são componentes do modo kernel responsáveis por executar as ações determinadas pelos filtros.

5.7. Ações sobre os Dados

Uma vez classificado um conjunto de dados, que pode ser um pacote, um fluxo ou um evento, o Windows Filtering Platform pode tomar uma das seguinte ações sobre os dados:

- Permitir;
- Bloquear;
- Encaminhar a um driver de chamada:
  - Permitir;
  - Bloquear;
  - Continuar;
  - Atrasar;
  - Aguardar mais dados;
  - Encerrar a conexão.

5.8. Operações do Windows Filtering Platform

O funcionamento do Windows Filtering Platform pode ser exemplificado em seis passos, conforme a seguir:

1. Um pacote ingressa na pilha de rede.
2. A pilha de rede chama um shim.
3. O shim invoca o processo de classificação na camada correspondente.
4. Durante a classificação, filtros são aplicados e suas ações correspondentes são tomadas.
5. Caso algum filtro de chamada se aplique, o driver de chamada correspondente é invocado.
6. O shim executa a ação final sobre o pacote.

6. FIRMAMENT

FIRMAMENT [Drebes 2005] é uma ferramenta de injeção de falhas de comunicação para ambientes GNU/Linux. A ferramenta foi construída como um módulo do sistema operacional para permitir a captura, análise e alteração de pacotes TCP/IP. Suas funcionalidades foram construídas a partir do NETFILTER, um arcabouço para filtragem de pacotes disponível no kernel Linux a partir da sua série 2.4.X. São quatro as possíveis ações tomadas sobre um pacote pelo FIRMAMENT: inserção de atraso, descarte, duplicação e modificação do seu conteúdo.

As falhas injetadas são configuradas através de faultlets, que são descrições de falhas, como elas devem ocorrer e em quais situações. Esses faultlets são descritos em uma linguagem de baixo nível denominada FIRMASM e executados pela máquina virtual FIRMVM. A linguagem FIRMASM é composta por 31 instruções, contendo manipulação dos 16 registradores de 32 bits, instruções de desvio, instruções de ação sobre pacotes, manipulação sobre sequências de caracteres, entre outros. Na Tabela 1, são mostradas as instruções suportadas pelo FIRMASM relativas às ações sobre os pacotes.

Instrução	Argumento 1	Argumento 2	Argumento 3	Descrição
ACP	-	-	-	Aceitação de pacote
DRP	-	-	-	Descarte de pacote
DUP	-	-	-	Duplicação de pacote
DLY	Rx	-	-	Atraso de pacote

Tabela 1. Instruções do FIRMASM para ações sobre pacotes

Devido a possibilidade de usos de instruções de desvio, FIRMAMENT permite construir faultlets com laços infinitos. Como não é possível detectar a existência de laços infinitos para qualquer faultlet, FIRMAMENT possui um mecanismo cão-de-guarda, construído na forma de um temporizador. Na ocorrência de um faultlet que excede o tempo determinado pelo mecanismo cão-de-guarda, a execução de tal faultlet é imediatamente abortada a fim de evitar o comprometimento do sistema operacional em que o FIRMAMENT executa. Esse tipo de abordagem, apesar de evitar a ocorrência de laços infinitos, pode abortar a execução de um faultlet livre de laço infinito, isso porque não se pode determinar o tempo de execução de uma determinada tarefa em um sistema que não seja de tempo real. Logo, a construção de códigos livres de laços infinitos é de responsabilidade daquele que cria o faultlet para o FIRMAMENT.

Devido ao seu código aberto e a sua arquitetura modular, FIRMAMENT pode ser facilmente estendido no seu conjunto de protocolos suportados, nas suas instruções de máquina virtual FIRMVM e nos seus comando de controle e operação do sistema. Para isso, deve-se fazer pequenas alterações no seu código.

7. Especificação do Projeto

O projeto de adaptação do FIRMAMENT para ambientes Windows consiste em desenvolver ferramentas de modo que seja possível executar os mesmos faultlets desenvolvidos na ferramenta FIRMAMENT para outras plataformas nos ambientes Windows e obter os mesmos comportamentos para os faultlets.

São objetivos do projeto:

- Adaptar o compilador FIRMASM aos ambientes Windows;
- Criar uma versão do FIRMAMENT para ambientes Windows;
- Avaliar o impacto do Windows Filtering Platform;
- Validar as ferramentas desenvolvidas.

8. Cronograma de Atividades

Na Figura 1, é apresentado um diagrama de Gantt do plano de atividades para o Trabalho de Graduação 2 (TG2) a ser desenvolvido no primeiro semestre de 2012. Os períodos apresentados são aproximados e podem sofrer pequenas variações durante o desenvolvimento do trabalho. O trabalho está dividido em cinco etapas:

1. Funcionalidades Básicas: desenvolvimento das funcionalidades básicas, ou seja, as quatro ações possíveis sobre os pacotes (atraso, duplicação, descarte, alteração do seu conteúdo).
2. Compilador: adaptação do compilador FIRMASM aos ambientes Windows.
3. Outras funcionalidades: interpretação do restante das instruções não contempladas na primeira etapa.
4. Testes: geração de faultlets, testes de desempenho e análise dos resultados obtidos.
5. Escrita do TG2: elaboração da monografia do Trabalho de Graduação 2.

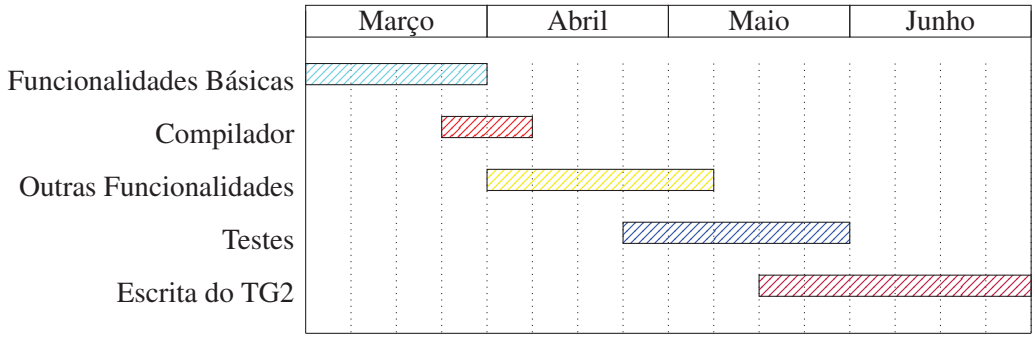

Figura 1. Cronograma de atividades para o Trabalho de Graduação 2 (TG2)

9. Conclusão

O uso de ferramentas para injeção de falhas de comunicação de dados são importantes na validação de aplicações que trocam mensagens. Isso porque as redes de computadores podem apresentar diversas falhas não encontradas em um ambiente de testes comum.

O ambiente do Windows Filtering Platform, com a sua variedade de filtros, pode permitir a construção de injetores de falhas nos ambientes Windows tal como o NETFILTER nos ambientes GNU/Linux.

Referências

Kumar, A.; Agrawal, D. P. (1990). Computer Network Reliability Evaluation from Application's Point of View.

Han, S; Rosenberg, H. A.; Shin, K.G. (1995) DOCTOR: An Integrated Software Fault Injection Environment.

Dai, S.; Jing, X.; Li, L.(2005) Research and Analysis on Routing Protocols for Wireless Sensor Networks.

Bolot, J.; Cedex, S. (1993) End-to-End Packet Delay and Loss Behavior in the Internet.

Hsueh, Mei-Chen; Tsai, T. K.; Iyer, R. K. (1997) Fault Injection Techniques and Tools.

Acker, E. V. (2009) Validação de Aplicações para Ambientes Móveis Utilizando Injeções de Falhas.

Dobler, R. J. (2010) Injeções de Falhas de Comunicação para Validação de Aplicações no Ambiente Android.

Insituto Brasileiro de Economia (2007) O Valor da Telefonia Móvel para a Sociedade Brasileira. Disponível em: www.telebrasil.org.br/RELATORIO%20FINAL%20FGV.pdf. Acessado em: Outubro, 2011.

Gupta, G.; Younis, M. (2003) Fault-Tolerant Clustering of Wireless Sensor Networks.

Drebes, R. J.(2005) FIRMAMENT: Um Módulo de Injeção de Falhas de Comunicação para Linux.

Microsoft Corporation (2011) Windows Filtering Platform. Disponível em: http://msdn.microsoft.com/en-us/library/windows/desktop/aa366510.aspx. Acessado em: Outubro, 2011.

Microsoft Corporation (2011) WFP Architecture. Disponível em: http://msdn.microsoft.com/en-us/library/windows/desktop/aa366509.aspx. Acessado em: Outubro, 2011.

APÊNDICE - FAULTLETS EMPREGADOS NOS TESTES

Neste apêndice, são mostrados os códigos dos faultlets utilizados nos testes de validação da ferramenta WINFIRMAMENT.

```
SET     9 R0       ; indica byte que será lido
READB  RO R1       ; leitura do tipo de protocolo
SET     1 R0       ; verifica se o pacote é ICMP (0x01)
SUB    R0 R1       ;
JMPZ   R1 ATRASO   ; se o pacote é ICMP, insere atraso
ACP                ; senão, aceita o pacote

ATRASO:            ; atraso fixo de 30ms
SET 30 R0          ; define o tempo de atraso
DLY R0             ; aplica atraso
ACP                ; aceita o pacote
```

Figura 6.1: Teste 1 - Atraso de Mensagens