Porte do Injetor de Falhas Firmament para o Ambiente Android

Rodrigo J. Dobler[1], Taisy S. Weber[1], Sérgio L. Cechin[1]

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

{rjdobler,taisy,cechin}@inf.ufrgs.br

Resumo. *Dispositivos móveis como celulares, palms e smartphones estão cada vez mais presentes em nossas vidas. Eles estão evoluindo muito depressa e, a cada nova versão, os aparelhos são lançados com muito mais recursos. Isto proporciona novos horizontes para os desenvolvedores de software. Hoje, devido a iniciativas de alguns fabricantes, muitas empresas e desenvolvedores independentes estão lançando programas para celulares. Porém, nem sempre são tomados cuidados com relação à tolerância a falhas e, desse modo, muitas aplicações podem apresentar problemas. Assim, nesse trabalho será portada uma ferramenta de injeção de falhas, Firmament, a qual foi desenvolvida para o sistema operacional Linux, para que ela possa ser utilizada no ambiente Android. Esta ferramenta irá permitir injeção de falhas na troca de mensagem sobre o protocolo IP de algumas aplicações rodando no emulador do Android. Isto irá permitir que se possa analisar o comportamento dessas aplicações na presença de falhas e assim, verificar se são ou não tolerantes a falhas.*

1. Introdução

Os dispositivos móveis trouxeram muitos benefícios para as nossas vidas. Os aparelhos celulares, por exemplo, permitem que nós possamos nos comunicar com outras pessoas ou acessar informação de quase qualquer lugar em que estejamos. Estes aparelhos melhoraram muitos nos últimos anos, o que possibilitou a criação de excelentes sistemas operacionais para eles.

Alguns fabricantes, como a Apple e a Google, lançaram ferramentas de desenvolvimento de software, para permitir que outras empresas e desenvolvedores independentes possam desenvolver aplicativos para as respectivas plataformas. Porém, apenas a Google tornou público o código fonte do Android e não impôs restrições ao desenvolvimento de aplicações. Essa estratégia da Google incentivou o uso do Android nos celulares e hoje um grande número de fabricantes o adota como sistema operacional de seus aparelhos, como Motorola, Samsung, Sony Ericsson, LG e outros. Assim, o Android começou a ganhar grande destaque no mercado de aparelhos móveis.

De acordo com a recente pesquisa publicada pela empresa Canalys (CANALYS), nos Estados Unidos, o maior mercado de smartphones do mundo, os dispositivos com Android representaram juntos 34% do mercado norte-americano no 2º trimestre de 2010. No mundo todo, nesse mesmo período, o número de smartphones que utilizam o Android cresceu 886% e isso se deveu as operadoras de telefonia celular, as quais estão promovendo amplamente os dispositivos Android e também ao grande

número de aplicações feitas para este sistema, sendo muitas delas de excelente qualidade e sem custo algum para o usuário. Esse grande número de aplicações desenvolvidas é realmente um grande atrativo, contudo, nem todas elas possuem implementações consistentes para detectar e corrigir erros e, assim, quando falhas acontecem, elas podem causar erros na aplicação, causando transtornos para o usuário.

Desse modo, seria necessário testar as aplicações desenvolvidas, injetando-se falhas, para que seja possível analisar como as aplicações irão se comportar. Isto permitiria fazer uma prevenção nas aplicações para que elas fossem capazes de contornar um estado de falha sem causar maiores problemas ao usuário ou sistema operacional. Para que isso seja viável, é necessário que as aplicações tenham alta dependabilidade.

Avizienis (Avizienis 2004) define dependabilidade, como sendo a habilidade de prestar serviços em que se pode justificadamente confiar. Este conceito engloba uma série de atributos como disponibilidade, confiabilidade, facilidade de manutenção, segurança funcional crítica, testabilidade, entre outros.

Normalmente, utilizam-se ferramentas de injeção de falhas para se poder testar um sistema eficientemente e, assim, encontrar soluções para os eventuais problemas, garantindo assim um aumento na dependabilidade do sistema. Hoje não existem muitas ferramentas de injeção de falhas e, destas, a maioria não utiliza métodos eficientes para realizar bons testes e assim, pode-se pensar que o aparelho possui um sistema de comunicação confiável quando na verdade pode não ter.

Falhas de comunicação são relevantes em ambientes móveis pois eles dependem do envio e recebimento de mensagens para poderem se comunicar. Assim, o uso de uma boa ferramenta de injeção de falhas torna-se essencial para que seja possível fazer testes eficientes e assim obter conclusões corretas.

Assim, este trabalho irá mostrar um passo a passo de como se fazer o porte para o ambiente Android de uma ferramenta de injeção de falhas, o Firmament, a qual foi desenvolvida para o sistema operacional Linux. A idéia é que esse trabalho possa ser utilizado como referência para outras pessoas que tenham interesse em portar ferramentas de injeção de falhas, ou de outros tipos, do Linux para o Android.

2. Especificação do Projeto

2.1. Netfilter

O Netfilter (RUSSEL; WELTE, 2002) é uma ferramenta para manipulação de pacotes que define pontos específicos, ou ganchos, na pilha de protocolos onde funções de callback podem ser registradas. Quando os pacotes alcançam esses pontos, eles são passados para essas funções, as quais possuem acesso completo ao seu conteúdo. Assim, elas podem processar os pacotes e decidir se os mesmos podem passar pela pilha de protocolos ou serem descartados.

O Netfilter permite o registro de funções de callback para as famílias de protocolos IPv4, IPv6, DECnet e ARP. Os ganchos estão disponíveis em diversos pontos de interesse: no recebimento e envio de pacotes, no encaminhamento de pacotes roteados e na entrega e recebimento de pacotes para os níveis superiores.

2.2. Firmament

É uma ferramenta de injeção de falhas, desenvolvida por Roberto Jung Drebes (DREBES,2005), (SIQUEIRA, 2009), a qual visa permitir a especificação de cenários de falhas de comunicação complexos, com baixa intrusividade, para o teste de protocolos de comunicação e sistemas distribuídos.

Esta ferramenta pode ser utilizada para aplicações de rede, nas quais o uso de implementações reais pode evidenciar deficiências da implementação que possivelmente escapariam à simulação. Ela também pode ser usada para o estudo de novos protocolos, pois a ferramenta permite colocar o sistema em estados incomuns ou de difícil reprodução.

O Firmament utiliza a arquitetura da interface de programação Netfilter, a qual está disponível a partir da versão 2.4 do kernel do Linux. Por utilizar apenas interfaces de programação de alto nível do kernel, a ferramenta é independente da arquitetura da máquina e, assim, pode ser utilizada em servidores, desktops e dispositivos embarcados que utilizem o Linux.

2.3. Android

O Android (ANDROID) é um ambiente de desenvolvimento que foi inicialmente desenvolvido pela Android Inc, empresa que foi adquirida pela Google, e depois foi lançado oficialmente pela Open Handset Alliance, um grupo formado por 65 empresas de tecnologia que atuam em diferentes áreas como operadoras de telefonia, empresas de software, empresas de semicondutores, fabricantes de celulares, etc.

O sistema operacional do Android foi construído com base na versão 2.6 do kernel do Linux e a utiliza para os serviços centrais do sistema, tais como segurança, gestão de memória, gestão de processos, etc. O kernel também atua como uma camada de abstração entre o hardware e o software.

O Software Development Kit, SDK, inclui um conjunto abrangente de ferramentas de desenvolvimento. Estas incluem um debugger, bibliotecas, um emulador de telefone (com base no QEMU), documentação do código da amostra, e tutoriais. Atualmente as plataformas suportadas para desenvolvimento incluem computadores de arquitetura x86 rodando Linux , Mac OS X 10.4.8 ou posterior, Windows XP ou Vista.

3. Desenvolvimento

3.1. Obtenção dos arquivos fontes do Android

Para se obter os arquivos fonte e depois compilá-los, precisa-se seguir os passos do tutorial mostrado no próprio site do Android. O tutorial encontra-se na página (ANDROID-SOURCE).

3.2. Recompilando o kernel do Android

O kernel do Android necessita ser recompilado para que se possa ativar o módulo do Netfilter. A ativação deste módulo torna-se necessária para que o Firmament, ferramenta de injeção de falhas, possa ser utilizado para atuar sobre as mensagens na rede. Também será necessário ativar o carregamento de módulos, para que o módulo recompilado do Firmament possa ser executado no emulador do Android.

No site do Android, na seção do SDK, será feito o download do SDK para Linux na página (ANDROID-SDK) e, depois segue-se o link (ANDROID-SDK-INSTALLING) para se obter o tutorial de instalação do SDK, o qual descreve como obter plataformas e outros componentes. Depois este link (ADT-PLUGIN) ensina como instalar o Android Development Tools plugin para o Eclipse e esse outro link (HELLO-WORLD), descreve como criar um Android Virtual Devices e rodar o emulador.

Para se obter e recompilar o kernel do Android, ativando-se os módulos necessários, segue-se os passos explicados nos sites (CROSS-COMPILATION).

Esse kernel recompilado é que será utilizado no emulador do Android para que seja possível carregar o Firmament e executar os testes de injeção de falhas no emulador do Android.

3.3. Recompilando o Firmament para o Android

Esta parte foi conduzida utilizando-se como referência o artigo (ACKER 2010), o qual foi muito útil para compilar os arquivos firm_asm e mas_mrif, os quais pertencem ao módulo do Firmament. A solução apresentada no site (LOADABLE-KERNEL-MODULE) serviu de apoio para realizar a compilação cruzada da máquina virtual firm_vm. A figura 1 mostra como isso deve ser feito:

O primeiro passo é compilar os arquivos firm_asm e msa_mrif. Obter o Firmament versão 2.6.29 e extrair em uma pasta no diretório ***home***. Os dois arquivos citados estão dentro da pasta ***asm***, dentro da pasta principal do ***Firmament***.

1. Agora, entrar no diretório ***mydroid***, criado de acordo com o tutorial para obtenção dos arquivos fontes do Android, digitando no terminal os comandos ***$cd*** e depois ***$cd mydroid***. Dentro desse diretório, encontramos várias pastas e dentre elas a pasta ***external***. Essa pasta contém os programas que estarão disponíveis no sistema.

2. Assim, entrar na pasta ***externa***l com o comando ***$cd external*** e criar uma pasta, com o nome de ***myapps*** executando ***$mkdir myapps*** no terminal. A seguir, copiar uma das pastas originais que existem dentro da pasta ***external***, como a ***ping*** por exemplo. Depois renomeia-se a pasta ***ping***, a qual foi copiada para dentro da pasta ***myapps***, para ***firm_asm***.

3. O próximo passo é entrar na pasta ***firm_asm*** e excluir todos os outros arquivos, deixando apenas o arquivo ***Android.mk*** dentro da pasta. Voltar para a pasta ***asm*** dentro do diretório do ***Firmament*** e copiar os seus arquivos fontes para dentro da pasta ***firm_asm***.

4. A seguir, editar o arquivo ***Android.mk*** e alterar as variáveis ***LOCAL_SRC_FILES e LOCAL_MODULE*** para o nome do programa e o nome do executável a ser gerado, isto é, ***LOCAL_SRC_FILES:= firm_asm.c e LOCAL_MODULE := firm_asm*** .

5. Agora precisa-se setar as variáveis de compilação com o comando:

$source /home/nome_usuário/mydroid/build/envsetup.sh e depois executar o comando ***$mm*** para compilar o módulo ***firm_asm***.

6. O executável resultante da compilação está no diretório ***/home/nome_usuário/mydroid/out/target/product/generic/system/bin/***.

7. Para compilar o ***msa_mrif.c***, repete-se os passos anteriores com a devida alteração do nome do arquivo a ser compilado.

8. Para a compilação da máquina virtual ***firm_vm***, precisa-se ajustar as bibliotecas e o kernel a ser utilizado na compilação.

Deve-se executar os seguintes comandos no terminal:

$cd

$ firmament-2.6.29/module

9. Depois, entrar na pasta ***module***, dentro do diretório do ***Firmament***. Dentro dessa pasta, encontra-se o arquivo ***Makefile***. Para alterar esse arquivo, digitar ***$gedit Makefile*** . Precisa-se adicionar essas duas linhas logo abaixo da linha ***obj-m = firm_vm.o***:

CROSS_COMPILE=/home/nome_usuário/mydroid/prebuilt/linux-x86/toolchain/arm-eabi-4.3.1/bin/arm-eabi-

KERNEL_DIR ?=/home/nome_usuário/kernel/

10. A seguir, alterar as linhas

all: build

build:

make -C $(KSRC) SUBDIRS=`pwd` modules

para apenas

all:

make -C $(KERNEL_DIR) M=`pwd` ARCH=arm CROSS_COMPILE=$(CROSS_COMPILE) modules

Isso garante que o Firmament vai ser compilado para a arquitetura ARM e a versão 2.6.29 do kernel goldfish do Android.

11. A parte seguinte é a retirar esta parte do código pois não tem função alguma:

install:

@# completely broken ATM - removes the whole dir

@#make -C $(KSRC) SUBDIRS=`pwd` modules_install

@echo "Install it yourself ..."

12.Salvar e fechar o arquivo ***Makefile***.

13. Agora é preciso entrar na pasta do ***Firmament***, a qual está no diretório ***home***. Para isso utiliza-se os comandos a seguir:

$cd

$cd firmament-2.6.29/

14. Depois, digitar no terminal os seguintes comandos:

$export ARCH=arm

$export CROSS_COMPILE=/home/nome_usuário/mydroid/prebuilt/

linux-x86/toolchain/arm-eabi-4.3.1/bin/arm-eabi-

$make

15. Assim, encontra-se o executável ***firm_vm.ko***, o qual é utilizado no emulador do Android junto com os outros dois arquivos que já foram compilados anteriormente: o ***firm_asm*** e o ***msa_mrif***.

Figura 1: Passos para recompilação do Injetor Firmament

3.4. Copiando os arquivos do Firmament para o Android

A figura 2 mostra como se deve proceder para copiar os arquivos compilados do Firmament para dentro do emulador do Android, utilizando-se para isso o Android Debug Bridge ou simplesmente ADB, o qual está referenciado no site (ANDROID-DEBUG-BRIDGE).

1. Agora abrir uma janela do terminal e digitar os seguintes comandos para garantir que o terminal está no nosso diretório ***home*** e, também, para criar uma pasta chamada ***firmament*** ali.

$cd

$mkdir firmament

2. Copiar para dentro dela os arquivos compilados do Firmament: os arquivos ***firm_asm*** e ***msa_mrif*** e o ***firm_vm.ko***, os quais estão no local especificado anteriormente.

3. Agora, executar o emulador com o kernel que foi recompilado da seguinte maneira:

No tutorial do hello world no site do Google, foi ensinado a criar um AVD com o nome de ***my_avd***. Assim, digitar no terminal:

$emulator -avd my_avd -kernel /home/nome_usuário/kernel/arch/

arm/boot/zImage

Assim, o emulador será executado com o kernel que foi recompilado.

4. Agora abrir outra janela do terminal e digitar o seguinte comando para entrar no adb do emulador:

$adb shell

5. O próximo passo é entrar na pasta ***data*** e criar dentro dela uma pasta chamada de ***firmament***:

#cd data

#mkdir firmament

6. Executar o comando a seguir para voltar ao terminal do Linux

#exit

7. A seguir, executar o seguinte comando para copiar os arquivos, os quais estão dentro da pasta ***firmament*** no diretório ***home***, para dentro da pasta ***firmament***, a qual foi criada dentro da pasta ***data*** no diretório do emulador:

$adb push /home/nome_usuário/firmament data/firmament

Assim, os arquivos recompilados do Firmament agora estão dentro do emulador do Android, no caminho ***/data/firmament***.

Figura 2: Copiando os arquivos do Firmament para dentro do Android

3.5. Verificando o Firmament dentro do Android

Depois disso foi criado um arquivo de teste apenas para ter certeza de que o Firmament estava funcionando dentro do emulador do Android. O arquivo criado tem o nome de teste, o qual foi copiado de forma semelhante para dentro do emulador do Android.

A figura 3 mostra como o arquivo é composto:

```
add r0 r1
acp
```

Figura 3: Conteúdo do arquivo utilizado para testar o Firmament dentro do Emulador do Android

Ele foi executado com os seguintes comandos da figura 4 para assegurar o funcionamento do Firmament dentro do Android:

```
#cd data
# cd firmament
# dmesg -c
# ./firm_asm teste
# ./firm_asm teste /proc/net/firmament/rules
# echo "stopflow all"
# stopflow all
# echo "stopflow all" > /proc/net/firmament/control
# ./firm_asm teste /proc/net/firmament/rules/ipv4_out
# echo "startflow ipv4_out" > /proc/net/firmament/control
```

Figura 4: Comandos utilizados para testar o funcionamento do Firmament

Isso apenas assegurou que o Firmament estava respondendo, porém falta fazer testes com perdas e atrasos de pacotes para que se possa fazer uma melhor análise.

3.6. Restrições

Os arquivos fontes do Android só podem ser compilados no sistema operacional Linux ou Mac OS. Assim, pode-se utilizar uma máquina virtual no Windows para executar uma versão do Linux ou fazer uma instalação independente de uma distribuição do Linux.

Na hora de fazer a compilação cruzada para recompilar o kernel do Android para a arquitetura ARM, Advanced Risc Machine, foi encontrada dificuldade pois foi necessário buscar material adequado para que fosse possível realizar esta etapa do trabalho. Esses mesmos problemas ocorreram na hora de fazer a compilação cruzada para o módulo da máquina virtual firm_vm do Firmament.

4. Conclusão

Por seu código ser aberto, o Android já tem uma grande quantidade de aplicações desenvolvidas para ele e esse número tende a crescer ainda mais. Assim, é necessário que essas aplicações sejam testadas para ver como elas se comportam na presença de falhas, o que permitiria aos desenvolvedores tomarem medidas para evitar transtornos aos usuários.

Firmament permite executar testes que explorem falhas que normalmente só aparecem em situações reais, pois só assim se terá certeza de que uma aplicação está realmente funcionando bem. Esse tipo de teste é uma das principais características desta ferramenta.

O passo a passo descrito aqui será de grande ajuda para quem desejar portar outras ferramentas para o ambiente Android, pois a compilação cruzada de módulos do Linux para o Android foi apresentada de forma objetiva e bem detalhada.

Referências

CANALYS. Disponível em: < http://www.canalys.com/index.html >. Acessado em: agosto, 2010.

AVIZIENIS, A.; Laprie, J.; Randell B,; Landwehr C. Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE trans. on dependable and secure computing, V. 1, n. 1, jan 2004, pp 11-33.

RUSSEL, R.; Welte, H. (2002). Linux net filter hacking HOWTO. 2002. Disponível em: < http://www.netfilter.org/documentation/ >.

DREBES, R. J. FIRMAMENT: Um módulo de injeção de falhas de comunicação para linux. 2005. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

SIQUEIRA, Tórgan Flores de ; Fiss, B. C. ; WEBER, Raul Fernando ; CECHIN, Sergio Luis ; WEBER, T. S. . Applying FIRMAMENT to test the SCTP communication protocol under network faults. In: Test Workshop, 2009. LATW '09. p. 1-6.

ANDROID. Disponível em: < http://www.android.com/ >. Acessado em: abril, 2010.

ANDROID-SOURCE. Disponível em: < http://source.android.com/source/download.html >. Acessado em: maio, 2010.

ANDROID-SDK. Disponível em: < http://developer.android.com/sdk/index.html >. Acessado em: abril, 2010.

ANDROID-SDK-INSTALLING. Disponível em: < http://developer.android.com/sdk/installing.html >. Acessado em: abril, 2010.

ADT-PLUGIN. Disponível em < http://developer.android.com/sdk/eclipse-adt.html >. Acessado em: abril, 2010.

HELLO-WORLD. Disponível em < http://developer.android.com/resources/tutorials/hello-world.html >. Acessado em: abril, 2010.

CROSS-COMPILATION. Disponível em: < http://wiki.strongswan.org/projects/strongswan/wiki/Android , http://my.opera.com/otaku_2r/blog/download-and-build-the-google-android e http://linuxclues.blogspot.com/2010/05/build-compile-linux-kernel-android.html >. Acessados em: maio, 2010.

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

LOADABLE-KERNEL-MODULE. Disponível em: < http://groups.google.com/group/android-kernel/msg/a43f2b1db7a5c3dc?pli1 >. Acessado em: maio, 2010.

ANDROID-DEBUG-BRIDGE. Disponível em: < http://developer.android.com/guide/developing/tools/adb.html >. Acessado em: abril, 2010.