FRESCO: Modular Composable Security Services for Software-Defined Networks

Seugwon Shin[1], Phillip Porras[2], Vinod Yegneswaran[2], Martin Fong[2] , Guofei Gu[1] , Mabry Tyson[2]

[1]Texas A&M University
[2]SRI International

Agenda

- Introdução
- Contribuição
- FRESCO
  - FRESCO Development Enviroment
  - FRESCO Resource Controller
  - FRESCO SEK
  - FRESCO Script Language
- Exemplos
  - Reflector Net
  - Legacy Security Applications
- Testes
- Conclusão
- Referências

Agenda

- Introdução
- Contribuição
- FRESCO
  - FRESCO Development Enviroment
  - FRESCO Resource Controller
  - FRESCO SEK
  - FRESCO Script Language
- Exemplos
  - Reflector Net
  - Legacy Security Applications
- Testes
- Conclusão

Introdução

OpenFlow

- OpenFlow é uma ferramenta baseada no paradigma de SDN.
- Padrão que vem atraindo cada vez mais o interesse da comunica de de redes de computadores.
- Se destaca por dividir o plano de dados do plano de controle em switchs OpenFlow.

Introdução

OpenFlow

- Habilita softwares a tomarem decisões para otimizar fluxos de rede de acordo com a demanda.

- Para redes modernas, que precisam lidar com virtualização de hosts, migração de aplicações, migração de máquinas virtuais, streamings de vídeos, etc, o OpenFlow pode oferecer a agilidade necessária para lidar com orquestração dinâmica de redes, o que os switchs das redes tradicionais não podem oferecer.

Introdução

OpenFlow

- Para um switch de OpenFlow, o plano de dados é programável, onde os regreas de fluxos são especificados através da Tabela de Fluxos (Flow Table).

- A tabela de fluxos contem uma série de regras que ditam como processar todos os fluxos de rede que passam pelo switch.

- Em resumo, a tabela de fluxo do OpenFlow contém as instruções que coordenam como encaminhar, modificar e bloquear cada pacote que passam pelo switch.

Introdução

OpenFlow

- Um controlador OpenFlow controla um conjunto de switchs OpenFlow.

- Eles suportam o protocolo OpenFlow, que permitem ao controlador realizar mudanças na tabelas de fluxo com rotas predefinidas ou novas rotas.

Introdução

OpenFlow

- Do ponto de vista de segurança de redes, o OpenFlow oferece aos pesquisadores inúmeras possibilidades de controle do fluxo de rede através do controlador.

- Uma aplicação de segurança OpenFlow pode implementar lógicas muito mais complexas do que bloqueio e encaminhamento de tráfego:
  - Lógica de produção de regras que incorporem o estado da conexão(aberta, fechada, hand-shake, entre outros);
  - Procedimentos de quarentena;
  - Migração de conexões;
  - Algoritmos de detecção de fluxos maliciosos poderiam ser redesenhados;

Agenda

- Introdução
- Contribuição
- FRESCO
  - FRESCO Development Enviroment
  - FRESCO Resource Controller
  - FRESCO SEK
  - FRESCO Script Language
- Exemplos
  - Reflector Net
  - Legacy Security Applications
- Testes
- Conclusão

Contribuição

FRESCO: um framework de desenvolvimento de aplicações de segurança para switchs OpenFlow. Este framework simplifica o desenvolvimento e deploy de aplicações.

Contribuição

Framework for Enabling Security Controls in OpenFlow networks

Contribuição

- Provê um API para criação de scritps que permite a criação de módulos de monitoramento e detecção de ameaças de segurança.

- Esses módulos podem trabalhar individualmente e em conjunto. Até a data de publicação do artigo, 2013, FRESCO possuía 16 módulos nativos.

Contribuição

- Integração com aplicações OpenFlow legadas e com aplicações baseadas em DPI(Deep Package Inspection), ou seja, IDSs e IPSs. Ex.: BotHunter, Snort, etc.

- Possui uma base de dados(FRESCO-DB) que simplifica o armazenamento e gerenciamento de sessões

Agenda

- Introdução
- Contribuição
- FRESCO
  - FRESCO Development Enviroment
  - FRESCO Resource Controller
  - FRESCO SEK
  - FRESCO Script Language
- Exemplos
  - Reflector Net
  - Legacy Security Applications
- Testes
- Conclusão

FRESCO

Arquitetura

- Camada de aplicação
  - Possui interpretador para os scripts(que são convertidos em módulos)
  - API para combinação entre módulos.
  - Execução dos módulos

- SEK - Security Enforcement Kernel
  - Interface com controlador OpenFlow.
  - Reforça as políticas de segurança implementadas.

FRESCO

Camada de Aplicação:

- Implementada utilizando os módulos python do NOX(Controlador OpenFlow).
- *FRESCO Development Enviroment(DE)*
  - Módulos
    - Elemento mais importante do FRESCO, pois todas as funções de segurança rodam a partir deles
    - Baseado em eventos
- *Resource Controller(RC)*

Agenda

- Introdução
- Contribuição
- FRESCO
  - FRESCO Development Enviroment
  - FRESCO Resource Controller
  - FRESCO SEK
  - FRESCO Script Language
- Exemplos
  - Reflector Net
  - Legacy Security Applications
- Testes
- Conclusão

FRESCO Development Enviroment(DE)

- API que abstrai toda complexidade do NOX e do protocolo OpenFlow.
- Abstrai as aplicações de coletar e gerenciar dados.
- Fornece 4 funções:

FRESCO Development Enviroment(DE)

Script-to-module translation

- Converte scripts para módulos e cria instâncias a partir de módulos.

FRESCO Development Enviroment(DE)

Database management

- Coleta e armazena informações de diferentes tipos de redes e informações de estado de conexão e provê uma interface para que as instâncias usem essa informação.

- Compartilha informações entre instâncias.

FRESCO Development Enviroment(DE)

Event Management

- Notifica uma instância sobre a ocorrência de um evento pré-definido.
- Suporte diferentes tipos de eventos: flow arrivals, denied connections, e session resets.
- Além do suporte de aplicações baseadas em DPI.

FRESCO Development Enviroment(DE)

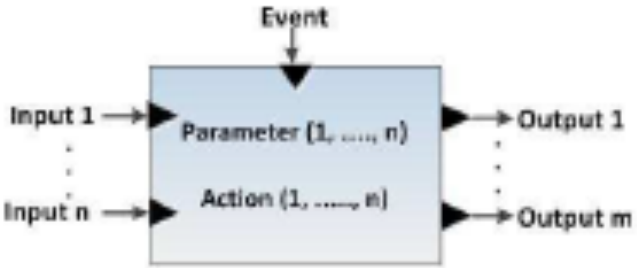
Instance Execution

- Carrega uma instância na memória, a partir de um módulo.

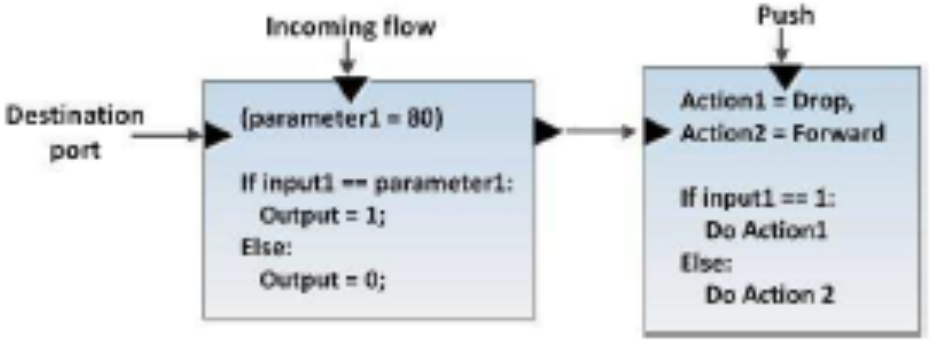
FRESCO Develoment Enviroment(DE)

- Cada módulo pode ser construído a partir da definição de 5 interfaces:
  - ***Input***: recebe valores;
  - ***Output***: transmite valores;
  - ***Parameter***: configuração específica do módulo ou inicialização de valores;
  - ***Action***: ação em cima de pacotes ou fluxos de rede;
  - ***Event***: notifica um módulo quando é necessário um processamento.

FRESCO Develoment Enviroment(DE)



Módulo Generalizado



Combinação de módulos:

Port Comparator + DROP Packages

FRESCO Develoment Enviroment(DE)

Suporta as seguintes ações:

- Básicas
  - *Drop*
  - *Output*(*forward*)
  - *Group*: processa um pacote se o mesmo fizer parte de um grupo espeficado.

Todas estão disponíveis em alto nível no FRESCO.

FRESCO Develoment Enviroment(DE)

- FRESCO utiliza uma das ações opcionais
  - *Set Action*: permite reescrever o cabeçalho de um pacote(IP e porta) para redirecionamento. Para simplificar o desenvolvimento, FRESCO quebra essa ação em outras três:
    - *Redirect*: redireciona pacotes de um host destino original para outro.
    - *Mirror:* copia pacotes e redireciona para uma parte de espelho para análises futuras.
    - *Quarantine*: isola um fluxo que uma rede considerado suspeito marcando pacotes de um fluxo suspeito.

Agenda

- Introdução
- Contribuição
- FRESCO
  - FRESCO Development Enviroment
  - FRESCO Resource Controller
  - FRESCO SEK
  - FRESCO Script Language
- Exemplos
  - Reflector Net
  - Legacy Security Applications
- Testes
- Conclusão

FRESCO Resource Controller

- Responsável por monitorar os switches OpenFlow e controlar os seus estados:
  - Como a tabela de fluxos de um switch possui tamanho limitado, existe a possibilidade de uma regra FRESCO não ser inserida. Entretanto, como tal regra lida com políticas de segurança, esta regra precisa de instalação imediata na tabela de fluxo. Assim, o FRESCO RC pode retirar da tabela regras antigas e obsoletas. É o que o autor chama de FRESCO *Garbage Collector.*

Agenda

- Introdução
- Contribuição
- FRESCO
  - FRESCO Development Enviroment
  - FRESCO Resource Controller
  - FRESCO SEK
  - FRESCO Script Language
- Exemplos
  - Reflector Net
  - Legacy Security Applications
- Testes
- Conclusão

FRESCO SEK

SEK - Security Enforcement Kernel

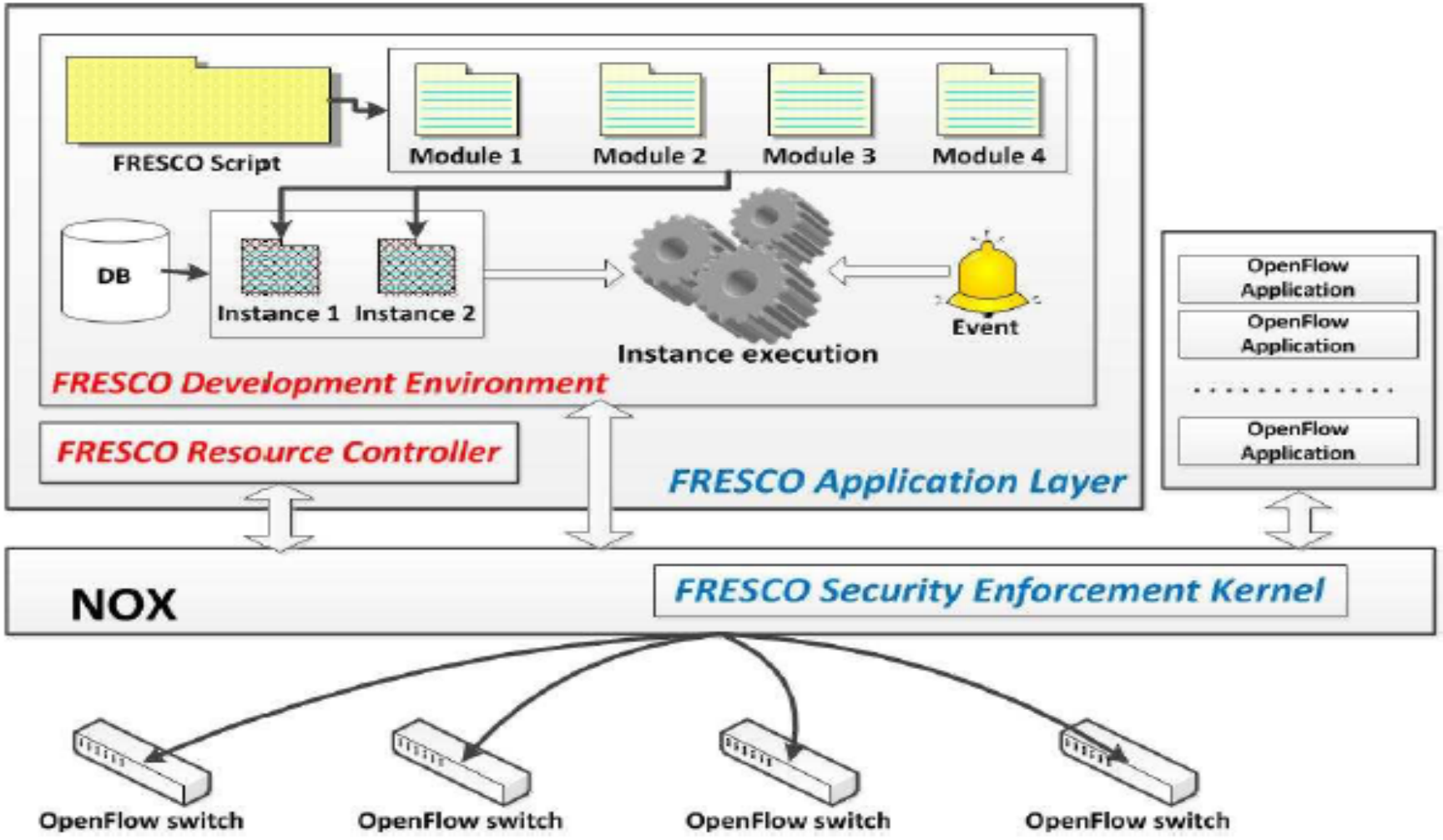
- Interface com controlador NOX.
- Traduz políticas de segurança implementadas em FRESCO em regras OpenFlow.
- Lida com conflitos de regras.

FRESCO SEK

Exemplo

- Um conjunto de políticas de segurança criadas no FRESCO coloca um servidor vulnerável em quarentena. Entretanto, uma outra aplicação OpenFLow de *load-balance* assume que o servidor em quarentena é o menos utilizado no momento, e desvia alguns pedidos para ele, para balancear a carga em um conjunto de servidores.

- Assim, o SEK garante que regras de fluxo derivadas de implementações FRESCO sejam priorizadas e executadas sobre regras criadas por outros tipos de aplicações


FRESCO Script
Module 1
Module 2
Module 3
Module 4
DB
Instance 1
Instance 2
Instance execution
Event
FRESCO Development Environment
FRESCO Resource Controller
FRESCO Application Layer
OpenFlow Application
OpenFlow Application
OpenFlow Application
NOX
FRESCO Security Enforcement Kernel
OpenFlow switch
OpenFlow switch
OpenFlow switch
OpenFlow switch

Agenda

- Introdução
- Contribuição
- FRESCO
  - FRESCO Development Enviroment
  - FRESCO Resource Controller
  - FRESCO SEK
  - FRESCO Script Language
- Exemplos
  - Reflector Net
  - Legacy Security Applications
- Testes
- Conclusão

FRESCO Script Language

- Para simplificar o desenvolvimento de aplicações de segurança, FRESCO possui uma linguagem de script própria.
- Para criar um novo módulo, se faz necessário a definição de 6 variáveis:
  - Type: define o tipo do módulo;
  - Input
  - Output
  - Parameter
  - Action
  - Event

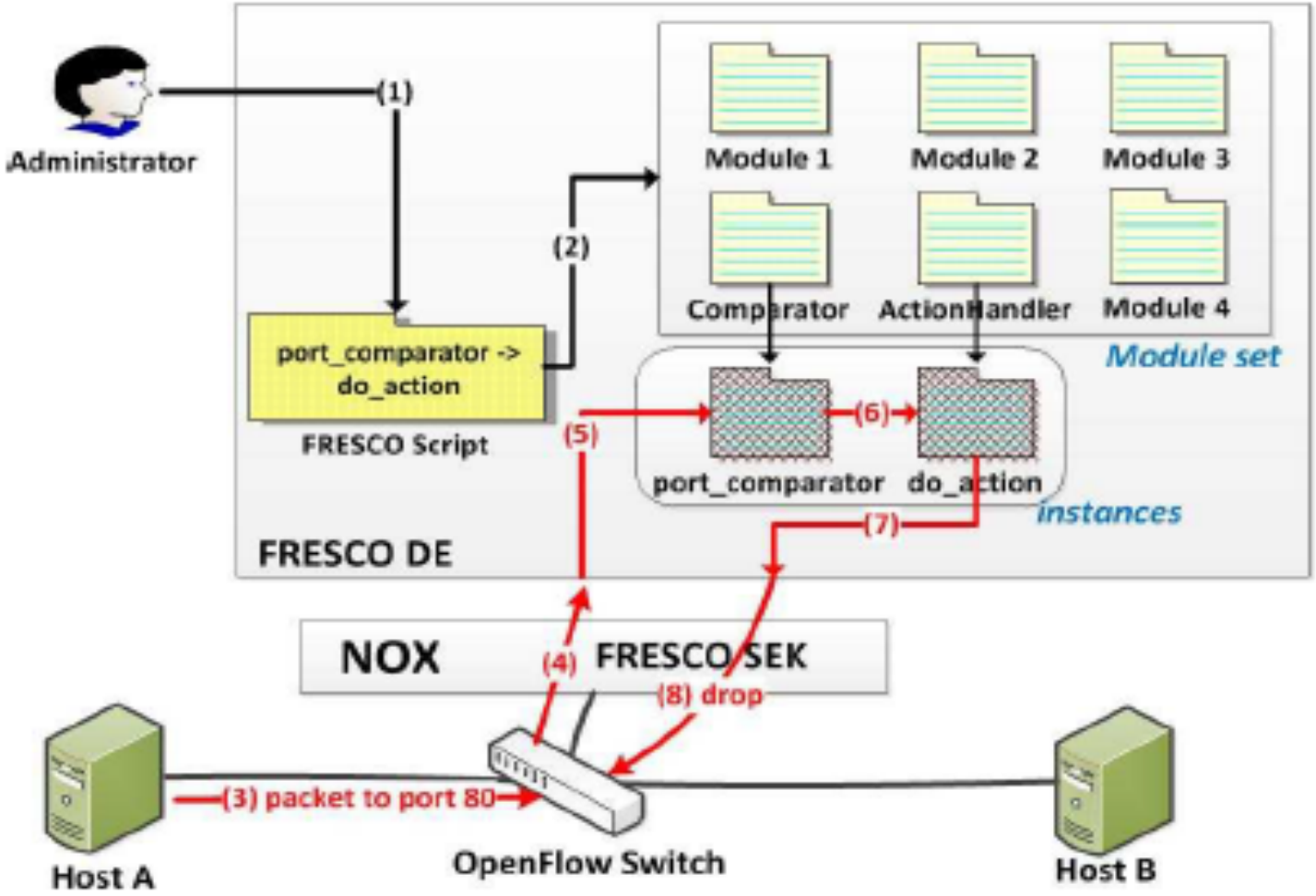
FRESCO Script Language

```
port_comparator (1)(1) {
   type:Comparator
   event:INCOMING_FLOW
   input:destination_port
   output:comparison_result
   parameter:80
/* no actions are defined */
   action: -
}
```

```
do_action (1)(0) {
   type:ActionHandler
   event:PUSH
   input:comparison_result
   output: - /* no outputs are defined */
   parameter: - /* no parameters are defined */
/* if input equals to 1, drop, otherwise, forward */
   action:comparator_result == 1 ? DROP : FORWARD
}
```

FRESCO Script Language

Variable	*Explanation*	*Possible Values*
instance name (#input)(#output)	denotes an instance name (should be unique)	(#input) and (#output) denote the number of inputs and outputs
type: [module]	denotes a module for this instance	[module] names an existing module
input: a_1,a_2,...	denotes input items for a module	a_n may be set of flows, packets or integer values
output: b_1, b_2,...	denotes output items for a module	b_n may be set of flows, packets or integer values
parameter: c_1,c_2,...	denotes configuration values of a module	c_n may be real numbers or strings
event: d_1,d_2,...	denotes events delivered to a module	d_n may be any predefined string
action : condition ? action,...	denotes set of conditions and actions performed in the module	condition follows the same syntax of *if condition* of python language; action may be one of the following strings (DROP, FORWARD, REDIRECT, MIRROR, QUARANTINE)
{ }	denotes the module start ({) and end (})	-


Administrator
(1)
(2)
Module 1
Module 2
Module 3
Comparator
ActionHandler
Module 4
Module set
port_comparator ->
do_action
FRESCO Script
(5)
(6)
port_comparator
do_action
instances
(7)
FRESCO DE
NOX
FRESCO SEK
(4)
(8) drop
(3) packet to port 80
Host A
OpenFlow Switch
Host B

Agenda

- Introdução
- Contribuição
- FRESCO
  - FRESCO Development Enviroment
  - FRESCO Resource Controller
  - FRESCO SEK
  - FRESCO Script Language
- Exemplos
  - Reflector Net
  - Legacy Security Applications
- Testes
- Conclusão

Exemplos

Dois Cenários foram implementados:

- Reflector Net: permitir que uma rede OpenFlow redirecione um scanner malicioso para um honeypot.

- Legacy Security Applications: comunicação com um aplicação BotMiner

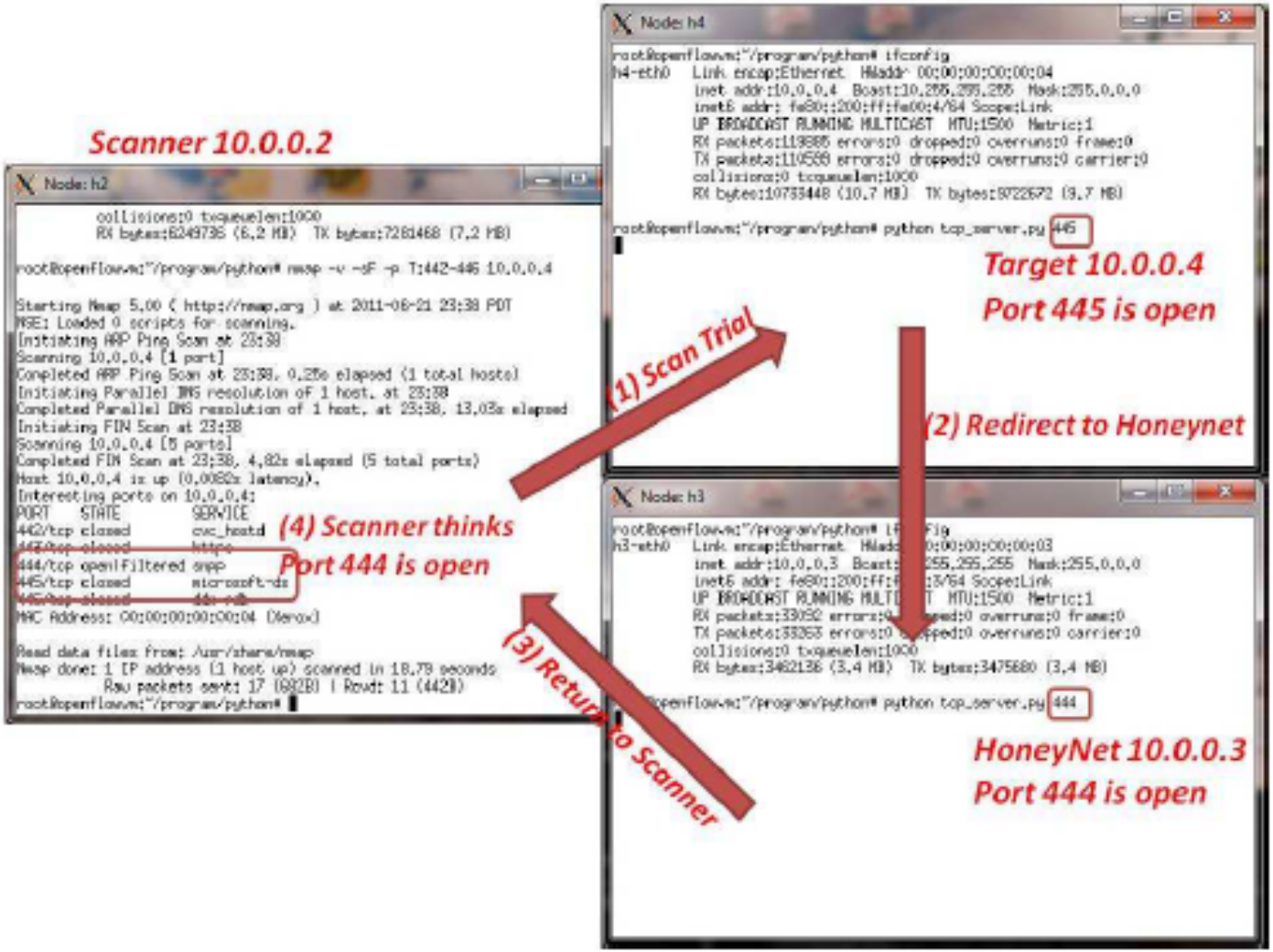
Agenda

- Introdução
- Contribuição
- FRESCO
  - FRESCO Development Enviroment
  - FRESCO Resource Controller
  - FRESCO SEK
  - FRESCO Script Language
- Exemplos
  - Reflector Net
  - Legacy Security Applications
- Testes
- Conclusão

Reflector Net

```
find_scan (1)(2){
   type:ScanDetector
   event:TCP_CONNECTION_FAIL
   input:source_IP
   output:source_IP, scan_result
   parameter:5
/* no actions are defined */
   action: -
}
```

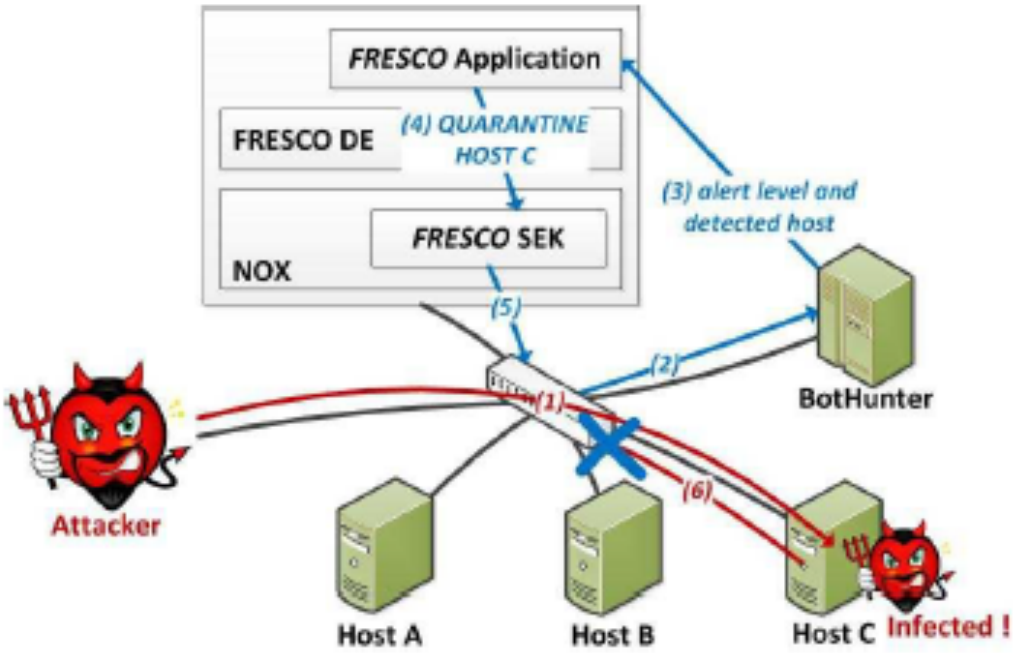
```
do_redirect (2)(0){
   type:ActionHandler
   event:PUSH
   input:source_IP, scan_result
   output: -
   parameter: -
/* if scan_result equals 1, redirect,
   otherwise, forward */
   action: scan_result == 1 ?
                   REDIRECT : FORWARD
}
```


Scanner 10.0.0.2
Node: h2
collisions:0 txqueuelen:1000
RX bytes:6249735 (6,2 MB) TX bytes:7261468 (7,2 MB)
root@Openflowvm:"/program/python# nmap -v -sF -p T:442-446 10.0.0.4
Starting Nmap 5.00 (http://nmap.org) at 2011-06-21 23:38 PDT
NSE: Loaded 0 scripts for scanning.
Initiating ARP Ping Scan at 23:38
Scanning 10.0.0.4 [1 port]
Completed ARP Ping Scan at 23:38, 0.25s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 23:38
Completed Parallel DNS resolution of 1 host. at 23:38, 13,03s elapsed
Initiating FIN Scan at 23:38
Scanning 10.0.0.4 [5 ports]
Completed FIN Scan at 23:38, 4,82s elapsed (5 total ports)
Host 10.0.0.4 is up (0,0082s latency).
Interesting ports on 10.0.0.4:
PORT STATE SERVICE
442/tcp closed cvc_hostd
443/tcp closed https
444/tcp open|filtered snpp
445/tcp closed microsoft-ds
446/tcp closed nds_nn
MAC Address: 00:00:00:00:00:04 (Xerox)
Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 18.79 seconds
Raw packets sent: 17 (692B) | Rcvd: 11 (442B)
root@Openflowvm:"/program/python#
Node: h4
root@Openflowvm:"/program/python# ifconfig
h4-eth0 Link encap:Ethernet HWaddr 00:00:00:00:00:04
inet addr:10.0.0.4 Bcast:10.255.255.255 Mask:255.0.0.0
inet6 addr: fe80::200:ff:fe00:4/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:113885 errors:0 dropped:0 overruns:0 frame:0
TX packets:110588 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:10733448 (10.7 MB) TX bytes:9722672 (9.7 MB)
root@Openflowvm:"/program/python# python tcp_server.py 445
Target 10.0.0.4
Port 445 is open
(1) Scan Trial
(2) Redirect to Honeynet
Node: h3
root@Openflowvm:"/program/python# ifconfig
h3-eth0 Link encap:Ethernet HWaddr 00:00:00:00:00:03
inet addr:10.0.0.3 Bcast:10.255.255.255 Mask:255.0.0.0
inet6 addr: fe80::200:ff:fe00:3/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:33092 errors:0 dropped:0 overruns:0 frame:0
TX packets:33263 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:3462136 (3.4 MB) TX bytes:3475680 (3.4 MB)
root@Openflowvm:"/program/python# python tcp_server.py 444
HoneyNet 10.0.0.3
Port 444 is open
(3) Return to Scanner
(4) Scanner thinks
Port 444 is open

Agenda

- Introdução
- Contribuição
- FRESCO
  - FRESCO Development Enviroment
  - FRESCO Resource Controller
  - FRESCO SEK
  - FRESCO Script Language
- Exemplos
  - Reflector Net
  - Legacy Security Applications
- Testes
- Conclusão

Legacy Security Applications



Legacy Security Applications

```
do_quarantine (2)(0){
   type:ActionHandler
   event:MESSAGE_LEGACY:FRESCO
   input:victim_ip,confidence_score
   output: -
   parameter:confidence_threshold
/* redirect all flows from source IP */
   action:confidence_score > confidence_threshold
          ? QUARANTINE(victim_ip)
}
```

Agenda

- Introdução
- Contribuição
- FRESCO
  - FRESCO Development Enviroment
  - FRESCO Resource Controller
  - FRESCO SEK
  - FRESCO Script Language
- Exemplos
  - Reflector Net
  - Legacy Security Applications
- Testes
- Conclusão

Testes

Implementação de dois algoritmos de decteção de anomalias na rede

- TRW-CB: detecta hosts infectados na rede pelo número de conexões TCP recusadas. Cria uma fila de hosts com suas respectivas probabilidades: a cada falha no TCP a probabilidade de ser considerado um host infectado aumenta, em contra-partida, a cada conexão com sucesso diminui a probabilidade.

Testes

- Rate Limit: a partir da observação que um vírus quando vai se propagar tenta conectar-se a máquinas diferentes em um curto espaço de tempo.
  - A cada novo pedido de conexão, verifica-se se o host de origem está uma lista de hosts confiáveis, se sim, continua o protocolo para conexão, se não coloca o host numa fila de atraso e só liberar a conexão depois de um tempo threadshoud estabelecido. Se um novo pedido de conexão chegar dentro do tempo limite estabelecido, libera um alerta.

Testes

Algorithms	Implementation		
	Standard	OpenFlow application	*FRESCO*
TRW-CB	1,060	741	66 (58 + 8)
Rate Limit	991	814	69 (61 + 8)

- Código OpenFlow representa de 70% a 80% da implementação padrão.
- Código FRESCO representa 6% a 7% da implementação padrão e 9% da implementação OpenFlow padrão.

Testes

	NOX	Simple Flow Tracker	Simple Scan Detector	Threshold Scan Detector	BotMiner Detector
Time (ms)	0.823	1.374	2.461	7.196	15.421

- Foram capturados os pacotes entre o controlador NOX e o switch OpenFlow com o objetivo de medir o tempo de sumissão de novo fluxo para o switch e o tempo de confirmação de implantação da regra.

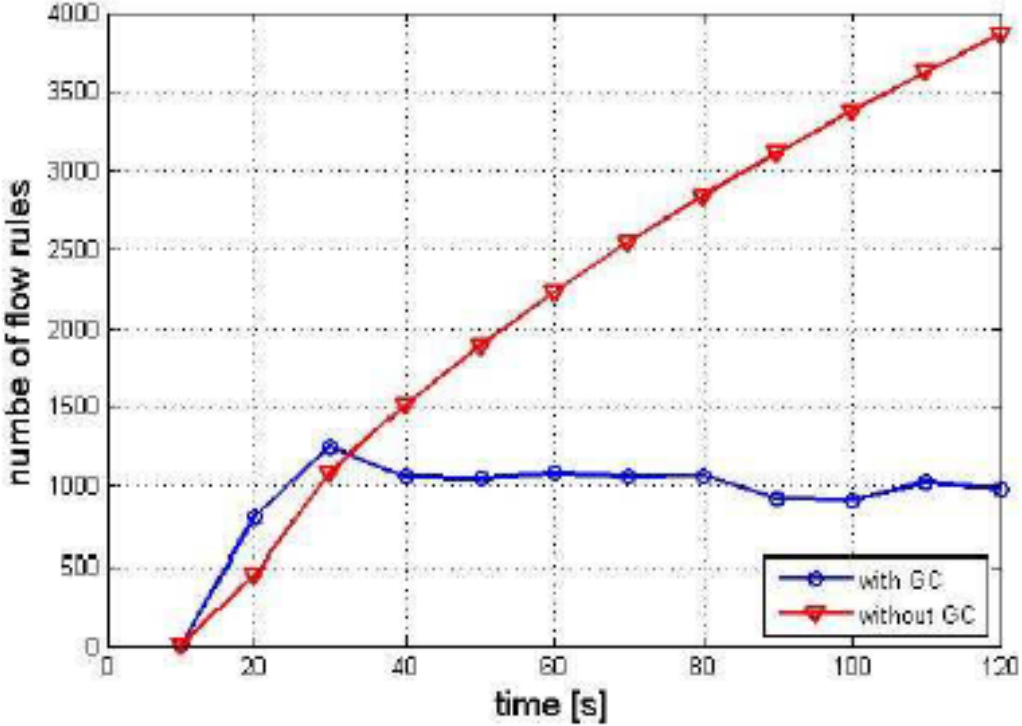
- Aplicações FRESCO precisam de 0.5 a 10.9 milissegundos de tempo para adicionar suas regras.

Testes

- Aplicação não FRESCO adicionou 4000 regras em um switch Openflow.

- Threshould considerado: 75% de 4000.

Testes


numbe of flow rules
time [s]
with GC
without GC
0
500
1000
1500
2000
2500
3000
3500
4000
0
20
40
60
80
100
120

Agenda

- Introdução
- Contribuição
- FRESCO
  - FRESCO Development Enviroment
  - FRESCO Resource Controller
  - FRESCO SEK
  - FRESCO Script Language
- Exemplos
  - Reflector Net
  - Legacy Security Applications
- Testes
- Conclusão

Conclusão

- Desenvolvimento e implantação de aplicações OpenFlow ainda são um desafio.

- Foi introduzido o FRESCO, um framework para desenvolvimento de aplicações OpenFlow visando atacar o problema citado.

- Para que as políticas de segurança fossem enfatizadas foi acoplado junto ao framework o FRESCO SEK.

- Planejam lançar o código junto a comunidade OpenFlowSDN

Referências

- J. Jung, V. Paxson, A. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In Proceedings of IEEE Symposium on Security and Privacy, 2004.

- S. A. Mehdi, J. Khalid, and S. A. Khayam. Revisiting Traffic Anomaly Detection Using Software Defined Networking. In Proceedings of Recent Advances in Intrusion Detection, 2011.

- S. Schechter, J. Jung, and A. Berger. Accuracy Improving Guidelines for Network Anomaly Detection Systems. In Proceedings of International Symposium on Recent Advances Intrusion Detection.

- J. Twycross and M. M. Williamson. Implementing and testing a virus throttle. In Proceedings of the USENIX Security Symposium, 2003.