Gestão e Segurança de Redes

Mestrado em Engenharia Informática
Universidade do Minho

SNMPv2cSec

Tiago Gomes
PG47696

Braga, 13 de fevereiro de 2023

Conteúdo

Lista de Figuras

1. Contextualização

O propósito do projecto insere-se na criação de um modelo SNMP que inclua mecanismos de segurança para garantia de privacidade e verificação da integridade dos dados. O objetivo é o desenvolvimento de um modelo capaz de garantir uma comunicação entre um manager SNMP, e um agent SNMP, em que como middleware existe um proxy responsável pela implementação de garantias de segurança na comunicação entre manager e proxy, sendo que a comunicação entre proxy e agente realiza-se de uma forma "insegura", apenas segundo as normas do SNMPv2c com recurso a uma *community string*.

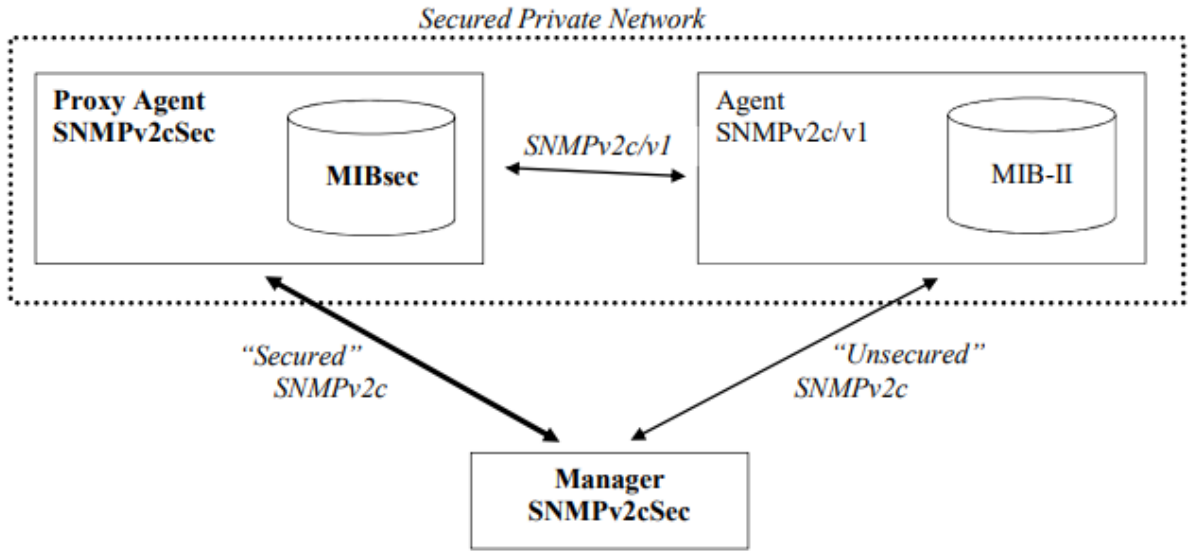

Figura 1.1: Visão Geral

2. Abordagem

2.1 Abstração

Para alcançar o propósito definido, foi desenvolvido uma *suite* composta por 3 programas, cada um representativo do modelo mencionado. Assim sendo temos a seguinte divisão em três componentes:

- **Manager.py** : Programa representativo de um manager snmp, capaz de:
  - **Autenticação**: com o respectivo proxy SNMP, via username, password e ainda com **2FA**, sendo o respectivo gerado no momento de autenticação.
  - **Execução de primitivas SNMP**: get-request, get-bulk, set-requests.
  - **Interpretação de Traps**: Escuta de notificações SNMP.
- **Proxy.py** : Middleware responsável pela comunicação entre o manager e o agent capaz de:
  - **Autenticação**: Verificação de credenciais e comunicação com manager para início de sessão (username, password e 2FA).
  - **Encaminhamento**: Encaminhamento de mensagens para o agente SNMP, provenientes do manager, assim como no outro sentido a nível de responses e traps.
- **Agent.py**: Agente SNMP, caapaz de:
  - **Interpretação de primitivas SNMP**: Interpretação das primitivas get-request, get-bulk, set-request e adequadas response.
  - **Envio de Notificações**: Notificações geradas em função de alterações de estado significativas de atributos presentes nas MIBs.

2.2 Decisões

De seguida encontram-se enunciadas algumas decisões relevantes tomadas aquando da realização do projecto.

- Ao nível do **agente**, considerei um caso de estudo de um agente SNMP a ser executado num ambiente desktop linux, em que como informações relevantes seriam métricas de hardware tais como: *utilização de cpu, sistema operativo, informações de cpu, ram disponível, uptime, hostname e utilização de disco.*
- Ao nível do proxy, foi definida a **MIBSec** responsável pelo estruturação e gestão das informações relevantes a cada operação SNMP, contendo todos os campos referidos no capítulo seguinte de forma a ser possível gerir os pedidos enviados.
- Ao nível do agente, as notificações enviadas são geradas artificialmente, isto é, são geradas notificações de *spikes* na utilização de disco, cpu ou ram, de forma a simular um cenário real de monitorização de dispositivos de rede.

2.2.1 MIBs

MIBSec

```
MIBSec OBJECT-TYPE
    SYNTAX SEQUENCE OF Linha
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Tabela para guardar resultados das queries dos agentes remotos
        , e respectiva informação de cada pedido"
    ::= { tabelasMIB 1 }

linha OBJECT-TYPE
    SYNTAX SEQUENCE OF Linha
    MAX-ACCESS not-accessible
    STATUS current
    INDEX { idOper }
    DESCRIPTION
        "Linha que descreve uma linha que descreve os dados de um pedido"
    ::= { MIBSecTab 1 }

Linha::=
    SEQUENCE {
        idOper INTEGER
        typeOper INTEGER
        idSource INTEGER
        idDestination OCTET STRING
        oidArg OCTET STRING
        valueArg Opaque
        typeArg INTEGER
        sizeArg INTEGER
        setValueType INTEGER
        setType INTEGER
    }

idOper OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Identificador do pedido efetuado"
    ::= { linha 1 }

typeOper OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Inteiro representativo do tipo do pedido efetuado
        0 para get-request, 1 para get-bulk, 2 para set-requests"
```

```
    ::= { linha 2 }

idSource OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Identificador do agente que efetuou o pedido"
    ::= { linha 3 }

idDestination OBJECT-TYPE
    SYNTAX OCTET STRING
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Identificador do agente que recebeu o pedido"
    ::= { linha 4 }

oidArg  OBJECT-TYPE
    SYNTAX OCTET STRING
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "OID do pedido"
    ::= { linha 5 }

valueArg OBJECT-TYPE
    SYNTAX Opaque
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Valor do pedido em bytes"
    ::= { linha 6 }

typeArg OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Tipo do de dados recebidos como resposta ao pedido"
    ::= { linha 7 }

sizeArg OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Tamanho do valor do pedido"
    ::= { linha 8 }

setValueType OBJECT-TYPE
```

```
    SYNTAX INTEGER
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "String representativa do datatype a atribuir no caso de um set request"
    ::= { linha 9 }

setValue OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "String representativa do valor a atribuir num set request"
    ::= { linha 10 }
```

HWINFO

A seguinte MIB, implementada no agente diz respeito ás informações de hardware a monitorizar.

```
HWINFO OBJECT-TYPE
    SYNTAX SEQUENCE OF Linha
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Tabela para representar dados de hardware do agente"
    ::= { tabelasMIB 1}

Linha OBJECT-TYPE
    SYNTAX SEQUENCE OF Linha
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Linha da tabela de hardware"
    ::= { HWINFO 1 }

Linha::=
    SEQUENCE {
        cpuInfo OCTET STRING,
        currentOS OCTET STRING,
        cpuStatus OCTET STRING,
        RamAvailable OCTET STRING,
        StorageAvailable OCTET STRING,
        upTime OCTET STRING,
        hostname OCTET STRING,
    }

    ::= { Linha 1 }

cpuInfo OBJECT-TYPE
    SYNTAX INTEGER
```

```
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Informação sobre o CPU"
    ::= { Linha 1 }

currentOS OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Informação sobre o sistema operativo"
    ::= { Linha 2 }

cpuStatus OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Informação sobre o status do CPU"
    ::= { Linha 3 }

RamAvailable OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Informação sobre a quantidade de memória disponível"
    ::= { Linha 4 }


StorageAvailable OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Informação sobre a quantidade de espaço disponível no disco"
    ::= { Linha 5 }

upTime OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Informação sobre o uptime do agente"
    ::= { Linha 6 }

hostname OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS read-write
    STATUS current
```

```
DESCRIPTION
    "Informação sobre o hostname do agente"
::= { Linha 7 }
```

2.3 Comunicação e Segurança

A comunicação é realizada, ao nível de transporte, em UDP na porta 161 para get-requests, get-bulks, set-requests e na porta 162 para as notificações. Ao nível de segurança, entre o manager e o proxy, as mensagens são encriptadas via **encriptação assimétrica RSA**, sendo que existe uma **private key** e *public key* para encriptação e desencriptação. Estas chaves são geradas através de um script desenvolvido (*rsaKeysGenerator.py*) sendo lidas pelo manager e proxy para realizar as respetivas operações de encriptação e desencriptação. Adicionalmente, é também utilizado username e password para autenticação acompanhado de um layer adicional de autenticação baseado numa *time-based one-time password* (TOTP), sendo esta gerada através de uma seed associada a esse dado user, podendo ser efetuada a correspondência no manager e proxy.

2.4 Runtime View

1. São executadas as 3 aplicações em 3 terminais distintos.
2. É pedido input ao utilizador no lado do **manager** para introdução do seu username e password, ambos os campos são enviados para o proxy devidamente encriptados, sendo realizada uma terceira comunicação onde é gerado um código 2FA e consequentemente verificado no lado do proxy.
3. **O manager inicia então uma espera ativa por comandos** snmp a introduzir pelo utilizador, assim como uma **thread de escuta de traps**.
4. Aquando do envio de um pedido o **proxy** verifica a sua conformidade a nível de sintaxe e encaminha para o agente.
5. O Agente recolhe os dados respetivos e envia para o proxy.
6. O proxy adiciona uma entrada na MIBsec correspondente ao pedido, e encaminha a resposta para o agente.

2.5 Testes

As aplicações são executadas sem argumentos, sendo a community string e ips a atribuir aos sockets hardcoded, com o seguinte sintaxe.

```
sudo python3 <app>
```

São necessárias permissões de root dado à atribuição de um ip específico, neste caso *127.0.0.4*, *127.0.0.3* e *127.0.0.2* para manager, proxy e agente respetivamente.

De seguida encontram-se algumas capturas de ecrâ exemplificativas dos estágios de execução do programa.

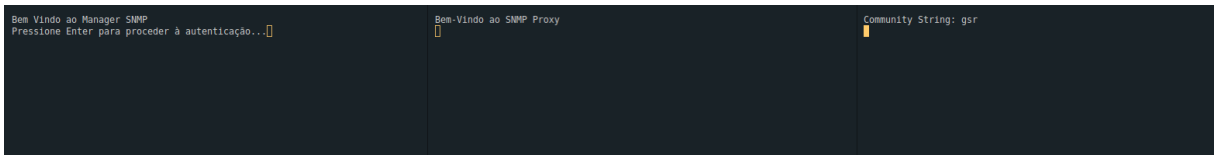

Figura 2.1: Inicialização

Figura 2.2: Espera ativa por comandos.

Figura 2.3: Exemplo de notificações.

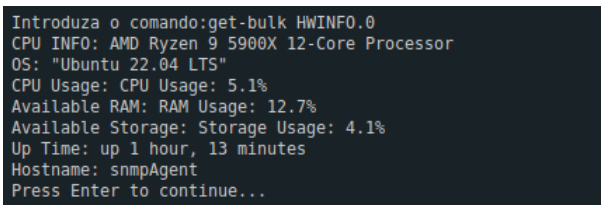

Figura 2.4: Exemplo de get-bulk.

3. Conclusão

Num paradigma em constante evolução ao nível de tipologia de dispositivos ligados a uma rede, assim como quantidade dos mesmos, uma monitorização eficiente é cada vez mais crucial. Este projeto permitiu uma clara compreensão da necessidade da existência de mecanismos de gestão de rede, nomeadamente, a importância do SNMP e utilidade do mesmo. Metodologias de segurança adicionais poderiam ter sido implementadas, no entanto as soluções apresentadas cumprem o propósito de garantir a segurança na comunicação entre o manager e o proxy.

3.1 Trabalho Futuro

Um *refactoring* implementado *design patterns* orientados à melhoria da escalabilidade podem ser implementados, melhorando a robustez e espectro aplicional da solução desenvolvida.