Table of Contents

[Inleiding 2](#_Toc361930544)

[Overzicht applicatie 3](#_Toc361930545)

[Beschrijving modules 4](#_Toc361930546)

[Applicatie installatie 6](#_Toc361930547)

[Configuratie 8](#_Toc361930548)

[Overzicht configuratiebestanden 8](#_Toc361930549)

[Configuratie algemene applicatieconfiguratie 8](#_Toc361930550)

[Configuratie definities 9](#_Toc361930551)

[Aanmaken nieuwe definitie 9](#_Toc361930552)

[Configuratie rulebestanden 14](#_Toc361930553)

[Aanmaken van een rule: 15](#_Toc361930554)

[Aanmaken actiescripts 16](#_Toc361930555)

[Webpagina 17](#_Toc361930556)

[Applicatie starten, stoppen en herstarten 19](#_Toc361930557)

# Inleiding

In het document dat voor u ligt zal de applicatie SecMon worden beschreven. Er wordt in gegaan op de applicatie zelf, hoe deze is opgebouwd met de verschillende modules, hoe de applicatie geconfigureerd en beheerd kan worden.

De applicatie SecMon is een product voortgekomen uit het project Security Monitor. Dit project is onderdeel van de minor Security and Forensics aan de Hogeschool van Amsterdam.

Het doel van het project was het opleveren van een applicatie die de VMWare omgeving van de Hoge school van Amsterdam kan beschermen tegen cyber-aanvallen van zowel binnen uit als buitenaf. De applicatie moet in staat zijn verschillende aanvallen te herkennen en daarop volgend een actie uit te voeren. Dit moet gerealiseerd worden door middel van het parsen van de firewall log(s).

De eisen die worden gesteld aan de applicatie zijn :

* In Python ontwikkeld;
* Compatible is met Python versie 2.7;
* Op een stock Gentoo omgeving moet kunnen draaien;
* IPv4 en IPv6 Compatible;
* Daemonized;
* Automatische continue parsing van iptables-logfiles;
* Configureerbare simpele rulesets voor matching;
* Configureerbare meervoudige acties bij triggering van rulesets;
* Gemakkelijk maar flexibel in het gebruik d.m.v. modules;

# Overzicht applicatie

* **secmon** *(root directory)*
  + **Core** *(core componenten directory)*
    - Config.txt
    - Configuration.py
    - Daemon.py
    - Definition.py
    - FileManager.py
    - Main.py
    - Monitor.py
    - QueryManager.py
    - Rule.py
    - SecMon.py
    - securitymonitor.py
    - SMI.py
    - Trigger.py
  + **Custom** *(gebruikers componenten directory)*
    - **Actions**
      * Email.py
    - **Rules**
      * RuleDefinitionTable.txt
      * rule<naam rule>.txt

# Beschrijving modules

In de core componenten directory ‘secmon/core/’ bevinden de volgende modules:

* Configuration.py
  + Deze module zorgt ervoor dat de configuratieinstellingen en rule bestanden worden ingelezen, zoals: Config.txt en rule\*.txt.
* Daemon.py
  + Deze module zorgt ervoor dat de applicatie daadwerkelijk gestart, gestopt en herstart kan worden als een Disk And Execution MONitor. Deze module wordt door securitymonitor.py aangesproken door wijze van argumenten.
* Definnition.py
  + Deze module zorgt voor het creëren van definities die binnen de applicatie gebruikt kunnen worden. Deze definities zijn *zelf* gedefinieerde (data)typen, zoals: SOURCEIP, TAGRGETIP, DATA etc.
* FileManager.py
  + Deze module zorgt ervoor dat bestanden gelezen kunnen worden en/of iets naar bestanden weggeschreven kan worden.
* Main.py
  + Deze module zorgt ervoor dat bij het vóór het starten van het monitoren de configuratie wordt ingeladen met bijbehorende rules en vervolgens dat log parser en monitor als een thread wordt gestart.
* Monitor.py
  + Deze module zorgt ervoor dat de logfile geparsed wordt, rulesfiles in threads worden gebruikt en wanneer er triggers zijn, met betrekking tot de rule, er dan een actie wordt uitgevoerd.
* QueryManager.py
  + Deze module zorgt ervoor dat rules op basis van query-op-query gematched worden op firewall log. Voor dit proces worden de rules van de rulebestanden gebruikt met daarnaast de gekoppelde keywords die behoren tot de rules.
* Rule.py
  + Deze module zorgt ervoor dat van de rule bestand de waarden en keywords worden ingelezen.
* SecMon.py
  + Deze module zorgt ervoor dat argumenten die door een gebruiker afgevuurd op deze module kunnen worden opgevangen en worden gebruikt. Deze module kan config/rule bestanden wijzigen via Command Line Arguments, bijvoorbeeld om het wachtwoord te wijzigen van een e-mail adres in Config.txt
* Securitymonitor.py
  + Deze module zorgt ervoor dat de applicatie als Daemon kan worden gestart door middel van argumenten. De volgende drie mogelijkheden kunnen worden gebruikt: start, stop of restart. Dit zijn de drie argumenten die een gebruiker kan opgeven om de applicatie te starten, stoppen of te herstarten als Daemon. De opgevangen argument zal vervolgens Daemon.py aaspreken om het opgegeven argument daadwerkelijk uit te voeren.
* SMI.py
  + Deze module zorgt ervoor dat er een interface wordt gemaakt die door andere custom modules gebruikt kunnen worden. Bijvoorbeeld dat email.py via SMI.py de Config.txt kan gebruikt met betrekking to emailadres of wachtwoord.
* Trigger.py
  + Deze module zorgt ervoor dat als er een trigger is de *juiste* actiescript(s) wordt geëxecuteerd met de aanlevering van de bijbehorende argumenten van de rule.

# Applicatie installatie

Benodigdheden:

* Python 2.7
* Directory waar executie mag plaats vinden
* Voor de web interface:
* PHP
* Apache

De applicatie bestaat uit de volgende structuur:

/var/secmon/core

/var/secmon/custom

/var/secmon/custom/rule

/var/secmon/custom/actions

In /secmon/core zitten alle kritische en essentiële bestanden, genaamd:

* Conf.txt
* Configuration.py
* Daemon.py
* Definnition.py
* ErrorHandler.py
* FileManager.py
* Main.py
* Monitor.py
* QueryManager.py
* Rule.py
* SecMon.py
* Securitymonitor.py
* SMI.py
* Trigger.py

In /secmon/custom/rule worden alle rules geplaatst/aangemaakt in plain-text formaat. Het rulebestand moet altijd beginnen met ‘rule’ en daarachter mag de naam opgegeven worden waar de rule voor dient, zoals:

* rule<naam van de rule, zoals: DOS>.txt ***moet worden aangemaakt als***: ruleDOS.txt

Extensie mag ook anders zijn, zolang het universeel en algemeen bekende plain-tekst extensie is.

In /secmon/custom/actions worden alle actions geplaatst die kunnen worden gebruikt als er en trigger heeft plaatsgevonden en die overeenkomen met de actie in een rulebestand. Deze actions zijn simpelweg actiescripts. Verder in de handleiding wordt de term actiescripts gebruikt.

Om de applicatie te installeren moeten de *directories*(core en custom in secmon!) met bijbehorende modules worden gekopieerd naar een plek naar keuze, zoals:

/var/secmon

Kopieer de directory ‘secmon’ met bijbehorende subdirectories ‘secmon/core & secmon/custom/\*’ in de directory ‘/var/’

De structuur ziet er als volgt uit:

/var/secmon/core

/var/secmon/custom

/var/secmon/custom/rule

/var/secmon/custom/actions

Voordat de applicatie gebruikt kan worden moet eerst de Config.txt file worden gewijzigd, deze kan gevonden worden in:

/var/secmon/core

In deze configuratiebestand moet bijvoorbeeld de logDir(firewalllog directory) en ruleDir plus e-mail gegevens in worden gevuld. De configuratie hiervan wordt later behandelt.

# Configuratie

## Overzicht configuratiebestanden

De applicatie kan op elke willekeurige locatie worden geplaatst. In dit voorbeeld gebruiken we:

/var/secmon

Op deze locatie plaatsen we de gehele SRC folder. Deze Folder bevat de mappen Core en Custom met de daaronder liggende bestanden. Een overzicht zoals het in dit voorbeeld te vinden is in /var/secmon/

* **Secmon**
  + **Core**
    - Config.txt
  + **Custom**
    - **Rules**
      * RuleDefinitionTable.txt
      * rule<naamdefinitie>.txt

## Configuratie algemene applicatieconfiguratie

Nadat de applicatie op de server is geplaatst zijn er een aantal punten die aangepast dienen te worden aan de specifieke locatie van applicatie en de modules. De bestanden die aangepast moeten worden zijn:

* Config.txt

In de Config.txt dienen de juiste waardes en paden opgegeven te worden achter de ‘=’ teken. De standaard opmaak ziet er als volgt uit.

// directories of the Files

logDir = [pad naar de firewall log]

ruleDir = [pad naar de rulebestanden]

actionDir = [pad naar actiebestanden]

interval = 30

fromaddr = sendfrom@example.com

toaddr = sendto@example.com

username = [smtpserver username]

password = [password]

server = smtp.server.com:[poort]

Deze waardes kunnen aangepast worden naar eigen configuratie en credentials. Om deze file aan te passen is er een optionele mogelijkheid om gebruik te maken van een webGUI. zie hoofdstuk Webpagina.

## Configuratie definities

RuleDefinitionTable.txt is een configuratiebestand die optioneel is om te wijzigen. Dit bestand zal alleen worden gewijzigd in een situatie waar een nieuwe definitie aangemaakt moet worden. Dit bestand wordt gebruikt als een mapping van sleutelwoorden die te vinden zijn in de firewalllog en die gebruikt kunnen worden in de rulebestanden. Als er een wijziging in RuleDefinitionTable.txt plaatsvindt dan ***moeten*** er ook wijzigingen plaatsvinden in drie andere bestanden, namelijk: Definition.py, QueryManager.py en Trigger.py. Een voorbeeld volgt straks. Dus de nieuwe definitie kan dan gebruikt worden als mapping. RuleDefinitionTable.txt bevat standaard deze opmaak:

DATA = ''

PROTOCOL = 'PROTO='

SOURCEIP = 'SRC='

SOURCEPT = 'SPT='

TARGETIP = 'DST='

TARGETPT = 'DPT='

DATETIME = '\w+\s\d+\s\d+:\d+:\d+'

TIME = '\d+:\d+:\d+'

Het woord *voor* de ‘=’ teken is een definitie die gebruikt kan worden in de rulebestanden voor het definiëren van rules. Het woord/waarde wat *ná* de ‘=’ komt is een sleutelwoord die te vinden is in de firewalllog en deze moet overeenkomen met wat er daadwerkelijk in een firewalllog voorkomt. Anders werkt de mapping niet. De mappings “DATETIME = '\w+\s\d+\s\d+:\d+:\d+'

& TIME = '\d+:\d+:\d+'” worden alleen intern binnen de applicatie gebruikt.

### Aanmaken nieuwe definitie

Voor het definiëren van een nieuwe definitie zijn de volgende bestanden benodigd:

* RuleDefinitionTable.txt: Hierin definieer je de mapping <Nieuwe definitie> = ‘<sleutelwoord firewalllog>’
* Definition.py: Hierin wordt de nieuwe <Nieuwe definitie> van de nieuwe definitie opgeslagen en bekent gemaakt aan de applicatie.
* QueryManager.py: Hierin moet de nieuwe <Nieuwe definitie> aangemaakt worden met een bijbehorende funtie van de nieuwe definitie. Zodat er ook daadwerkelijk iets gedaan wordt met de data van de definitie.
* Trigger.py: Hierin kan er na een trigger van een rulebestand de waarde van de nieuwe definitie worden meegegeven als argument naar actiescripts.

Voorbereiding nieuwe definitie:

In dit voorbeeld gaan we een nieuwe definitie toevoegen om MAC adressen te kunnen gebruiken in rulebestanden. Voor de voorbereiding is het hier onderstaande minimaal nodig:

* Sleutelwoord wat in de firewalllog wordt gebruikt, dat is voor een MAC adres: ‘**MAC=**’
* Nieuwe definitie die als mapping wordt gebruikt. Deze kan gebruikt worden in de rulebestanden. In dit voorbeeld gebruiken we ‘**MAC**’ als mapping.
* Reguliere expressie voor MAC adressen. In dit voorbeeld gebruiken we “*([a-fA-F0-9]{2}[:|\-]?){6}*”
* Definieer *alle* wijzigingen met een nieuwe definitie op een alfabetisch volgorde. Dit met betrekking tot structuur en enumeratie(Definition.py).

Uitvoering aanmaken nieuwe definitie:

1. Open in de directory /var/secmon/custom/rules het bestand ‘RuleDefinitionTable.txt’

2. Voeg de nieuwe definitie MAC toe op alfabetische volgorde. Dat betekent bij dit voorbeeld dat de definitie MAC tussen, DATA = ‘’ ***en*** PROTOCOL = ‘PROTO=’ komt. Bij het toevoegen van de nieuwe definitie MAC heeft dit de volgende structuur en naamgeving:

MAC = ‘MAC=’

Na het toevoegen van de definitie ziet RuleDefinitionTable.txt er nu alsvolgt uit:

DATA = ''

**MAC = 'MAC='**

PROTOCOL = 'PROTO='

SOURCEIP = 'SRC='

SOURCEPT = 'SPT='

TARGETIP = 'DST='

TARGETPT = 'DPT='

DATETIME = '\w+\s\d+\s\d+:\d+:\d+'

TIME = '\d+:\d+:\d+'

3. De mapping is aangemaakt. Nu moet er intern in de applicatie nog gebruik van gemaakt worden. Dit doen we door wijzigingen door te voeren in Definition.py en QueryManager.py. Open Definition.py in de directory /var/secmon/core/.

4. Op regel 14 of onder ‘class Definitions:’ vind je alle definitie namen die bekent zijn binnen de applicatie. Deze zijn op alfabetisch volgorde geplaatst. Aan het einde van de definitielijst staat er range(x). De ‘x’ is het bereik van de aantal definities wat afhankelijk is van de hoeveelheid definities die er zijn opgegeven. In dit geval zijn er momenteel 13 definities en daarom is de range ook range(13). Dit wordt gebruikt omdat er zelf gedefinieerde (data)type worden aangemaakt of anders gezegd een enumaration. De definitielijst ziet er standaard als volgt uit:

ACTION,COUNT,DATA,DATETIME,DESCRIPTION,INTERVAL, NAME,PROTCOL,SOURCEIP,SOURCEPT,TARGETIP,TARGETPT,TIME = range(13)

5. Voeg de nieuwe definitie ‘MAC’ toe aan de hier bovenstaande definitielijst. Dat betekent dat in dit voorbeeld het woord ‘MAC’ toegevoegd moet worden tussen INTERVAL & NAME. Let wel op om ook de range() te verhogen met de hoeveelheid toegevoegde definities. Momenteel is dat maar één definitie en dat betekent dat de range niet 13 blijft, maar 14 wordt. De definitielijst ziet er als volgt uit:

ACTION,COUNT,DATA,DATETIME,DESCRIPTION,INTERVAL,MAC, NAME,PROTCOL,SOURCEIP,SOURCEPT,TARGETIP,TARGETPT,TIME = range(14)

6. De laatste wijziging dat moet plaatsvinden is de definitie leeg declareren wat onderin Definition.py wordt gedaan. Onderin Definition.py staan meerdere definities onder de volgende structuur: Definition.<Definitienaam> ‘’. Voeg ook weer de nieuwe definitie toe op alfabetische volgorde. In dit voorbeeld is dat dit:

Definition.MAC ‘’

7. Binnen de applicatie in nu de definitie MAC bekent en kan worden gebruikt als ‘mapping’. Alhoewel, er nu momenteel nog niks wordt gedaan met de waarde die wordt gekoppeld aan de definitie MAC. Om dit te realiseren moet er in QueryManager één functie worden aangemaakt en ‘***in***’ een functie ‘MAC’ worden toegevoegd. Open QueryManager.py in de directory /var/secmon/core/.

8. Lokaliseer de functie ‘def execute(self, query):’. Daarbinnen vind je een while loop en if statements. Binnen deze functie wordt er gezocht naar definities binnen de rulebestanden. Aangezien er in deze functie nog geen if statements is toegevoegd om de definitie MAC te vinden, wordt er nog niks mee gedaan en moet deze toegevoegd worden. Alle if statements binnen deze functie hebben dezelfde werking, maar zoeken op andere definities. Kopieer één if statement, zoals: COUNT:. Zie hieronder voor het voorbeeld COUNT if statement.

if *self*.current.\_\_contains\_\_(*"COUNT"*):

*self*.getCount()

query = query.replace(*self*.current, *""*)

Plak deze if statement tussen de DATA en PROTO if statements. Vervang het woord COUNT door de definitie(***met double-quotes!***) “MAC =”. Vervolgens moet er in deze if statement ook nog de functie naam worden gewijzigd van self.getCount() naar self.getMAC(). Via de if statement wordt een waarde opgehaald in de rulebestand van de definitie. Het voorbeeld ziet er nu na de wijziging alsvolgt uit:

if *self*.current.\_\_contains\_\_(*"MAC ="*):

*self*.getMAC()

query = query.replace(*self*.current, *""*)

9. In deze stap maken we de functie getMAC() aan, aangezien deze wordt aangeroepen in de bovenstaande if statement. Als deze niet wordt aangemaakt wordt er ook niks gedaan met de waarde bij de bijbehorende definitie. Alle functies van de definities staan ***na/onder*** de functie “def execute(self, query):”. Kopieer in dit voorbeel de functie ‘def getData(self):’ en plak deze tussen de functie ‘def getData(self):’ en ‘def getProto(self):’. Het voorbeeld van de gekopieerde functie:

def **getData**(*self*):

regex = Definitions.getValueDefinition(*"DATA"*)

if *self*.current.\_\_contains\_\_(*"\*"*):

regex = regex + *""*

else:

regex = regex + *self*.current.split(*"="*)[1].strip()

return *self*.executeRegex(regex)

10. In de gekopieerde functie moeten drietal wijzigingen plaatsvinden. Zie de rood gearceerde achtergrond van stap 9. Het gaat om de volgende drie wijzigingen:

* De naam van de functie(def getData(self):),
* string van de definitie(Definitions.getValueDefinition(“DATA”)
* Toevoegen reguliere expressie: regex = regex + ""

11. Wijzig de drie benodigde wijzigingen naar het volgende(gebruik de juiste reguliere expressie, wat bij de voorbereiding werd aangegeven). En sla de wijzigingen op. Zie hieronder voor de gewijzigde functie(rood gearceerd).

def **getMAC**(*self*):

regex = Definitions.getValueDefinition(*"MAC"*)

if *self*.current.\_\_contains\_\_(*"\*"*):

regex = regex + *"([a-fA-F0-9]{2}[:|\-]?){6}"*

else:

regex = regex + *self*.current.split(*"="*)[1].strip()

return *self*.executeRegex(regex)

12. Na de wijziging is het volgende gerealiseerd met de nieuwe functie voor de definitie MAC: de functie haalt de waarde op van de definitie ‘MAC =’ in de vorm van een opgegeven MAC adres of asterix(\*). Via de if statement wordt een reguliere expressie opgehaald als er in de rulebestand een asterix is opgegeven of als deze niet gevonden is wordt er in de else het opgegeven MAC adres opgehaald.

13. Als laatste stap moet Trigger.py worden gewijzigd. Als er een trigger aan de hand van een rule kan de nieuwe definitie worden meegegeven aan een actiescript, maar dat moet eerst aangemaakt worden. Open Trigger.py in de directory /var/secmon/core/.

14. Declareer de nieuwe definitie eerst als leeg door onder de ‘class Trigger:’ de definitie mac = “” toe te voegen bij de andere definities. De declaraties na het toevoegen ziet er als volgt uit:

class **Trigger**:

*'''*

*Trigger:*

*The Trigger class performs actions such as executing modules.*

*After the check which action will be used in the list, a certain amount of parameters can be parsed.*

*These parameters are pre-defined. Which parameters are used can be defined in the module.*

*'''*

action = *""*

data = *""*

description = *""*

**mac = *""***

name = *""*

protocol = *""*

sourceip = *""*

sourcept = *""*

targetip = *""*

targetpt = *""*

current = *""*

15. Deze stap is hetzelfde als stap 8 en 9. Lokaliseer eerst de functie “def ExecuteTrigger(self, query):” en kopieer binnen de while loop *één* if statement. Deze kunnen we hergebruiken. In dit voorbeeld kopiëren we de if statement “ACTION =” en plakken we tussen “Description =” ***en*** “NAME =”.

16. Wijzig de if statement naar het volgende voor de definitie MAC:

if *self*.current.\_\_contains\_\_(*"MAC ="*):

*self*.mac = *self*.current.split(*"="*)[1].strip().replace(*"'"*, *""*).replace(*"="*, *""*).strip()

query = query.replace(*self*.current, *""*)

17. de laatste wijziging wat plaats gaat vinden in Trigger.py is de wijziging om het argument mee tegen aan een actiescript. Lokaliseer onderin Trigger.py “subprocess.call()”. Deze staat er twee keer in en zal ook voor beide gewijzigd moeten worden. In subprocess.call() staan de definities die als argumenten meegegeven worden aan actiescripts. Voeg de definitie MAC toe als self.mac. Voor beide subprocess.call() ziet dat eruit als hieronder:

subprocess.call([sys.executable, path, *self*.action, *self*.data, *self*.description, ***self*.mac**, *self*.name, *self*.protocol, *self*.sourceip, *self*.sourcept, *self*.targetip, *self*.targetpt])

De wijzigingen die nodig zijn voor een nieuwe definitie is doorgevoerd.

## Configuratie rulebestanden

In de directory /var/secmon/custom/rule/ kunnen rulebestanden worden aangemaakt en gewijzigd. In een rulebestand worden sleutelwoorden met een waarden aangemaakt. De meeste sleutelwoorden zijn woorden die ‘gemapped’ zijn aan sleutelwoorden die in de firewalllog teruggevonden kunnen worden, zoals het sleutelwoord: TARGETIP. Deze is gemapped aan het sleutelwoord ‘DST=’ binnen de firewalllog. Sleutelwoorden zoals: NAME, DESCRIPTION, COUNT, TIMER en ACTION zijn geen mappings die gekoppeld zijn aan de firewalllog, maar zijn wel essentieel voor de applicatie. Een rule moet binnen de curly braces ‘{’ & ‘}’ gedefinieerd worden. Hieronder staan de mogelijk sleutelwoorden met opmaak:

{

NAME = ‘’

DESCRIPTION = ‘’

SOURCEIP =

SOURCEPT =

TARGETIP =

TARGETPT =

PROTOCOL =

DATA =

COUNT [<,<=,>,>=,=]

TIMER [<,<=,>,>=,=]

INTERVAL = hh:mm:ss

ACTION = ‘’

}

Beschrijving:

|  |  |  |
| --- | --- | --- |
| Sleutelwoord of  startstop markering | Gebruikte of beschikbare  Operator | Uitleg |
| { |  | Start markering van de rule |
| NAME  DESCRIPTION  SOURCEIP  SOURCEPT  TARGETIP  TARGETPT  PROTOCOL  DATA  COUNT  TIMER  INTERVAL  ACTION | =  =  =  =  =  =  =  [<,<=,>,>=,=]  [<,<=,>,>=,=]  hh:mm:ss  = | Hier kan binnen de ‘’ een naam opgegeven worden van de rule.  Hier kan binnen de ‘’ een beschrijving opgegeven worden van de rule.  Hier kan een source IP adres opgegeven of asterix(alle IP’s) opgeven.  Hier kan een source poort opgegeven of asterix(alle poorten) opgeven.  Hier kan een target IP adres opgegeven of asterix(alle IP’s) opgeven.  Hier kan een target poort opgegeven of asterix(alle poorten) opgeven.  Hier kan een protocol, zoals: UDP of TCP opgegeven worden.  Hier kan een packet teller opgegeven worden. *Functie Afh. van operator!*  Hier kan een tijdbereik opgegeven worden. *Functie Afh. van operator!*  Hier kan een hh:mm:ss worden opgegeven. *Alleen m.b.t. slow attacks!*  Hier kan binnen de ‘’ een of meerdere actiescript(s) worden opgegeven. |
| } |  | Einde markering van de rule |

### Aanmaken van een rule:

Rules kunnen worden aangemaakt in plain-text formaat onder de locatie /var/secmon/custom/rules/.

Als een nieuwe rulebestand wordt aangemaakt is er een minimale voorwaarde waar de naamgeving aan moet voldoen. De naam van de rulebestand moet beginnen met kleine letters ‘rule’. Wat achter de ‘rule’ komt mag zelf gedefinieerd worden, zoals: ruleDoS, ruleTCP, ruleUDP etc. Als voorbeeld maken we twee soorten rulebestanden aan onder de naam ruleTest.txt en ruleLAND.txt. In deze plain-text bestand kunnen de sleutelwoorden gebruikt worden die in het kopje ‘Configureren en aanmaken rulebestanden’ worden beschreven.

Een rule hoeft niet zozeer *alle* beschikbare sleutelwoorden te gebruiken, dit omdat voor de ene rule niet specifiek op dat gene hoeft gezocht te worden en in de andere rule wel. Die sleutelwoorden hoeven dan niet tussen de start en eind markering gezet te worden(‘{‘ & ‘}’).

De opzet en invulling van de onderstaande rule “ruleTEST.txt” is als volgt:

“”

ruleTest.txt

{

NAME = ‘Test rule’

DESCRIPTION = ‘This rule is a test which checks….’

SOURCEIP = 192.168.1.14

SOURCEPT = 22

TARGETIP = 192.168.1.33

TARGETPT = 443

PROTOCOL = TCP

DATA = SYN

COUNT > 400

TIMER <= 10

ACTION = ‘Email.py’

}

De opzet en invulling van de onderstaande rule “ruleLAND.txt” is als volgt:

“”

ruleLAND.txt

{

NAME = ‘LAND attack rule’

DESCRIPTION = ‘LAND rule that checks the equality of source and target IP’

SOURCEIP = \*

TARGETIP = SOURCEIP

PROTOCOL = TCP

DATA = SYN

INTERVAL = 02:00:00

ACTION = ‘Email.py’

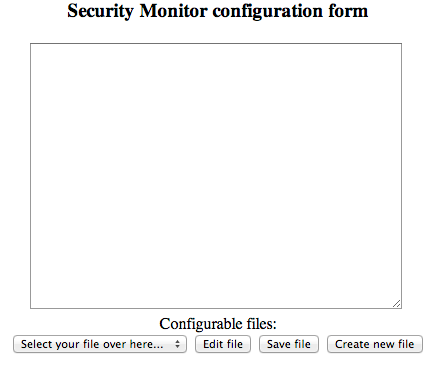
}

## Aanmaken actiescripts

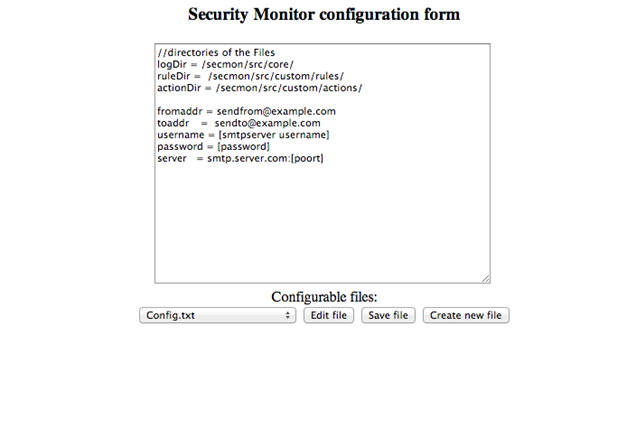
Actiescripts kunnen naar eigensmaak worden gemaakt bijvoorbeeld in Pyhton. In Trigger.py worden argumenten meegegeven aan actiescripts die staan opgegeven in een een rulebestand. In de standaardlevering van de applicatie is een email notificatie script meegeleverd. Deze is te vinden onder /var/secmon/custom/actions. Binnen deze directory kunnen meerdere actiescripts worden aangemaakt, vergeet niet om deze op te geven in een rulebestand anders werkt de actiescript niet. Om de actiescript op te geven in een rulebestand moet er achter het sleutelwoord ACTION = en binnen de single quotes de naam + extensie van de actiescript worden opgegeven. Bijvoorbeeld : ACTION = ‘Email.py’ of ACTION = ‘Email.py, test.test’. Meerdere opgegeven actiescripts moeten worden gescheiden met een komma.

# Webpagina

Met de levering van de applicatie wordt er ook een webform meegeleverd geschreven in PHP. Via dit webform kunnen de configuratie files worden aangepast. Als de directory wijzigt, moet in het webform ook de directory/pad worden gewijzigd, dit wordt opgegeven achter de variabele **$rootDir**.



Via het dropdown menu “ select your file over here” kan de Config.txt gekozen worden. Klik edit file en het bestand word geopend in het tekstvak.



De waardes in de configuratie kunnen nu aangepast worden. Zodra dit voltooid is dient er op “Save File” geklikt te worden om de nieuwe configuratie op te slaan.

De webGui is opgebouwd in PHP en heeft ook apache in combinatie met de PHP nodig om te kunnen werken.

# Applicatie starten, stoppen en herstarten

Het uitvoeren van het onderstaande vereist minimaal zonder enige toepassingen/wijzigingen dat de gebruiker in de directory zit waar het bestand ‘securitymonitor.py’ zich ook bevindt. Dit wordt gerealiseerd door de cd command, die hieronder staat opgegeven.

Het starten van de applicatie kan door middel van het onderstaande worden uitgevoerd:

* + - cd /var/secmon/core
    - securitymonitor.py start

Het stoppen van de applicatie wordt via de volgende manier uitgevoerd:

* + - cd /var/secmon/core
    - securitymonitor.py stop

Het herstarten van de applicatie wordt via de volgende manier uitgevoerd:

* + - cd /var/secmon/core
    - securitymonitor.py restart