UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA


# Indicadores de Segurança em Plataformas de Monitorização

Raimundo Henrique da Silva Chipongue

**Mestrado em Segurança Informática**

Dissertação orientada por:
Prof. Doutor Hugo Alexandre Tavares Miranda
e co-orientada pelo Mestre António José Carvalho Broega

2017

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA


# Indicadores de Segurança em Plataformas de Monitorização

Raimundo Henrique da Silva Chipongue

**Mestrado em Segurança Informática**

Dissertação orientada por:
Prof. Doutor Hugo Alexandre Tavares Miranda
e co-orientada pelo Mestre António José Carvalho Broega

# Agradecimentos

Ao *Prof. Doutor Hugo Alexandre Tavares Miranda*, a minha vénia e o meu mais profundo reconhecimento pela oportunidade de realizar este trabalho ao lado de alguém que "transpira"sabedoria. O meu respeito e admiração pela sua serenidade, pela capacidade de persuadir, pela capacidade de incentivar, pela disponibilidade e pelo seu Dom no ensino da Ciência, inibindo sempre a vaidade em prol da simplicidade e eficiência. Levo valiosos ensinamentos que seguramente servirão para o resto da minha vida académica e profissional.

Os meus agradecimentos vão também ao meu co-orientador *Mestre António José Carvalho Broega* por toda paciência, colaboração, contribuição e disponibilidade.

À *Prof.ª Doutora Maria Dulce Pedroso Domingos*, por todo apoio, paciência, disponibilidade e dedicação.

À minha esposa *Suweli Chipongue*, mulher da minha vida, pelo apoio incondicional em todos os momentos, principalmente nos de incerteza, muito comuns para quem tenta trilhar novos caminhos, ainda por cima distante da maior parte da família, por acreditar as vezes mais do que eu que os nossos sonhos serão sempre realizados com maior ou menor dificuldade.

Aos meus filhos *Geovana Gabriela, Maria Helena e Rui Ezequiel*, com maior ênfase a pequena Maria Helena que de forma inocente e em idade de total fragilidade se viu afastada da convivência do pai por longos 2 anos.

Aos meus pais *Julieta e Ezequiel Henrique*, que dignamente souberam transmitir seus princípios, fazendo valer a importância da família e do trabalho com honestidade e persistência para alcançar "um a um"os objetivos traçados.

Aos meus professores e colegas por todo apoio e disponibilidade que demonstraram ao longo destes 2 anos.

Aos meus colegas e amigos Luís Ferreira, Pedro Chaves e Sérgio Pinto, que com sabedoria souberam acolher-me.

Os meus agradecimentos são também extensivos àqueles que foram e continuam a ser o meu suporte diário "minha família", avós, tios, irmãos, primos, cunhados e sobrinhos. Em especial os meus cunhados *Rosa e Mauro Silva*, a minha mãe/sogra *Elisa Silva*, a minha irmã *Rosália Chipati*, os meus padrinhos *Marinela* e *Jorge Abreu*, ao meu irmão e companheiro de longa data *André Satumbo*, a minha "encarregada de educação"*Helga Silva*, os meus compadres *Endira e Oliveira Silva*, o meu primo *Feliciano Somavie*, a minha tia *Maria Helena* e a minha amiga *Francisca Bentral*.

*Este trabalho é dedicado a minha família, por estarem sempre ao meu lado, nos bons e maus momentos, por serem parte dos meus sonhos, por me ajudarem a realizar mais um sonho, pelo apoio que cada um a seu jeito me têm dado. Porém, entre todos importa destacar a minha esposa Suweli, os meus pais Julieta e Henrique, o meu irmão Vladmiro Chipongue, por todo o apoio, compreensão e paciência evidenciado em todos momentos da minha vida. Aos meus filhos* **Geovana, Arlene e Rui.**

# Resumo

A monitorização é um mecanismo preventivo e de controlo eficaz para detetar e resolver problemas, melhorando o desempenho e a disponibilidade dos sistemas e serviços. A notável separação entre administradores de sistemas e especialistas de segurança, aliada à remodelação dos sistemas de monitorização da Direção de Serviços Informáticos da Faculdade de Ciências, e a pouca oferta de módulos com foco em eventos de segurança no repositório oficial do Nagios, foram alguns dos aspetos que motivaram este trabalho. O Objetivo é proporcionar aos intervenientes um único ambiente para visualização dos eventos de administração de sistemas em geral, e eventos de segurança em particular.

Nesta dissertação discutimos o uso das ferramentas de monitorização no incremento da segurança das infraestruturas tecnológicas, através da monitorização 24x7 dos sistemas e serviços em busca de vulnerabilidades conhecidas e comportamentos anómalos. O Nagios foi a ferramenta de eleição e foram desenvolvidos módulos com foco na segurança, que depois de configurados, testados e instalados em ambiente de produção, permitiram uma avaliação de resultados e de desempenho. São também discutidos aspetos de segurança relacionados ao sistema de monitorização Nagios, resultando em recomendações que minimizam problemas de segurança a que é suscetível.

Os resultados obtidos fazem-nos concluir que é possível usar a flexibilidade destas ferramentas para monitorização de sistemas e serviços em busca de comportamentos anómalos e vulnerabilidades de segurança conhecidas, gerando indicadores de segurança e notificações em tempo real.

**Palavras-chave:** Monitorização, Segurança, Nagios, Plugins, Alertas.

# Abstract

Monitoring is an effective mechanism for detecting and solving various problems in IT, and improving the performance and availability of systems and services. The remarkable separation between system administrators and security specialists, coupled with the redesign of the monitoring infrastruture of the Directorate of Informatics Services of the Faculty of Sciences and the limited supply of security-focused modules in the official Nagios repository were some of the incentives that led us to this work. The goal is to provide for users a single environment for viewing system administration and security events in particular.

Moreover, in this article we discuss the usage of these tools in increasing the security of technological infrastructures through monitoring systems and services in search of known vulnerabilities and anomalous behaviors. Nagios was the tool of choice; this work discribes several, Nagios modules developed with security features, which after being configured, tested and installed in a production environment, allowed an evaluation of results and performance. Nagios related security issues are also discussed, resulting in recommendations that mitigate security issues to which it is susceptible.

Thus, the results shows that it is possible to use the flexibility of these tools to monitor systems and services for abnormal behaviors and known security vulnerabilities, generating security indicators and notifications in real time.

**Keywords:** Monitoring, Security, Nagios, Plugins, Alerts.

# Índice

# Lista de Figuras

# Lista de Abreviaturas

**CPU** ..... Central Processor Unit

**RAM** ..... Random-Access Memory

**LTS** ..... Long Term Support

**Ciências** ..... Faculdade de Ciências da Universidade de Lisboa

**DSI** ..... Direcção de Serviços Informáticos

**HTTP** ..... Hypertext Transfer Protocol

**SMTP** ..... Simple Mail Transfer Protocol

**POP3** ..... Post Office Protocol 3

**FTP** ..... File Transfer Protocol

**SNMP** ..... Simple Network Management Protocol

**NRPE** ..... Nagios Remote Plugin Executor

**DI** ..... Departamento de Informática

**IDS** ..... Intrusion Detection System

**IP** ..... Internet Protocol

**VoIP** ..... Voice over IP

**GPL** ..... General Public License

**TI** ..... Tecnologia de Informação

**SSH** ..... Secure Shell

**SSL** ..... Secure Sockets Layer

**TLS** ..... Transport Layer Security

**CA** ..... Certificate Authority

**DNS** ..... Domain Name System

**DoS** . . . . . Denial of Service

**DDoS** . . . . . Distribute Denial of Service

**API** . . . . . Application Programming Interface

**NSCA** . . . . . Nagios Service Check Acceptor

**ID** . . . . . Identity

**OWASP** . . . . . Open Web Application Security Project

**MITM** . . . . . Man-In-The-Middle

**BIND9** . . . . . Berkeley Internet Name Domain - Version 9

**XML** . . . . . eXtensible Markup Language

**CMS** . . . . . Content Management System

**CGI** . . . . . Common Gateway Interface

**SLA** . . . . . Services Level Agreement

# Capítulo 1

# Introdução

A relevância e sensibilidade da informação armazenada e manipulada nos sistemas informáticos, faz com que uma das principais prioridades das organizações seja manter as redes e sistemas tão seguros quanto possível. Este problema tem ligação com as necessidades de supervisionar a disponibilidade e desempenho dos sistemas, uma vez que a sua segurança é um dos requisitos que devem ser assegurados.

Em redes corporativas, em que o número de equipamentos conectados pode superar as centenas, a tarefa dos administradores torna-se bastante árdua, o que pode levar à morosa deteção dos problemas, suas causas, ponto exato da ocorrência e consequente atraso na resolução.

Os sistemas de monitorização são ferramentas importantes na deteção de comportamentos fora dos padrões previamente estabelecidos, gerando notificações em tempo real e desta forma contribuindo para acelerar a deteção e resolução de problemas nos sistemas informáticos.

Estando essas ferramentas disponíveis para administradores de sistemas e redes, surge a oportunidade de usá-las para melhorar a segurança dos sistemas informáticos, não com o objetivo de substituir as atuais ferramentas de Deteção de Intrusão (*Intrusion Detection System*, IDS), mas com a intuito de proporcionar aos administradores a possibilidade de terem num único ambiente alguns indicadores sobre a segurança das suas infraestruturas.

Para este efeito, propõe-se a configuração de um sistema de monitorização, que complemente a habitual geração de alertas sobre o funcionamento dos serviços, servidores e equipamentos de rede, com alertas sobre problemas de segurança, através do desenvolvimento de módulos focados na segurança.

## 1.1 Motivação

Culturalmente, tem-se vindo a verificar uma separação entre administradores de TI e técnicos de segurança ou, nalguns casos, a um isolamento das funções dos segundos apenas numa das áreas cobertas pelos primeiros. Esta separação, apesar de compreensível do ponto de vista organizacional, não é de todo benéfica, e pode levar a que um mesmo problema no sistema tenha duas interpretações contraditórias, possivelmente levando à tomada de ações contraproducentes. Por exemplo, apesar de se tratar de uma ação maliciosa, a ocorrência de um ataque de negação de serviço pode ser interpretado inicialmente como uma limitação de desempenho ou perda de disponibilidade pelo administrador do

sistema.

Este problema em muitos casos contribui para o atraso na deteção de problemas nos sistemas e a sua resolução em tempo útil.

Por outro lado o facto de termos notado que em repositórios de algumas das mais populares plataformas de monitorização, como por exemplo, o Nagios Exchange[1], os indicadores de segurança são abordados de forma relativamente marginal, foi outro dos incentivos que levaram a que esta pesquisa fosse realizada.

Acrescentou-se o facto de ter coincidido com a reformulação do sistema de monitorização de redes e serviços da Direção de Serviços Informáticos da Faculdade de Ciências (DSI-Ciências), que foi uma oportunidade para aplicar este modelo através do desenvolvimento de módulos essencialmente orientados para a geração de indicadores de segurança, visando garantir maior confiabilidade dos seus sistemas e consequentemente dos seus serviços.

Não menos importante é a crença de que no final desta dissertação, o autor adquiriu melhor e maior perceção do tema do trabalho.

## 1.2 Objetivos

Com este estudo pretendemos contribuir para aproximação de dois grupos de técnicos fundamentais na administração de sistemas, nomeadamente os administradores de TI e técnicos de segurança, ao fomentar que os administradores de sistemas e especialistas de segurança tenham uma visão única do sistema através do uso da plataforma de monitorização Nagios.

Com o uso do Nagios como plataforma de monitorização pretendemos também facilitar uma análise cruzada da informação, melhorando a colaboração e consequentemente incrementar a segurança e eficiência dos sistemas.

Para isso o trabalho passa por identificar, criar e implementar indicadores de segurança para a plataforma de monitorização Nagios. Estes indicadores irão procurar vulnerabilidades conhecidas e comportamentos ou estados que possam indicar a ocorrência de problemas de segurança.

Com este trabalho pretende-se demonstrar o quão importantes podem ser as plataformas de monitorização na melhoria não só da administração de redes e sistemas, mas também no incremento da segurança.

Augura-se também contribuir para o aumento da qualidade e quantidade de plugins com foco na segurança disponibilizados no repositório Nagios Exchange[2], assim como vaticina-se que a implementação em ambiente de produção em Ciências possa melhorar a proteção das suas infraestruturas.

## 1.3 Principais contribuições

Esta dissertação apresenta a nossa contribuição para a administração da segurança em redes e sistemas informáticos, com particular realce para o sistema de monitorização de

[1] `https://exchange.nagios.org`

[2] `https://exchange.nagios.org/directory/Plugins/Security/`

Ciências e para a comunidade de desenvolvimento da plataforma Nagios. A seguir resumimos as principais contribuições, que são o resultado do cumprimento dos objetivos definidos na secção anterior:

**Aproximação dos administradores de TI e técnicos de segurança:** este estudo contribui para aproximar as duas realidades distintas entre administradores de TI e técnicos de segurança, ao fomentar que os administradores de sistemas e especialistas de segurança tenham uma visão única do sistema através da plataforma de monitorização Nagios. Esta aproximação permite uma análise cruzada de dados, melhora a colaboração entre as duas áreas e consequentemente incrementa a segurança e eficiência dos sistemas.

**Comunidade de desenvolvimento da plataforma Nagios:** com a elaboração desta dissertação foram criados um conjunto de módulos relacionados com aspetos práticos da administração de segurança de uma organização. Estes módulos foram recentemente submetidos e aprovados pelos editores do Nagios Exchange e contribuem para incrementar a qualidade e quantidade dos plugins que monitorizam eventos relacionados com segurança.

**Indicadores de segurança para a plataforma de monitorização de Ciências:** o desenvolvimento de módulos com caraterísticas de segurança e consequente incorporação na plataforma de monitorização de Ciências, permitiu que se passasse a ter alguns indicadores de segurança nesta plataforma, contribuindo desta forma para a segurança dos sistemas de Ciências, sob administração da Direção de Serviços Informáticos (DSI).

**Importância das ferramentas de monitorização para a segurança:** esta dissertação contribui também para destacar a importância destas ferramentas no que a segurança de redes e sistemas diz respeito, usando a sua flexibilidade no desenvolvimento de módulos focados na monitorização de comportamentos e eventos que sinalizem problemas de segurança.

## 1.4 Hardwares e Softwares utilizados

### 1.4.1 Hardware

Para o desenvolvimento e teste deste trabalho, foram utilizados 6 computadores da fabricante DELL, instalados no laboratório de redes do Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa (DI-Ciências), com processadores Intel(R) Core(TM)2 Duo CPU E7500 @2.93GHz, HD ATA WDC WD2500AAJS-6 de 250 Gigabytes e 2 Gigabytes de memoria RAM.

Adicionalmente, e já numa fase avançada do trabalho, foram usados 4 switchs, sendo dois Northel Business Policy Switch 2000, um HP Procurve 2545, e um 3Com SuperStack II PS Hub 40. Foram igualmente usados três encaminhadores, dos quais dois Cisco 2600 e um Cisco 1800.

### 1.4.2 Software

Os computadores acima descritos foram equipadas com Sistema Operativo Linux Ubuntu 16.04.1 LTS. Uma vez que o trabalho realizado foi desenvolvido utilizando a linguagem de programação Python, não se prevêm limitações ao sistema Operativo utilizado, sendo necessário garantir a disponibilidade das bibliotecas e do compilador adequado para a correta execução dos módulos.

Foram configurados em alguns deles programas adicionais, imprescindíveis para desenvolver e testar partes do trabalho, dos quais destacamos o Nagios, Wordpress, Python3, Python-idle3, Bind9, HAProxy, Nagios-nrpe-server, e que em seguida se introduzem brevemente.

**Nagios** É uma ferramenta de monitorização que se destaca por ser de código aberto, e pela forma como permite estender a monitorização aos mais variados equipamentos e seus respetivos serviços. Esta poderosa ferramenta é descrita de forma detalhada na Seção 3.8.

**WordPress** [3] É um sistema para administração de conteúdo (*Content Management System*, CMS) escrito em PHP, e concebido principalmente para a criação de sites e blogues via web. É uma das mais populares ferramentas para criação de blogues, e é descrito com maior detalhe na Seção 4.12.

**Python** [4] É uma linguagem de programação criada por Guido Van Rossum em 1991, orientada a objetos de alto nível, e com um grau de abstração relativamente elevado, muito mais próximo à linguagem humana e longe do código máquina. A sua versão mais recente é a 3.6.1, disponibilizada a 21 de Março de 2017.

**Idle3-tools** [5] É um ambiente de desenvolvimento integrado para Python. É disponibilizado em cada versão do Python, e neste caso o idle3 foi lançado com o Python3. No entanto, não vem incluído no pacote Python de muitas distribuições Linux. É completamente escrito em Python e com o kit de ferramentas de GUI Tkinter (funções de empacotamento para Tcl/Tk).

**Nagios-nrpe-server** É um agente Nagios que permite a execução remota dos plugins em maquinas com sistema operativo Linux, mediante chamada ao plugin check_nrpe, executado no servidor de monitorização Nagios, é descrito com maior detalhe na Seção 3.8.3.

**Bind9** É um servidor para o protocolo DNS, atualmente é um dos mais usados em sistemas operativos Unix.

**HAProxy** É uma ferramenta que de entre outras funções proporciona o balanceamento de carga em servidores web, e contribui para a alta disponibilidade. Adicionalmente evita a exposição direta dos servidores, uma vez que as requisições são feitas ao servidor HAProxy, e este trata de direcionar os pedidos aos servidores com base nas suas disponibilidades.

[3] `https://wordpress.org/`
[4] `https://www.python.org/downloads/`
[5] `https://github.com/bountin/idle3-tools`

Os equipamentos e softwares acima mencionados permitiram a configuração de uma rede de testes, representada na Figura 1.1, que simula um ambiente real, o que permitiu que os testes estivessem mais próximos da realidade de um ambiente de produção.

## 1.5 Estrutura da dissertação

Esta dissertação encontra-se dividida em seis capítulos a seguir descritos:

**Capítulo 1** É feita uma introdução à dissertação, onde são descritas nomeadamente a motivação, os objetivos, o hardware e software utilizado, o cronograma das atividades, e o caso de estudo.

**Capítulo 2** Neste capítulo é apresentada a Faculdade de Ciências da Universidade de Lisboa, assim como a sua Direção de Serviços Informáticos (DSI), na qualidade de Instituição onde o trabalho foi desenvolvido e implementado.

**Capítulo 3** São aqui abordados alguns trabalhos relacionados ao tema, pelo que são mencionados alguns artigos de investigação que tiveram como foco as ferramentas de monitorização em geral ou o Nagios em particular.

É também neste capítulo que descrevemos algumas ferramentas de monitorização conhecidas, com maior ênfase ao Nagios, seu uso para melhorar a segurança dos sistemas, plugins, requisitos de instalação e configuração, arquitetura, categorização dos equipamentos, funcionalidades, funcionamento, configuração, vantagens/-limitações e implementação na Direção de Serviços Informáticos da Faculdade de Ciências da Universidade de Lisboa.

**Capítulo 4** Aqui são apresentados e detalhados os módulos desenvolvidos, suas características, seu uso e problemas de segurança mitigados.

**Capítulo 5** Neste capítulo são apresentados os detalhes da passagem dos módulos para ambiente de produção, assim como os resultados dos testes e das avaliações efetuadas. São também discutidos alguns aspetos de segurança relacionados com o uso do Nagios.

**Capítulo 6** Este é o capítulo reservado às conclusões obtidas com a realização da dissertação, e ao trabalho que futuramente pode ser desenvolvido com o propósito de dar seguimento a esta pesquisa.

## 1.6 Cronograma

Para elaboração desta dissertação, foi definido o cronograma abaixo descrito, com tarefas a realizar durante nove meses. A figura Figura 1.2 representa o desenvolvimento das atividades e cumprimento das tarefas ao longo do período programado.

**Outubro e Novembro** O Desenvolvimento do trabalho teve início a 1 de Outubro. Estes dois primeiros meses estiveram reservados à familiarização com a plataforma

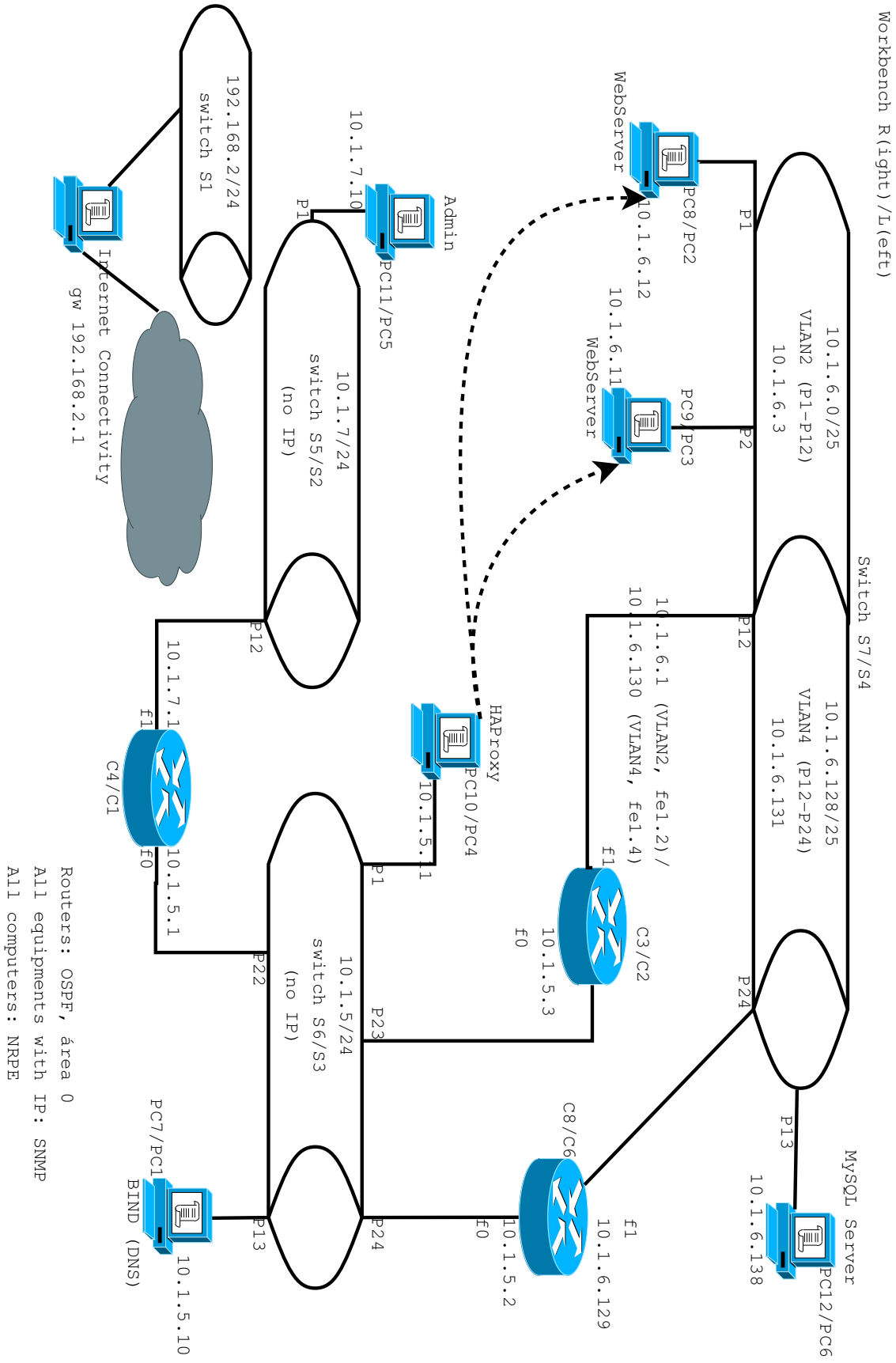


Fig. 1.1: Mapa da rede configurada para testes

de monitorização Nagios, o que consistiu não só na identificação e leitura da documentação oficial, mas também na instalação e configuração de um ambiente de desenvolvimento e testes.

Foi também neste período que teve início a elaboração do relatório preliminar, e o levantamento de requisitos, nomeadamente o conjunto de características de segurança que seriam desejavelmente incorporadas na plataforma de monitorização.

**Dezembro** O mês de Dezembro foi reservado à conclusão e submissão do relatório preliminar. Neste período deu-se sequência ao levantamento de requisitos e conceção da lista de módulos a desenvolver, detalhados na Subseção 1.7.3.

**Janeiro a Março** Os meses de Janeiro, Fevereiro e Março foram reservados ao desenvolvimento dos módulos identificados nas fases anteriores, e realização de testes em ambiente de desenvolvimento.

**Abril e Maio** Os módulos concebidos nas fases anteriores, foram instalados, configurados e avaliados em ambiente de produção, depuração do código e submissão à plataforma Nagios Exchange.

**Junho** O mês de Junho destinou-se à escrita da dissertação e do artigo a submeter à conferência cientifica.

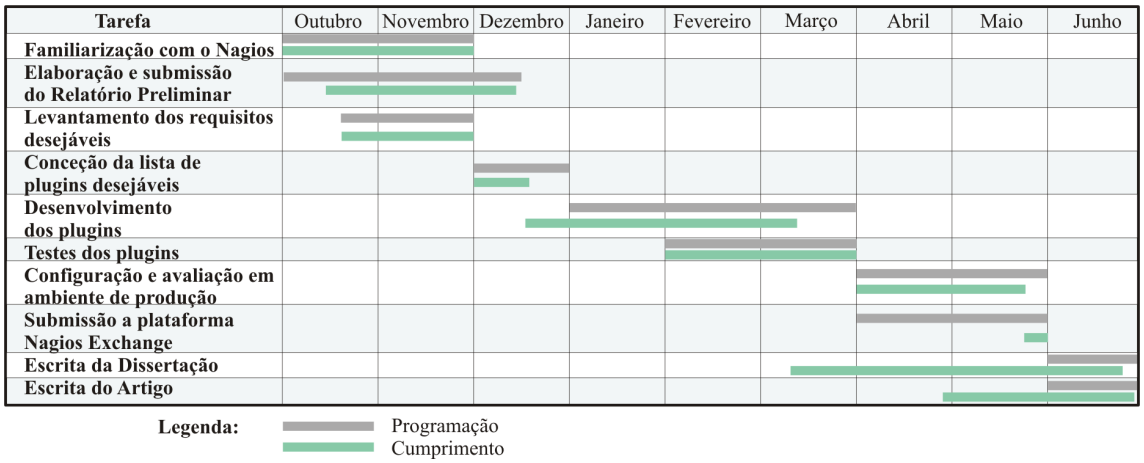


Fig. 1.2: Cronograma de atividades

## 1.7 Caso de Estudo

### 1.7.1 Familiarização com o Nagios

Seguindo o plano de trabalho, nos meses de Outubro e Novembro, a missão foi realizar estudos sobre a plataforma de monitorização Nagios, visando melhorar a perceção das suas características e do seu modo de funcionamento.

O processo de configuração e compreensão do Nagios foi efetuado usando um ambiente de testes, compreendido num conjunto de maquinas virtuais e físicas, instaladas em

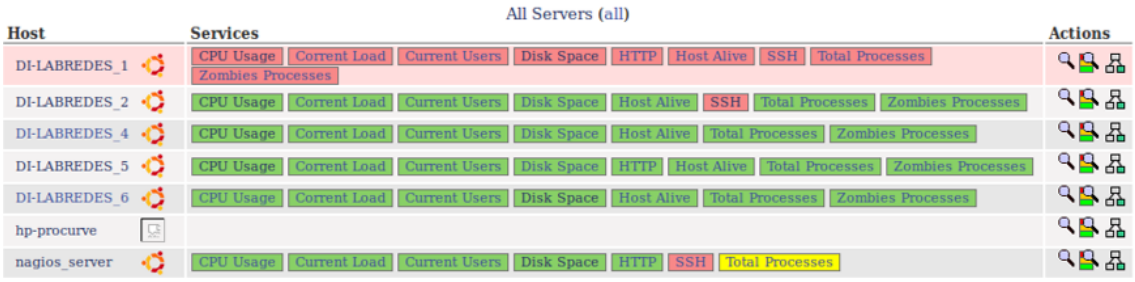


Fig. 1.3: Interface web do Nagios, com serviços, servidores e equipamentos de rede monitorados

dois computadores portáteis, tendo sido instalado e configurado o Nagios em uma dessas maquinas, e posteriormente configurados alguns plugins já existentes na plataforma Nagios Exchange[6], considerada o repositório oficial dos plugins do Nagios.

Posteriormente, após estarem reunidas as condições necessárias, foi configurado um ambiente de desenvolvimento e testes no laboratório de redes do Departamento de Informática (DI), usando os hardwares e softwares mencionados na Seção 1.4.

### 1.7.2 Configuração do ambiente de desenvolvimento e testes

Nesta fase foram configurados 6 computadores para o desenvolvimento e testes dos módulos, tendo sido instalados nestes equipamentos aplicações necessárias para a empreitada a que nos propusemos.

A Figura 1.3 representa a interface web do Nagios com os seis equipamentos mencionados anteriormente a serem monitorizados, enquanto que a Figura 1.4 representa a interface web do Nagios com o mapa das mesmas maquinas. Nesta fase foram instalados e configurados alguns plugins para monitorizar serviços, servidores e equipamentos de rede.

### 1.7.3 Identificação dos módulos a desenvolver

Neste período, a par da familiarização com o Nagios, houve a necessidade de perceber os tipos de serviços prestados e administrados pela Direção de Serviços Informáticos (DSI), o que permitiu identificar os seguintes potenciais módulos a desenvolver:

**DoS ou DDoS** Alerta do rápido sobrecarregamento de um serviço, resultante de várias tentativas de utilização dos serviços a partir de um ou múltiplos computadores;

**Desfiguração de paginas web (Defacement)** Geração de alertas aquando da alteração intencional e maliciosa do conteúdo de páginas na Internet, verificando a existência de palavras chave tipicamente associadas a este tipo de ataques;

[6] `https://exchange.nagios.org`

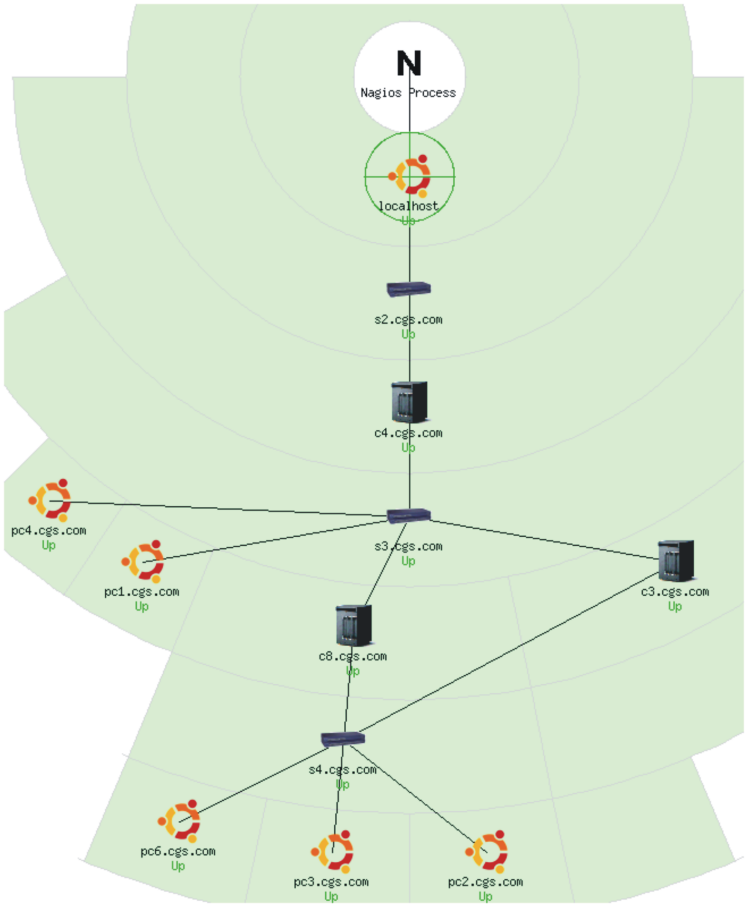


Fig. 1.4: Interface web do Nagios, com o mapa dos servidores e equipamentos de rede monitorados

**Verificação de portos** Controlar portos abertos e alertar sempre que for verificada a abertura de um ou mais portos não autorizados. Para tal, o módulo deverá criar uma lista branca de portos, e sempre que se verificar um porto aberto que não conste da referida lista será gerada a respetiva notificação;

**Deteção de *portscanning*** Controlar e alertar sempre que se verifique um *scanner* na maquina monitorizada, implicando tentativa de descoberta de portos abertos, serviços e versões em uso;

**Verificação de atualizações do WordPress** Criação de um módulo capaz de verificar a versão do WordPress em uso, compará-la com a última versão disponibilizada e alertar os administradores por meio de notificações predefinidas, caso a versão em uso não seja a mais recente;

**Falhas de autenticação** Verificar e alertar sempre que se registar um determinado número de tentativas mal sucedidas de autenticação SSH;

**Testes às configurações SSL** Verificar a validade dos certificados SSL/TLS, e recorrendo a *web services*, verificar um conjunto de padrões, configurações e qualidade dos certificados, capazes de ter implicação na segurança das comunicações;

**DNS** Verificação da alteração na correspondência entre nomes de domínios e endereço IP;

**Controlo das aplicações instaladas** Verificar a lista de aplicações instaladas e alertar sempre que nova aplicação for instalada;

**Ficheiros de passwords** Verificar alterações no ficheiro de passwords;

**Programas em *background*** Verificar e listar programas executados em *background*, alertando sempre que sejam programas desconhecidos, ou quando estes estejam a consumir memória excessiva;

**Testar configurações e validade do DNSSEC** Verificar se um domínio usa a extensão de segurança de DNS, assim como suas configurações e validade;

**Tentativas de acesso à conteúdo privado no servidor Apache** Alertar sempre que se registe elevado número de respostas com código 404, como resultado de requisição por um ou vários endereços IP;

**Alteração do estado de ficheiros** Monitorizar o estado de ficheiros e diretorias, alertando sempre que se verifique alterações;

**Verificação de vulnerabilidades web** Desenvolver um modulo capaz de realizar testes de segurança em domínios web, retornando a quantidade de vulnerabilidades encontradas;

**DNS *Blacklist*** Criar um módulo capaz de pesquisar pelo endereço IP de um determinado servidor de correio eletrónico em listas negras. Caso o endereço IP do servidor responsável pelo envio de correio eletrónico seja encontrado em uma destas listas notificar os administradores.

# Capítulo 2

# Faculdade de Ciências da Universidade de Lisboa

A Faculdade de Ciências da Universidade de Lisboa (Ciências) é uma instituição pública de Ensino Superior, criada por decreto de 19 de Abril de 1911, cuja missão é expandir os limites do conhecimento científico e da tecnologia, transferir esse conhecimento para a sociedade e promover a educação dos seus estudantes através da prática da investigação [34].

No ano letivo 2015/2016 teve um total de 5.159 estudantes inscritos nos diversos ciclos de formação, enquanto que relativamente aos recursos humanos, contou com 619 funcionários no ano 2015, incluindo docentes, investigadores e não docentes [35].

O Campus da Ciências tem cerca de 80.000 m2, com 11 edifícios, 10 departamentos, 20 anfiteatros, 7 bibliotecas, 52 salas de aulas, 6 salas de estudos, 315 laboratórios, 677 gabinetes, 1 centro de dados, 1 refeitório, 8 bares, 1 reprografia e 3 oficinas, tal como ilustra a Figura 2.1.

## 2.1 A Direção de Serviços Informáticos

A Direção de Serviços Informáticos (DSI), na qualidade de unidade de serviço de Ciências, é o órgão responsável pela administração, implementação, suporte e promoção da utilização dos serviços e sistemas informáticos, apoiando o planeamento dessas atividades, apoio à tomada de decisão superior e reporte às entidades competentes, nos termos instituídos. Em outras palavras, cabe à DSI [33]:

- Garantir o atendimento e suporte aos utilizadores e aos laboratórios de informática;
- Administrar os sistemas de informação, dando suporte aplicacional à gestão de Ciências;
- Administrar as infraestruturas tecnológicas incluindo o planeamento e administração da arquitetura de servidores e serviços, instalação e configuração de servidores físicos e tecnologias de virtualização, gestão de sistemas de informação, redes e de serviços de apoio;
- Planear e administrar a infraestrutura de comunicações de Ciências, incluindo a instalação de equipamentos, ativos de rede e sistemas de telefonia IP "VoIP";

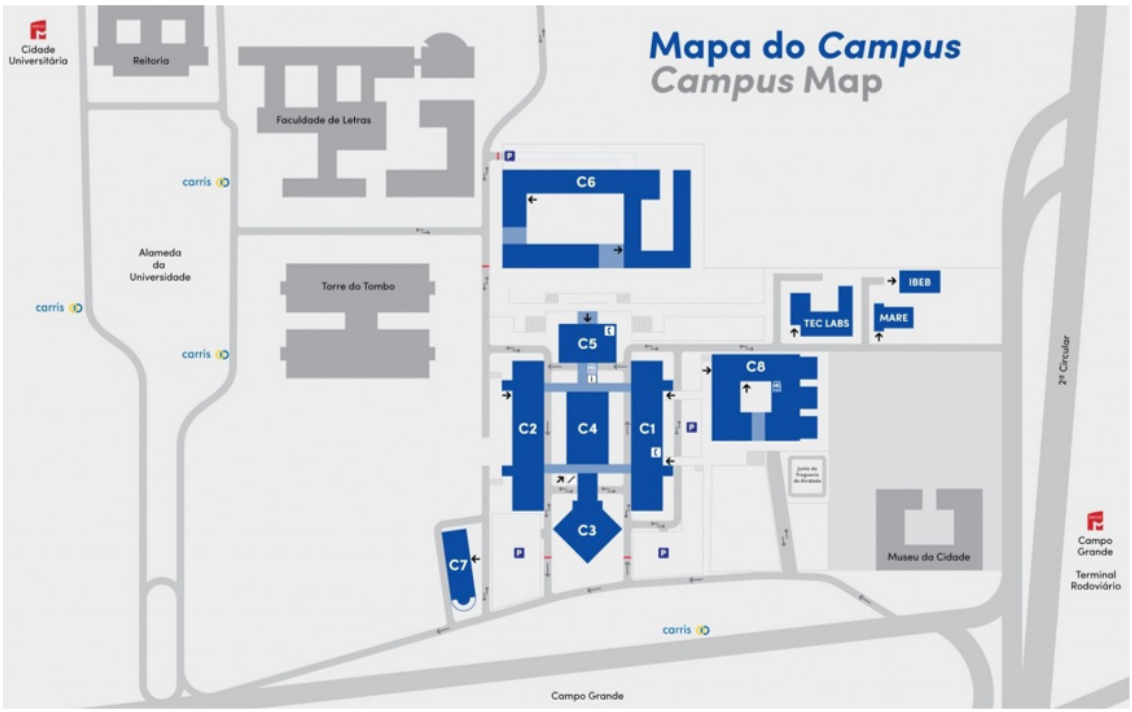


Fig. 2.1: Mapa do Campus da Faculdade de Ciências [33]

- Apoiar as atividades de *e-learning* e multimédia no âmbito de aulas ou eventos.

A DSI é composta pelas Áreas de Serviços e Servidores, Sistemas de Informação e Desenvolvimento, Redes e Comunicações e pelo Gabinete de Suporte ao Utilizador, que juntas garantem o cumprimento das atribuições acima mencionadas.

No âmbito das suas atribuições cabe a DSI:

- Administrar mais de 180 servidores, dos quais 160 são servidores virtuais.
- Dar suporte a mais de 2700 computadores que integram a rede de Ciências.
- Administrar a rede interligada por meio de 117 *switches* e 100 pontos de acesso (APs) à rede sem fios.
- Garantir a disponibilidade e funcionalidade dos 500 domínios alojados nos seus servidores, sendo que a maior parte destes são administrados pelos proprietários, com algumas exceções das quais se destacam o portal de Ciências[1], a plataforma de *e-learning* (moodle[2]) e o sistema de gestão académica (fénix[3]).

## 2.2 Enquadramento do trabalho

Tendo em conta a dimensão do sistema informático administrado pela Direção de Serviços Informáticos da Faculdade de Ciências da Universidade de Lisboa (DSI-Ciências), e

[1] `https://ciencias.ulisboa.pt`
[2] `https://moodle.ciencias.ulisboa.pt/`
[3] `https://fenix.ciencias.ulisboa.pt/`

atendendo ao facto de estar em curso a reformulação do sistema de monitorização, tornou-se oportuno a inclusão de módulos desenvolvidos com a capacidade de produzirem alertas de segurança.

A implementação de uma ferramenta de monitorização de sistemas em Ciências, é um processo de elevada importância, por vários fatores, das quais se destacam:

**Recursos humanos limitados** A DSI-Ciências tem um número reduzido de técnicos, e o uso de uma ferramenta de monitorização vem auxiliar os administradores, que passam a ter uma visão geral do sistema, com alertas via SMS, correio eletrónico ou qualquer outro meio definido [15], o que garante um controlo do sistema mesmo quando se está ausente do local de trabalho;

**Dimensão da rede** A rede de Ciências, tal como descrita anteriormente, é extensa, com um elevado número de equipamentos, o que dificulta a sua monitorização ativa;

**Número de utilizadores** O número de utilizadores dos serviços disponibilizados pela DSI-Ciências é elevadíssimo, o que implica maior e rigoroso controlo, tarefa que a monitorização 24x7 e automatizada simplifica;

**Diversidade de tráfego** Por se tratar de uma instituição de ensino superior, é comum a existência de um elevado tráfego na rede, produzido em ambiente de investigação. Porém, é fundamental que se tenha um controlo deste tráfego, e o uso de uma ferramenta de monitorização incrementa a capacidade dos administradores poderem manter um controlo eficaz destes conteúdos.

# Capítulo 3

# Trabalhos Relacionados

As ferramentas de monitorização simplificam e padronizam a administração de redes e sistemas, uma vez que permitem que se tenha um controlo centralizado, personalizado, remoto e eficaz dos ativos. Ajudam a diminuir o tempo de reação dos administradores, e consequentemente evitam ou minimizam o impacto das falhas nas redes, nos sistemas e seus serviços.

O Nagios (descrito com maior detalhe na Seção 3.8) é a ferramenta de monitorização de redes e sistemas pioneira, estatuto que associado à sua fácil usabilidade e flexibilidade tornou-a numa das mais populares ferramenta deste género distribuída gratuitamente [13][23][15]. No entanto com o passar dos anos foram surgindo várias ferramentas, sendo que maior parte destas baseiam-se no núcleo do Nagios, com diferença em pequenos detalhes como por exemplo na instalação e configuração.

As ferramentas de monitorização vêm sendo estudadas já a algum tempo, e existe por isso uma série de projetos disponíveis, muitos deles publicados e alguns referenciados nesta dissertação.

Hsiu-Lien Yeh [37] apresenta um sistema de monitorização chamado SIAM baseado no Nagios. O SIAM oferece suporte a serviços de monitorização para sistemas baseados em *Storage Resource Broker* (SRB)/*the Integrated Rule-Oriented Data System* (iRODS), incluindo funções de tolerância a falhas e notificações.

Josune Hernantes, Gorka Gallardo e Nicolás Serrano descrevem em [15], como conseguiram melhorar a qualidade do serviço prestado usando ativamente a tecnologia de monitorização de infraestruturas tecnológicas. Neste artigo, os autores apresentam estudos comparativos das mais recentes e populares ferramentas de monitorização.

Mongkolluksamee [24] foca-se no estudo das limitações do Nagios. Duas identificadas são a difícil configuração, assim como a interface do utilizador pouco atraente. No entanto, destacam o facto destas limitações poderem ser melhoradas usando programas externos denominados *addons*. Os *addons* nem sempre são fáceis de configurar, pelo que segundo os autores são necessárias algumas habilidades por parte dos administradores para poderem configurar um ambiente de monitorização funcional e amigável baseado no Nagios.

Matýsek [23] apresenta um trabalho desenvolvido com o objetivo de testar e preparar um sistema de monitorização de redes e serviços de computadores. O Nagios é a ferramenta analisada e descrita ao pormenor, incluindo por exemplo conceitos e parâmetros de configuração. Os autores também abordam a instalação e aplicação das funcionalidades

do Nagios nos sistemas operativos mais populares.

Apesar da variedade de trabalhos publicados sobre as ferramentas de monitorização, não é fácil encontrar trabalhos que visem tirar proveito do potencial destas ferramentas para gerar indicadores de segurança e consequentemente incrementar a segurança dos sistemas.

No entanto, as ferramentas de monitorização de redes e sistemas já são uma realidade no dia-a-dia de muitas organizações, e pode ser encontrada uma grande diversidade no mercado. As ferramentas existentes caraterizam-se pela sua diversidade, mas a implementação numa infraestrutura de tecnologias de informação carece de alguma ponderação. Para escolha da ferramenta adequada é necessário ter em conta vários fatores, essencialmente a avaliação da que melhor atende às necessidades de cada infraestrutura e o fator custo benefício.

Algumas opções conhecidas são IBM Tivoli Monitoring, Monitis, Icinga, Zenoss, Cacti, Ganglia, Zabbix e Nagios, que a seguir descrevemos de forma breve.

## 3.1 IBM Tivoli Monitoring

O IBM Tivoli[1] é uma ferramenta de monitorização proprietária. A sua instalação é considerada simples, porém a configuração, a atualização e a refinação dos recursos analíticos e de resposta exigem conhecimentos avançados e experiência em Tecnologia de Informação (TI).

Esta ferramenta dispõe de uma interface Web intuitiva com espaços de trabalho personalizáveis, inclui um *data warehouse* de fácil utilização e recursos avançados de relatórios. Esta ferramenta fornece uma análise dinâmica de limiares e desempenho para melhorar a deteção de incidentes. Permite também uma monitorização pró-ativa e administração automatizada de falhas.

O IBM Tivoli coleciona informações de monitorização para relatórios, análise de desempenho e previsão de tendências. O licenciamento inclui suporte por telefone, correio eletrónico e acesso à wiki contendo documentação do produto.

## 3.2 Monitis

Monitis[2] é uma ferramenta de monitorização de código aberto e gratuita baseada em computação na nuvem do TeamViewer[3][4]. Fornece recursos para monitorização de serviços e servidores Windows e Linux, e dispensa instalação, configuração ou atualização, por ser hospedada na nuvem.

O seu uso exige uma conexão a Internet, e que se esteja registado no sitio oficial da ferramenta, passando a dispor de um espaço de monitorização 24/7 a partir de qualquer

[1] `https://www.ibm.com/`

[2] `http://www.monitis.com/`

[3] `https://www.teamviewer.com/pt/`

[4] O TeamViewer é um software para acesso remoto, partilha de ambiente de trabalho, conferência online e transferência de ficheiros entre computadores. No entanto, o foco principal da aplicação é o acesso remoto a computadores. Também estão incluídos recursos de colaboração e de apresentação.

lugar. Os alertas desta ferramenta podem ser por correio eletrónico, SMS, Twitter ou chamada telefónica.

## 3.3 Icinga

O Icinga[5] é igualmente uma ferramenta de monitorização de código aberto, resultante de uma ramificação das versões inicias do Nagios [2].

Surgiu de modo a apresentar uma alternativa viável ao popular software de monitorização e a criar uma evolução diferente ao modelo adotado por essa aplicação. Esta ferramenta tem a Interface Gráfica do Utilizador (*Graphical User Interface*, GUI) melhorada e pode ser integrada com várias base de dados. a sua interface da API REST facilita a integração com outras aplicações.

Notifica os administradores sobre a ocorrência de erros e reconfigurações, adicionalmente gera dados de desempenho para relatórios. É dimensionável, extensível e pode monitorizar ambientes grandes e complexos em locais dispersos.

## 3.4 Zenoss Core

O Zenoss Core[6] é um software de código aberto para monitorização de redes, aplicações, servidores físicos e virtuais, com base no servidor de aplicações Zope. Distribuído sob a licença *GNU General Public License* (GPL) versão 2, o Zenoss core fornece uma interface Web que permite que os administradores de sistema monitorizem a disponibilidade, as configurações, o desempenho e os eventos relevantes para o funcionamento do sistema sem recorrer a utilização de agentes [5].

A Zenoss Inc, organização proprietária deste software disponibiliza três soluções, nomeadamente o *Zenoss Core* de código aberto e distribuído gratuitamente, o *Zenoss Service Dynamicsa* distribuído comercialmente, e o *Zenoss as a Service* (ZaaS) serviço de oferta.

## 3.5 Cacti

Cacti[7] é também uma plataforma de código aberto para monitorização de ativos de redes informáticas [37][2]. Esta ferramenta recolhe e exibe quase em tempo real as informações sobre o estado dos equipamentos em uma rede de dados.

O Cacti utiliza o Protocolo Simples de Administração de Redes (*Simple Network Management Protocol*, SNMP) para colecionar informações de monitorização e usa o *Round Robin Database* (RRDtool)[8] para a apresentação gráfica destas informações em uma in-

[5] `https://www.icinga.com/`

[6] `http://www.zenoss.org/`

[7] `http://www.cacti.net/`

[8] RRDTool é um sistema de base de dados *round-robin* criado por Tobias Oetiker sob licença GNU GPL. Foi desenvolvido para armazenar séries de dados numéricos sobre o estado de indicadores como temperatura, uso de CPU, etc. RRD é um modo abreviado de se referir a *Round Robin Database* (base de dados *round-robin*).

A base de dados gerada possui um tamanho máximo o qual uma vez atingido não é ultrapassado. Os dados numéricos armazenados são consolidados conforme a configuração fornecida, de modo que a resolução

terface Web. O armazenamento de dados é realizado numa base de dados MySQL.

## 3.6 Ganglia

Ganglia[9] é um sistema de monitorização de código aberto, desenvolvido pelo projeto UC Berkeley Millennium, sob a licença *Berkeley Software Distribution* (BSD). Foi concebido para sistemas de elevado desempenho, possui uma arquitetura escalável, distribuída e hierárquica das máquinas monitorizadas [37], e se baseia num protocolo de *multicast* para monitorizar os estados dos equipamentos e sistemas.

O Ganglia usa tecnologias como XML para representação de dados, php para desenvolvimento Web e *Round Robin Database* (RRDtool) para armazenamento e visualização de dados.

## 3.7 Zabbix

Zabbix[10] é um software *open source* escrito e distribuído sob a *General Public License* (GPL) versão 2, que oferece grande desempenho para colecionar dados e pode dimensionar-se para monitorização de grandes ambientes. Esta ferramenta permite a monitorização de servidores, dispositivos de rede e aplicações sem recurso a agentes, colecionando estatísticas precisas e dados de desempenho.

A instalação desta ferramenta é simples, mas a configuração pode ser complexa, especialmente para adicionar novas monitorizações personalizadas. O Zabbix tem uma poderosa interface gráfica do utilizador (*Graphical User Interface*, GUI), com geração de relatórios e gráficos extensivos como parte do pacote padrão que combina a monitorização e funcionalidades de avaliação de tendências. As notificações no Zabbix podem ser por correio eletrónico ou SMS [15][2].

## 3.8 Nagios

Este trabalho utiliza a plataforma de monitorização Nagios[11], ferramenta de código aberto, distribuída sob a licença GPL, de fácil utilização [2], e cuja base consta de um núcleo que permite a incorporação de plugins desenvolvidos para executar tarefas específicas, exibindo todos os resultados obtidos numa interface Web.

Em seus primeiros passos como plataforma de monitorização, foi designado de Netsaint, escrito e mantido por Ethan Galstad, que com a ajuda de um leque de programadores, mantém ativamente uma biblioteca de plugins oficiais e não oficiais [13][15].

Esta ferramenta é também disponibilizada comercialmente para organizações que optem pelo suporte técnico na sua implementação e manutenção, versão conhecida como ***Nagios XI***, com planos anuais a partir dos 2.495,00 USD [12][15].

deles seja reduzida de acordo com o tempo que eles estão armazenados. Neste processo, apenas as médias dos valores antigos são armazenados.

[9] `http://ganglia.info/`

[10] `http://www.zabbix.com/`

[11] `https://www.nagios.org/`

O Nagios alerta quando ocorrem problemas e também quando estes são resolvidos. Estas notificações podem ser por correio eletrónico, pagina web, SMS, ou qualquer outro meio definido pelo utilizador [15][23][37]. Foi originalmente escrito para o sistema operativo Linux, no entanto, hoje em dia já existe suporte para outros sistemas operativos [13].

O desenvolvimento de plugins para o Nagios é considerado simples, e por isso, permite que os utilizadores criem seus próprios padrões de monitorização, em função das suas necessidades, usando qualquer linguagem de programação.

### 3.8.1 Plugins

O Nagios não possui nenhum mecanismo interno para realizar operações de diagnóstico. Utiliza por isso os plugins, que são pequenos programas externos concebidos para efetuar essas tarefas [7]. A Figura 3.1 representa a camada de abstração dos plugins do Nagios.

Os Plugins são programas executáveis configurados para verificar o estado de um *host* ou serviço, efetuando testes aos dispositivos e que retornam o estado de acordo com os parâmetros previamente definidos e invocados pelo Nagios [25]. Eles retornam quatro possíveis estados [23]:

**OK** Quando o sistema, um componente do sistema ou um serviço estiver a funcionar dentro dos parâmetros previamente estabelecidos;

**WARNING** Quando o sistema, um componente do sistema ou um serviço estiver a funcionar acima dos limites máximos estabelecidos para o normal funcionamento, no entanto não são ultrapassados os limites mínimos considerados *critical*;

**CRITICAL** Quando o sistema, um componente do sistema ou um serviço estiver a funcionar acima dos limites máximos considerados *warning*, ou quando ocorrer um *timeout*;

**UNKNOWN** Diante de erro do *plugin* ou definição incorreta de parâmetros.

Em uma visita rápida ao repositório oficial de plugins do Nagios, é possível encontrar alguns plugins desenvolvidos com o objetivo de automatizar a segurança dos sistemas, dos quais destacamos o *ckeck_kdc*[12], o *check_sslcertexpiry*[13], o *check_ipsec*[14].

### 3.8.2 Requisitos de instalação e configuração

A instalação e configuração do Nagios requer no mínimo uma máquina física ou virtual com um sistema operativo da família Unix e um compilador C. É também necessário uma configuração adequada do protocolo TCP/IP, tendo em conta que a maior parte das suas ações são feitas pela rede, recorrendo a este protocolo.

Outros requisitos são a instalação e configuração de um servidor web (Apache), que permite o uso da interface web do Nagios. A biblioteca gráfica (Graphics Library, GD) e o compilador GCC (GNU Compiler Collection, GCC) são outros requisitos necessários.

[12] `https://exchange.nagios.org/directory/Plugins/Security/check_kdc/details`

[13] `https://exchange.nagios.org/directory/Plugins/Security/CheckSSLCertExpiry/details`

[14] `https://exchange.nagios.org/directory/Plugins/Security/check_ipsec/details`

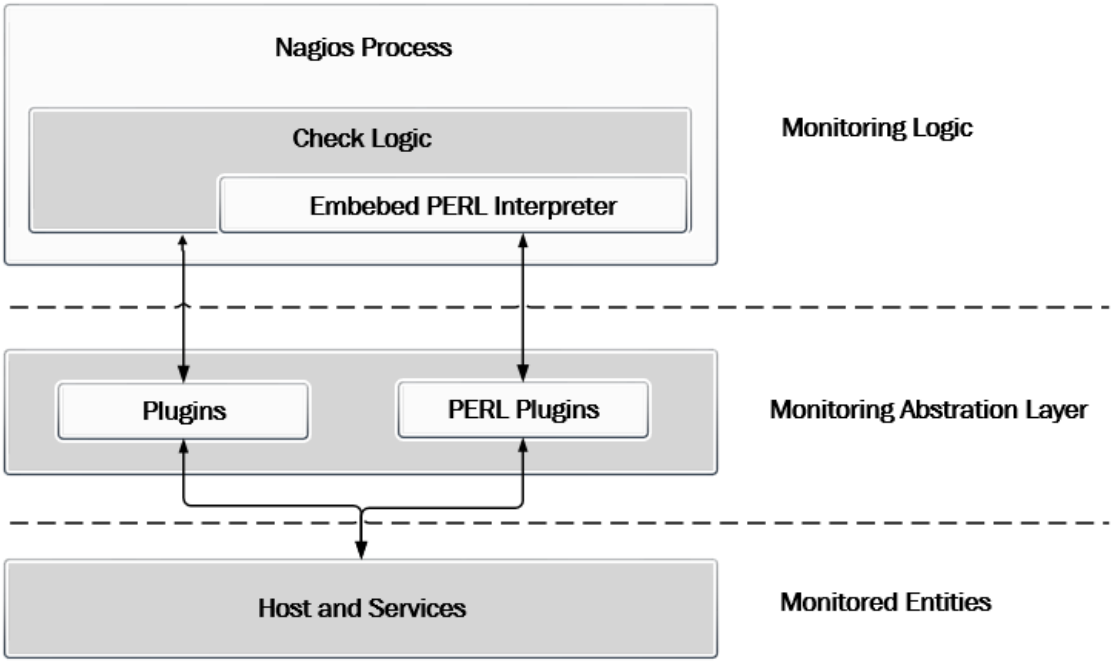


Fig. 3.1: Plugin do Nagios como uma camada de abstração [13]

### 3.8.3 Arquitetura

**Agentes e gestores**

Os softwares de administração de redes que utilizam o modelo cliente-servidor[9] necessitam de dois componentes. O componente que é executado no computador local e atua como um cliente, também chamado de gestor e o componente que é executado num dispositivo remoto na rede, e atua como um servidor, também conhecido por agente. Ambos são ilustrados na Figura 3.2. Para monitorização remota os principais agentes usados pelo Nagios são o *Simple Network Management Protocol* (SNMP) [9][23], o *Nagios Remote Plugin Executor* (NRPE) [7][23][27], o *Nagios Service Check Acceptor* (NSCA) [7][27] e o NSClient++ [7][23]:

**Simple Network Management Protocol, (SNMP)** O Protocolo Simples de Administração de Redes [9] é o protocolo padrão de administração de redes utilizado na Internet [23]. Ele define o formato e o significado das mensagens trocadas entre gestor e agente. Em vez de definir muitas operações, o SNMP usa o paradigma *fetch-store*, em que um gestor envia operações de *fetch* para buscar valores de variáveis no dispositivo ou operações de *store* para armazenar valores em variáveis do dispositivo. A Figura 3.3 ilustra o modelo cliente-servidor, representado por gestor e agente.

**Nagios Remote Plugin Executor (NRPE)** O Executor Remoto de Plugins Nagios é um complemento para o Nagios em Linux. Foi criado para estender a monitorização de máquinas remotas, executando os plugins através da *shell* [7][23][27].

Para concretizar a monitorização remota é necessário que estejam instalados os pacotes *check_nrpe* e *nagios-nrpe-server* no servidor Nagios e na máquina monitorizada

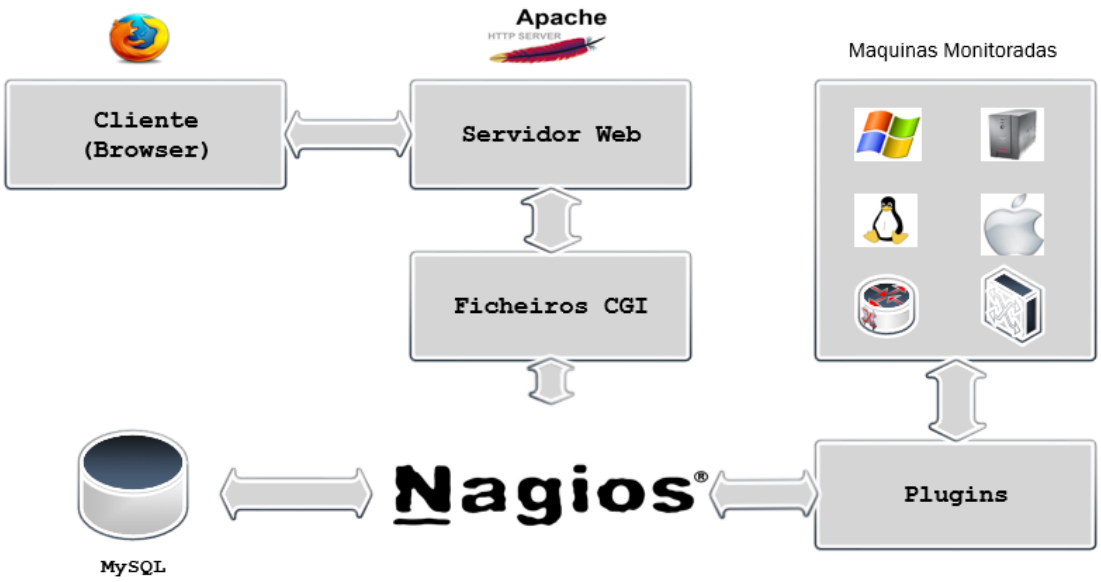


Fig. 3.2: Arquitetura do Nagios

remotamente, como ilustra a Figura 3.4.

Para efetuar uma consulta de monitorização, o Nagios executa o plugin *check_nrpe*, que envia uma *string* contendo o nome do comando que pretende executar na máquina remota (por exemplo: *check_http*). O NRPE ao receber a *string*, faz uma pesquisa no seu ficheiro de configuração em busca do comando correspondente, que ao ser encontrado é executado e retorna o resultado pela mesma via.

O NRPE pode efetuar observação de forma direta ou indireta. Na primeira, o agente tem como função monitorizar serviços e recursos locais como a carga da Unidade Central de Processamento (*Central Processing Unit*, CPU), a memória que está a ser utilizada, o número de utilizadores, etc. No segundo caso, o agente tem a capacidade de consultar e aplicar um comando a um agente remoto.

Este tipo de monitorização é caracterizado por apresentar uma série de vantagens, nomeadamente ao nível dos mecanismos de segurança e encriptação que consegue oferecer. Além disso, a sobrecarga na comunicação entre servidor e agente é inferior quando comparado com outros tipo de comandos, o que significa que o servidor Nagios e a máquina remota utilizam menos recursos da CPU para desempenhar as suas tarefas, situação bastante importante quando estão a ser monitorizados um número elevado de máquinas.

**Nagios Service Check Acceptor (NSCA)** O NSCA é um programa que permite que um computador envie mensagens de verificação ao servidor Nagios por sua própria iniciativa, ou seja, permite que o servidor de monitorização receba informações sobre a disponibilidade e funcionamento de serviços ou equipamentos de forma passiva [7][27]. O cliente NSCA pode executar como uma aplicação independente do servidor Nagios [30]. A Figura 3.5 representa o modo de funcionamento do NSCA.

**NSClient++** O NSClient++ é uma ferramenta análoga ao NRPE, utilizada para monitorizar máquinas com o sistema operativo Windows. Funciona como um serviço e vem


Fig. 3.3: Agente e gestor Nagios

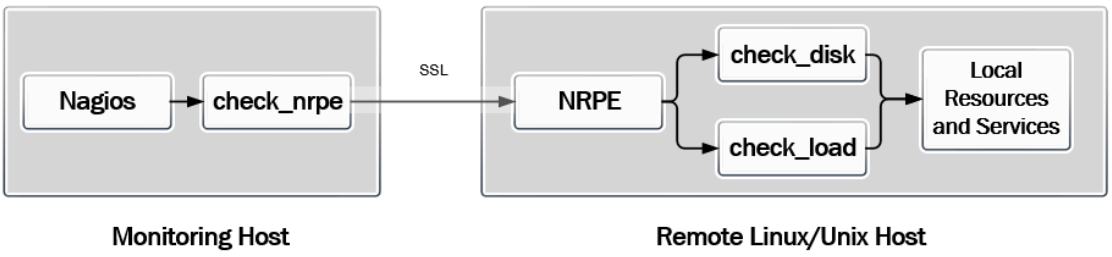


Fig. 3.4: Arquitetura do NRPE [13]

constituído com uma série de módulos e funções que permitem a monitorização de recursos [7][23].

Esta ferramenta pode operar em dois modos distintos: o primeiro é idêntico ao programa original NSClient e utiliza o comando *check_nt* para monitorizar a máquina remota, como ilustra a Figura 3.6; o segundo modo é uma implementação do NRPE, onde o gestor se conecta ao agente através do plugin *check_nrpe* [30].

### 3.8.4 Categorização dos equipamentos

Na arquitetura do Nagios, os alvos monitorizados são distinguidos por serviços, servidores e equipamentos de rede, que representam máquinas físicas e/ou virtuais (servidores, encaminhadores, comutadores, estações de trabalho, impressoras, etc.) e aquilo que o Nagios classifica como serviços disponibilizados por estes equipamentos, como por exemplo: verificar o estado da Unidade Central de Processamento (*Central Processing Unit*, CPU), da memória, do disco de armazenamento, ou do estado dos protocolos de rede (HTTP, ICMP, SMTP, POP3, FTP, SNMP, ...) [13][29].

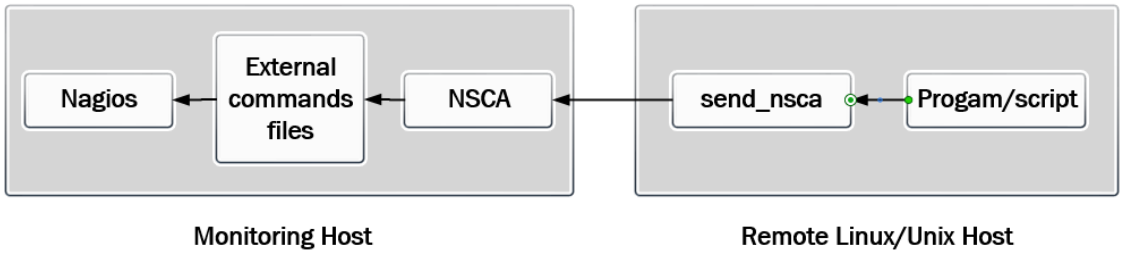


Fig. 3.5: Submissão dos resultados de verificação passiva de hosts pelo NSCA [13]

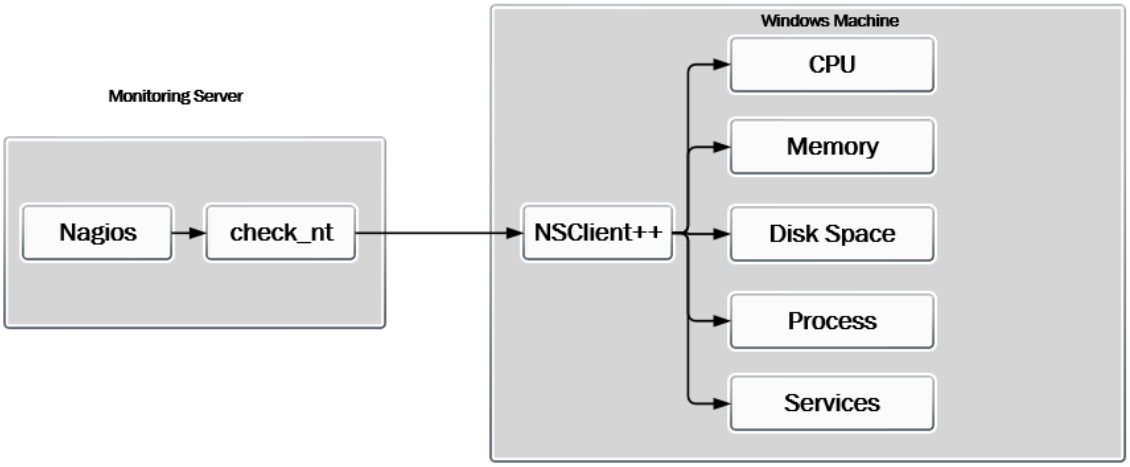


Fig. 3.6: Arquitetura do NSCLIENT++ [13]

### 3.8.5 Funcionalidades

O uso do Nagios como plataforma de monitorização oferece às organizações vários benefícios, dos quais se destacam [8]:

**Acompanhamento exaustivo** Permite a monitorização de todos os componentes da infraestrutura de Tecnologia de Informação (TI), incluindo aplicações, serviços, sistemas operativos, protocolos de rede, métricas de sistemas e infraestrutura de rede.

**Centralização** Oferece aos administradores de sistemas uma visão central de toda a sua infraestrutura, operações e processos de negócio.

**Notificações** São enviados alertas para os administradores de sistemas, utilizando qualquer mecanismo que venha a ser configurado, principalmente usando o correio eletrónico e mensagens de texto por SMS.

**Correção de problemas** Manipuladores de eventos permitem que as aplicações, serviços, servidores e dispositivos de rede sejam reiniciados automaticamente quando são detetados problemas.

**Tendências e capacidade de planear** Dá aos administradores a capacidade de planear melhorias na infraestrutura, evitando que sejam surpreendidos com sistemas e serviços desatualizados.

**Relatórios** Verifica se os acordos de nível de serviços (*Services Level Agreement*, SLA) são atendidos, proporcionando registos, históricos de quedas, notificações e resposta de alertas para análise posterior.

**Arquitetura extensível** Oferece fácil integração com *software in-house* e aplicações de terceiros.

### 3.8.6 Funcionamento

Como descrito na Subseção 3.8.1, a monitorização baseada no Nagios depende de um conjunto de plugins responsáveis por proceder às verificações necessárias e analisar os resultados recebidos, o que o torna numa ferramenta bastante poderosa, dando aos utilizadores a possibilidade de desenvolver os seus próprios plugins, e assim conseguirem realizar as mais variadas verificações, utilizando qualquer linguagem de programação.

O Nagios possibilita que a monitorização seja distribuída, existindo duas ou mais estações descentralizadas, enviando os seus resultados para uma máquina central, responsável por apresentar o panorama geral da rede. Há também a possibilidade de ser configurado para efetuar monitorizações redundantes, configurando duas estações para controlar os mesmos serviços. Uma delas envia as notificações e a outra assume essa tarefa em caso de falha da primeira.

Outra grande vantagem no funcionamento do Nagios, e ainda no que diz respeito a monitorização, é a sua capacidade de escalonamento. Com esta capacidade, caso haja um número muito elevado de itens a monitorizar o Nagios garante que não há qualquer risco de alguns não serem observados por falta de tempo, pois de forma automatizada escalona a sua atividade, garantindo que todas as tarefas são executadas, exercendo a mínima sobrecarga possível.

Além disso, é importante realçar que o Nagios também tem a capacidade de representar a rede hierarquicamente (*hosts* pais e *hosts* filhos), distinguindo equipamentos indisponíveis de equipamentos inalcançáveis [27]. Isso significa que a partir do servidor de monitorização é possível construir uma árvore hierárquica, onde o servidor Nagios fica no topo e os *hosts* seguintes serão posicionados ao longo das ramificações, conforme topologia da rede.

### 3.8.7 Configuração

A configuração do Nagios é uma tarefa complexa, entretanto simplificada porque a maior parte dos parâmetros necessários para se ter um sistema funcional já estão por omissão definidos para um uso básico [7].

A configuração central é definida no ficheiro nagios.cfg, geralmente armazenado na diretoria /etc/nagios, onde é feita a ligação para outras definições, armazenadas em diferentes arquivos, com exceção das definições de CGI [28]:

**nagios.cfg** Por omissão este é o ficheiro de configuração principal do Nagios, onde são definidos um conjunto de variáveis e caminhos para os restantes ficheiros de configuração. Um caminho muito importante aqui definido pode ser encontrado na variável log_file, para onde o Nagios escreve todos os erros que encontra. Por omissão o ficheiro é */usr/local/nagios/var/nagios.log* [25];

**cgi.cfg** Contem as permissões de acesso dos utilizadores da interface do Nagios e configurações de mapa;

**resource.cfg** Define as diretorias onde são armazenados os plugins;

**hosts.cfg** Este é o ficheiro onde são configurados os *hosts*, ou seja, é neste ficheiro onde por omissão são especificadas as máquinas a serem monitorizadas, onde entre outros atributos, encontramos o nome e endereço do *host*;

**commands.cfg** São aqui definidos os comandos que invocam os plugins utilizados pelo Nagios;

**hostgroups.cfg** Neste ficheiro são definidos os grupos de máquinas, o que é bastante útil, pois permite associar máquinas com caraterísticas semelhantes, permitindo a administração das máquinas por grupo, com grande benefícios na configuração, assim como na visualização dos mapas;

**contacts.cfg** É neste ficheiro onde são armazenadas as configurações dos contactos aos quais deverão ser enviados os alertas, na definição de contactos são considerados de entre outros atributos o nome do contacto, o período de notificação, o comando de notificação e o correio eletrónico;

**contactgroups.cfg** Neste ficheiro são armazenadas as configurações dos grupos de contactos, o que faz com que o Nagios perceba os grupos aos quais devem ser enviadas notificações;

**services.cfg** Neste ficheiro são definidos os serviços a serem verificados pelo Nagios;

A Figura 3.7 ilustra a interdependência dos vários ficheiros de configuração que suportam o Nagios.

### 3.8.8 Vantagens e Limitações

**Algumas vantagens**

- Existem muitos profissionais com elevada experiência em administração do Nagios;
- Um bom conhecimento do Nagios faz com que a configuração manual o possa transformar em uma ferramenta muito poderosa para monitorizar eventos isolados ou particulares;
- Tem uma vasta oferta de plugins para o adaptar às necessidades finais do utilizador;
- Configuração básica bastante facilitada.

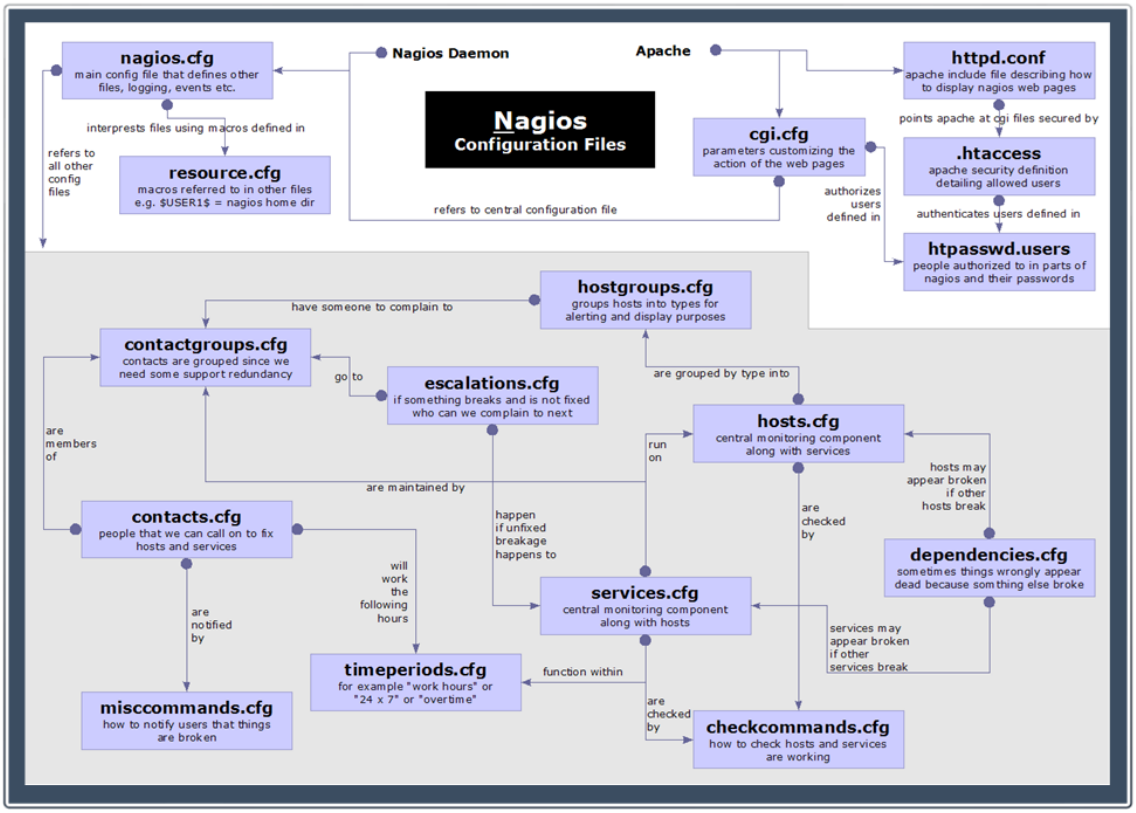


Fig. 3.7: Ficheiros de configuração do Nagios

**Algumas limitações**

- Os processos de configuração avançada são dificultados pela necessidade de proceder a modificações manuais para uma correta configuração da ferramenta;
- A Interface Gráfica do Utilizador (*Graphical User Interface*, GUI) não possui facilidade de utilização;
- Curva de aprendizado íngreme e dispendiosa;
- Relatórios muitos simples.

### 3.8.9 Implementação do Nagios em Ciências

Tal como mencionado na Seção 1.1, este trabalho foi impulsionado pelo esforço empreendido pela Direção de Serviços Informáticos (DSI), na implementação de um novo sistema de monitorização integrado utilizando o Nagios.

Como fruto deste trabalho, no momento da escrita desta dissertação, a plataforma Nagios da DSI monitorizava cerca de 182 *hosts* e 806 serviços. Porém, esta monitorização tinha características típicas de administração de sistemas, não incluindo indicadores voltados para segurança informática.

Alguns dos plugins instalados nesta plataforma monitorizam:

- A utilização da Unidade Central de Processamento (*Central Processing Unit*, CPU);

- A utilização de memória RAM;
- Espaço de armazenamento em disco;
- A existência de atualizações para algumas aplicações em uso;
- O funcionamento dos protocolos HTTP, DNS, DHCP e LDAP;
- As configurações e funcionamento da base de dados MYSQL;
- A temperatura dos equipamentos.

O processo de configuração e implementação do Nagios em Ciências foi uma óptima oportunidade para aprofundar o nosso estudo, desenvolvendo e implementando módulos específicos, apresentados no Capítulo 4.

# Capítulo 4

# Plugins de Monitorização

Este capítulo apresenta os doze módulos desenvolvidos no âmbito da elaboração desta dissertação. Cada um dos módulos tem como objetivo contribuir na deteção precoce de uma vulnerabilidade de segurança utilizando a plataforma de monitorização Nagios. A estrutura de apresentação de cada um dos módulos é a seguinte:

## 4.1 Monitorização do ficheiro de log do Apache

**Introdução** O Servidor HTTP Apache[1] é um servidor Web bastante popular, usado principalmente em sistemas operativos Linux. Ambos são bastante usados em servidores de páginas desde a popularização da Internet em 1995, superando por exemplo o servidor web nginx e o sistema operativo Windows respetivamente [17]. Em Maio de 2017, a quota de utilização do Apache em Portugal foi de 57,70% [31].

O Servidor Apache é compatível com praticamente todas as distribuições Linux atuais. Uma vez instalado, basta ativar o serviço "httpd"através do ntsysv, linuxconf, mcc, ou outra ferramenta disponível na distribuição em uso.

As respostas do servidor Apache às requisições, são associadas a um código de resposta, que por si só identifica o estado da requisição ou tratamento dado à requisição, e estes agrupam-se em classes, conhecidas como 1xx, 2xx, 3xx, 4xx e 5xx [17]:

**Classe 1xx (informativa/resposta provisória)** Indica a receção da informação e que o seu tratamento está em curso.

**Classe 2xx (sucesso/bem sucedido)** Os códigos desta classe são associados a conclusão com sucesso da operação solicitada.

**Classe 3xx (redirecionamento)** Os códigos desta classe são responsáveis por sinalizar o redirecionamento do pedido, neste caso a informação adicional inclui o novo URL onde o recurso pode ser encontrado. Nesta classe o código mais conhecido é o 301, que indica que a pagina foi movida permanentemente.

**Classe 4xx (erro do cliente/requisição)** Esta classe representa a ocorrência de erros no pedido, frequentemente associados a solicitação de conteúdos não existentes. O estado 401 pode sinalizar a tentativa de acesso a um determinado conteúdo por

[1] `http://www.apache.org/`

parte de um cliente com endereço IP interdito. Enquanto que o estado 403 é exibido quando o cliente não dispõe de permissão para aceder ao conteúdo requisitado. Já o estado 404, o mais conhecido desta classe, sinaliza a tentativa de acesso a conteúdo inexistente.

**Classe 5xx (erro do servidor)** Os códigos associados a esta classe revelam a ocorrência de um erro a nível do servidor, que o impede de atender o pedido. O estado 500 é o mais conhecido desta classe e sinaliza a impossibilidade de processar ou aceitar a solicitação devido a um problema interno do servidor.

**Problemas de segurança** Os três estados da classe 4xx descritos acima são exibidos em casos em que o cliente pode estar a executar ações maliciosas, por tentar aceder a conteúdo não autorizado, conteúdo para o qual não dispõe da credencias necessárias ou conteúdo inexistente. Estes eventos podem evidenciar uma tentativa de descoberta de conteúdos, ou tentativa de acesso por força bruta (*brute force attack*), em que são usadas ferramentas automatizadas, capazes de fazerem centenas de tentativas de acesso com combinações predefinidas, requerendo imediata intervenção dos administradores.

**Solução** O módulo Nagios ***check_apache_status*** audita de forma automatizada o ficheiro de log do servidor Apache onde são registadas as respostas aos pedidos recebidos. Na pesquisa que o módulo efetua ao ficheiro de log, são procurados os estados de respostas acima mencionados (401, 403 e 404), gerando alertas para a sua existência com o estado *warning* ou *critical*, desde que a quantidade encontrada esteja dentro dos limites mínimos e máximos passados por meio dos argumentos *-w* e *-c*.

Adicionalmente, o módulo agrupa a quantidade de códigos por endereço IP, fazendo com que o administrador seja também alertado quando uma determinada quantidade de códigos da classe 4xx tenham sido gerados pelo mesmo endereço. Após deteção, o módulo copia todas as linhas com os referidos códigos e as armazena em um ficheiro de texto que deverá ser manualmente inspecionado para melhor perceção da ocorrência.

### O módulo

**Nome** check_apache_status.py

**Linhas de código** 134 linhas (122 sloc)

**Linguagem de programação** Python 2

**Bibliotecas python** os, sys, optparse, strftime fron time, time e datetime.

**Argumentos obrigatórios** Os argumentos seguintes devem ser especificados sempre que o módulo é executado:

**-p ou –path** Usado para especificar o caminho do ficheiro de log do servidor Apache.

**-c ou –critical** Usado para especificar os valores máximos aceitáveis. Este argumento recebe dois valores, um para o número total de códigos encontrados, e o outro para o número de códigos por endereço IP.

**-w ou –warning** Usado para especificar os valores a considerar como warning, à semelhança do *-c* este argumento também é duplo.

**Argumentos opcionais** Os argumentos seguintes são invocados opcionalmente, conforme necessidade do utilizador:

**-n ou –number** Usado para especificar o número de linhas do ficheiro de log a ser lido.

**-V ou –version** Usado para consultar a versão do módulo.

**-A ou –author** Usado para consultar os dados do autor.

**Exemplo de execução em linha de comando**

```
./check_apache_status.py -p /var/log/apache2/access.log -c
300,200 -w 200,150
```

**Status** Em resposta à execução do módulo, o Nagios apresentará um dos seguintes estados:

**OK** Exibido quando não forem encontrados códigos de resposta 401, 403 e 404, ou se o número de erros encontrados for menor que os valores passados como argumento através da opção *-w*.

**WARNING** Exibido quando a quantidade de códigos de resposta encontrados for igual ou superior aos valores passados como argumento por meio da opção *-w* e menor que os valores passados pela opção *-c*.

**CRITICAL** Quando a quantidade de códigos de resposta encontrados for maior ou igual aos valores passados como argumento através da opção *-c*.

**Dependências** Bash (cat, tail e grep)

**Sistema Operativo** Linux

**Testes** Este módulo foi testado nas seguintes distribuições Linux:

Xubuntu-16-04.01

Lubuntu 16.10

Gentoo 2.2

CentOS 7

Debian 8

**Código** Anexo A.1

## 4.2 Monitorização dos programas instalados

**Introdução** Nas plataformas Linux é vulgar existirem centenas de programas, cada um realizando uma tarefa específica. A lista completa de programas pode variar a cada nova atualização ou instalação de pacotes de softwares ou do próprio sistema operativo. Estes programas encontram-se distribuídos por várias diretorias, tornando o seu controlo complexo.

Tendo estes ficheiros a capacidade de execução de qualquer tarefa, é fundamental supervisioná-los para garantir que o computador funcione dentro dos padrões esperados.

**Problemas de segurança** Na sequência de um acesso legítimo ou não (por exemplo usando *Rootkit*[2]), um utilizador malicioso pode instalar outros programas, um exemplo é a instalação de uma *backdoor*, que posteriormente possibilitaria um acesso sem restrições à máquina alvo.

Pode ainda um utilizador com privilégios mínimos conseguir escalar privilégios, obtendo acesso *root* durante uma ação maliciosa, e no decorrer de suas atividades ilícitas, instalar ou atualizar uma ferramenta.

**Solução** Para facilitar a deteção destes cenários foi desenvolvido o módulo ***check_app***. O módulo auxilia no controlo dos softwares instalados, monitorizando por omissão as principais diretorias onde são armazenados estes ficheiros. Opcionalmente podem ser definidas outras diretorias que se deseja monitorizar, de acordo com os interesses do utilizador e com base na configuração do sistema. O módulo utiliza o estado *CRITICAL* para sinalizar a instalação de uma nova aplicação.

As diretorias monitorizadas por omissão são /usr/bin/, /sbin/, /usr/sbin/, /bin/, /usr/-local/bin/. O *script* na sua primeira execução cria um ficheiro texto contendo a listagem de todos programas encontrados nas diretorias monitorizadas. Nas execuções seguintes este compara o conteúdo do ficheiro com o resultado de nova consulta, armazenando em um segundo ficheiro de texto as diferenças encontradas. Depois de validar a nova lista, o administrador terá simplesmente de apagar os ficheiros temporários, que devem ser armazenados numa diretoria de acesso restrito.

### O módulo

**Nome** check_app.py

**Linhas de código** 89 linhas (80 sloc)

**Linguagem de programação** Python 2

**Bibliotecas python** os, sys, optparse e difflib.

**Argumentos obrigatórios** Para executar este módulo não é obrigatório a especificação de qualquer argumento.

[2]Rootkit é um conjunto de ferramentas geralmente usadas para esconder processos, ficheiros ou dados do sistema operativo, atacar e infetar outros computadores, roubar informações e interagir em tempo real com o sistema comprometido

**Argumentos opcionais** Os argumentos seguintes são invocados opcionalmente, conforme necessidade do utilizador:

**-p ou –path** Usado para especificar a diretoria ou caminho completo da diretoria a ser monitorizada.

**-V ou –version** Usado para consultar a versão do módulo.

**-A ou –author** Usado para consultar os dados do autor.

**Exemplo de execução em linha de comando** `./check_app.py`

**Status** O módulo apresenta um dos seguintes estados:

**OK** Exibido quando não forem encontradas novas aplicações ou ficheiros nas diretorias monitorizadas.

**CRITICAL** Quando uma ou mais aplicações novas forem encontradas.

**Dependências** Bash (find)

**Sistema Operativo** Linux

**Testes** Este módulo foi testado nas seguintes distribuições Linux:

Xubuntu-16-04.01

Lubuntu 16.10

Gentoo 2.2

CentOS 7

Debian 8

**Código** Anexo A.2

## 4.3 Monitorização de ataques SYN Flood

**Introdução** Controlar o numero e estados das ligações de rede ativas é fundamental para se ter noção do que acontece na nossa rede, e pode ser um passo essencial para antecipar respostas a ações maliciosas. O comando netstat é um utilitário de rede de elevada importância que pode ser usado para verificar essas ligações. Este utilitário lista para cada ligação o protocolo em uso, o endereço local, o endereço externo, o número do porto de comunicação e o estado da ligação.

**Problemas de segurança** Ligações inexplicáveis podem sinalizar uma ameaça de segurança pois podem indicar que alguém não autorizado está a aceder ao ativo de rede. Além de serem um indício de uma tentativa de acesso, estas ligações impõem um consumo desnecessário de recursos com implicações no desempenho dos serviços.

Os estados SYN_RECV de ligações utilizando o protocolo TCP/UDP são há muito explorados por utilizadores maliciosos, valendo-se da sua característica de estado de ligação persistente. Este estado é caracterizado pelo facto da ligação TCP/UDP permanecer aberta após o envio da resposta, isto é, a ligação mantém-se aberta enquanto os objetos são recebidos, e a resposta utiliza a mesma ligação, que só será terminada após algum tempo sem ser usada.

O ataque que explora estes estados de ligações é conhecido como ***SYN Flood*** (*SYN Flood Attack*). É um ataque de negação de serviço (*Denial of Service* - DoS) ou de negação de serviço distribuído (*Distributed Denial of Service* - DDoS), que consiste na sobrecarga direta na camada de transporte e indireta na camada de aplicação do modelo OSI do sistema-alvo com uma sequência de requisições SYN [11][18]. A Figura 4.1 ilustra este ataque.

Apesar dos ataques DoS ou DDoS não comprometerem os sistemas com invasões ou infeções com vírus, eles tornam os serviços e/ou sistemas indisponíveis, gerando prejuízos financeiros e/ou reputacionais.

De forma resumida um ataque de negação de serviço é uma tentativa maliciosa de interromper um serviço, tornando inativas redes, aplicações ou serviços. Esta interrupção é provocada geralmente pela injeção de um volume excessivo de dados, esgotando as capacidades dos recursos. Estes ataques são capazes de [11]:

- Desativar um computador, serviço específico ou uma rede inteira;
- Ter como alvo alarmes, impressoras, telefones, computadores e demais ativos de rede;
- Atingir recursos do sistema como largura de banda, espaço em disco, velocidade do processador ou informações de encaminhamento;
- Explorar as vulnerabilidades do sistema operativo, de modo a extrair recursos do sistema;
- Bloquear o sistema operativo.

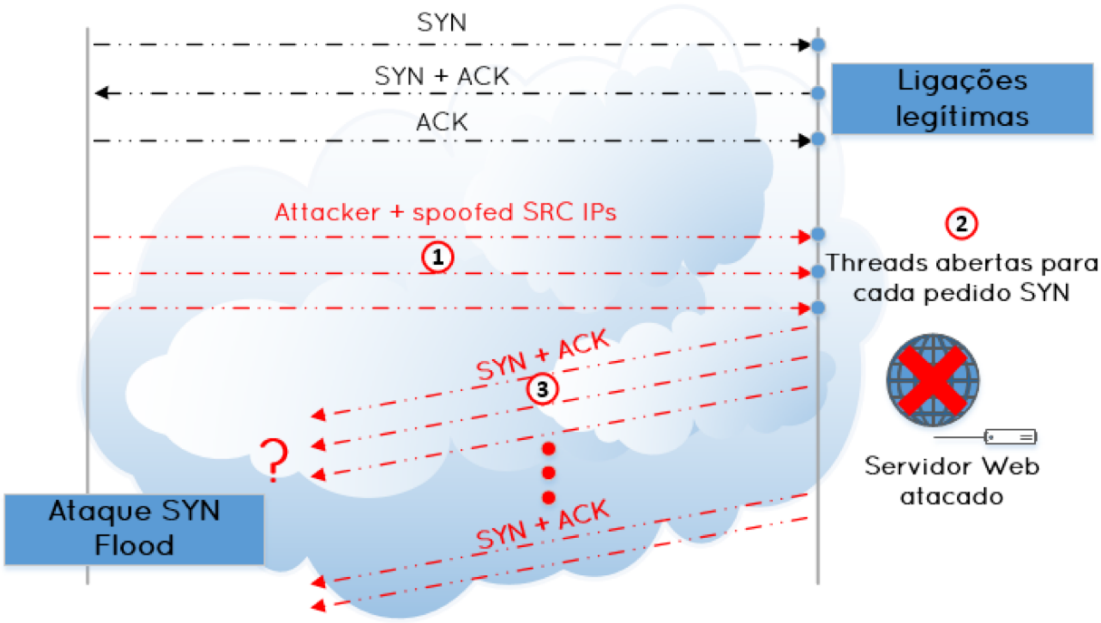


Fig. 4.1: TCP SYN Flood Attack [18]

**Solução** Para detetar e alertar sobre a ocorrência de eventos deste género foi desenvolvido o módulo ***check_synflood***, que monitoriza as ligações de rede do sistema, alertando sempre que se verifique um elevado número de estados SYN_RECV. As quantidades de ligações a serem consideradas excessivas são passadas como argumentos, e quando atingido ou excedido os limites o Nagios alerta com os estados *WARNING* ou *CRITICAL*.

**O módulo**

**Nome** check_synflood.py

**Linhas de código** 85 linhas (74 sloc)

**Linguagem de programação** Python 2

**Bibliotecas python** os, sys e optparse.

**Argumentos obrigatórios** Os argumentos seguintes devem ser especificados quando o módulo for executado:

**-w ou warning** Usado para especificar o número de ligações a partir do qual deve-se considerar como *warning*.

**-c ou critical** Usado para especificar o número de ligações a partir do qual deve o resultado ser considerado como *critical*.

**Argumentos opcionais** Os argumentos seguintes são invocados opcionalmente, conforme necessidade do utilizador:

**-V ou –version** Usado para consultar a versão do módulo.

**-A ou –author** Usado para consultar os dados do autor.

**Exemplo de execução em linha de comando** `./check_synflood.py -c 300 -w 200`

**Status** Em resposta a execução do módulo, o Nagios exibe um dos seguintes estados:

**OK** Exibido quando o número de ligações com o estado SYN_RECV for inferior ao valor passado como argumento através da opção *-w*.

**WARNING** Quando o número de ligações encontrados for menor que o valor passado como argumento através da opção *-c* e maior ou igual que o valor passado como argumento por meio da opção *-w*.

**CRITICAL** Quando o número de ligações for igual ou superior ao valor passado como argumento através da opção *-c*.

**Dependências** Bash (netstat)

**Sistema Operativo** Linux

**Testes** Este módulo foi testado nas seguintes distribuições Linux:

Xubuntu-16-04.01

Lubuntu 16.10

Gentoo 2.2

CentOS 7

Debian 8

**Código** Anexo A.3

## 4.4 Monitorização de paginas web

**Introdução** Uma página web é qualquer documento que faz parte de um sítio na Internet, geralmente em formato *HTML* e com ligações de hipertexto que permitem a navegação pelo restante conteúdo do sítio. Atualmente, uma das maiores preocupações da segurança na Internet é garantir que o conteúdo armazenado em um servidor, seja disponibilizado com as propriedades de integridade, autenticidade e confidencialidade.

**Problemas de segurança** A alteração maliciosa e propositada do conteúdo de paginas web e consequente desfiguração de sítios é um ataque bastante usado por *hackers* ou *defacers*, conhecido como *defacement* [16]. O *defacement* é uma flagrante subversão de duas das principais propriedades de segurança da informação, nomeadamente a integridade e autenticidade, cuja motivação pode diferir de um ataque para outro. As formas mais utilizadas para realizar um *defacement* são:

- Explorar vulnerabilidades do servidor Web onde o sítio está hospedado;
- Explorar vulnerabilidades dos pacotes ou da linguagem de programação utilizada para desenvolver a aplicação Web;
- Explorar erros da aplicação Web;
- Invadir o servidor onde o sítio está hospedado e realizar alteração no conteúdo e na estética dos elementos que compõem o sítio;
- Roubar senhas de acesso à interface Web.

**Solução** O módulo ***check_defacement*** foi desenvolvido com o intuito de alertar os administradores para a presença de conteúdo suspeito em paginas web, auditando a pagina web passada por url como argumento. O módulo procura palavras potencialmente perigosas, retornando o estado *CRITICAL* caso uma ou mais sejam encontradas. Por omissão está definido um conjunto de palavras, que podem ser ignoradas ou a elas concatenadas outras em função das necessidades do utilizador.

**O módulo**

**Nome** check_defacement.py

**Linhas de código** 148 linhas (133 sloc)

**Linguagem de programação** Python 3

**Bibliotecas python** os, sys, optparse, urllib.request, re e urllib.

**Argumentos obrigatórios** O argumento seguinte deve ser especificado quando o módulo for executado:

**-U ou –url** Usado para especificar o url da pagina que será monitorizado.

**Argumentos opcionais** Os argumentos seguintes são invocados opcionalmente, conforme necessidade do utilizador:

**-V ou –version** Usado para consultar a versão do módulo.

**-A ou –author** Usado para consultar os dados do autor.

**-i ou –ignore** Usado para ignorar a lista de palavras perigosas pré-instaladas.

**-I ou –Ignore** Usado para especificar uma ou mais palavras existentes na lista pré-instaladas a serem ignoradas.

**-K ou –keyword** Usado para adicionar à lista predefinida palavras a serem procuradas no conteúdo web.

**Exemplo de execução em linha de comando**

```
./check_defacement.py -U https://ciencias.ulisboa.pt
```

**Status** Em resposta a execução do módulo, o Nagios exibe um dos seguintes estados:

**OK** Exibido quando não forem encontradas palavras consideradas perigosas.

**CRITICAL** Quando uma ou mais palavras forem encontradas.

**Dependências** Bash (wget, grep)

**Sistema Operativo** Linux

**Testes** Este módulo foi testado nas seguintes distribuições Linux:

Xubuntu-16-04.01

Lubuntu 16.10

Gentoo 2.2

CentOS 7

Debian 8

**Código** Anexo A.4

## 4.5 Monitorização da correspondência entre nomes de domínio e endereço IP associado

**Introdução** Um nome de domínio é uma anotação de fácil memorização associada a um endereço IP e que serve para identificar um computador na Internet. As informações de correspondência entre nomes de domínios e endereços IP são armazenadas em bases de dados de computadores chamados Sistemas de Nomes de Domínio (*Domain Name System, DNS*), que asseguram a indicação do endereço certo para a entrega dos pedidos. Essa tarefa é operada através da conversão do nome de domínio num endereço IP que identifica a localização dos computadores na Internet.

Para atingir alta escalabilidade e resiliência a falhas esta base de dados usa replicação intensiva e *cache*.

**Problemas de segurança** Infelizmente o protocolo DNS não foi projetado de forma a ser imune a ataques maliciosos como envenenamento de cache (***DNS cache poisoning***). Estes ataques comprometem a integridade do servidor de DNS. O *DNS Cache Poisoning* caracteriza-se pela injeção de informações falsas na memória cache do servidor alvo, afetando a precisão das pesquisas DNS, levando a que o tráfego destinado a um endereço IP seja encaminhado para outro [38].

Ao receber dados não autenticados e armazena-los em sua *cache*, um servidor de DNS corre o risco de ter os dados forjados. Nesse caso ocorre um ataque de envenenamento da *cache* do servidor que irá fornecer esses dados aos seus clientes.

**Solução** Foi desenvolvido o módulo ***check_dns*** para auxiliar a deteção destes eventos. O módulo monitoriza nomes de domínio e correspondente endereço IP, ambos passados como argumentos, alertando com estado *CRITICAL* caso sejam verificadas incorreções na correspondência entre ambos. Por omissão as consultas DNS são feitas no servidor de DNS da google *(8.8.8.8)*, podendo o utilizador definir outro.

**O módulo**

**Nome** check_dns.py

**Linhas de código** 123 linhas (110 sloc)

**Linguagem de programação** Python 3

**Bibliotecas python** os, sys, OptionParse fron optparse, urllib.request, ipaddress e socket.

**Argumentos obrigatórios** Os argumentos seguintes devem ser especificados quando o módulo for executado:

- **-H ou –host** Usado para especificar o nome de domínio.
- **-I ou –hostaddress** Usado para especificar o endereço IP associado, ou que se espera estar associado ao nome passado como argumento através da opção *-H*.

**Argumentos opcionais** Os argumentos seguintes são invocados opcionalmente, conforme necessidade do utilizador:

**-V ou –version** Usado para consultar a versão do módulo.

**-A ou –author** Usado para consultar os dados do autor.

**-d ou –dnsserver** Usado para passar como argumento o endereço IP do servidor de DNS onde a consulta deve ser feita, por omissão é usado o servidor de DNS da google com o endereço IP 8.8.8.8.

**Exemplo de execução em linha de comando**

```
./check_dns.py -H www.ciencias.ulisboa.pt -I 194.117.42.133
```

**Status** Em resposta à execução do módulo, o Nagios exibe um dos seguintes estados:

**OK** Exibido quando a consulta ao servidor de DNS retornar os valores esperados, ou seja o endereço IP retornado da consulta DNS corresponder com o IP passado como argumento por meio da opção *-I*.

**CRITICAL** Quando o endereço IP retornado pela consulta ao servidor de DNS for diferente do endereço IP passado como argumento.

**Dependências** Bash (dig)

**Sistema Operativo** Linux

**Testes** Este módulo foi testado nas seguintes distribuições Linux:

Xubuntu-16-04.01

Lubuntu 16.10

Gentoo 2.2

CentOS 7

Debian 8

**Código** Anexo A.5

## 4.6 Monitorização de listas negras

**Introdução** As listas negras [20] são uma das mais populares ferramentas de combate ao crescente problema de *spam* e tentativas de *phishing* utilizando o correio eletrónico. Estas listas são normalmente implementadas sob a forma de registos DNS. Neste modelo, um servidor de correio eletrónico é inserido numa lista negra, adicionando o seu endereço IP à um domínio gerido pela lista, num formato previamente convencionado.

Para verificar a confiabilidade de um servidor, as ferramentas antispam efetuam consulta ao servidor de DNS, pesquisando o endereço IP do servidor de correio eletrónico no formato definido pela lista. O conteúdo do registo retornado é irrelevante uma vez que a sua existência no servidor de DNS sinaliza a presença do servidor na lista negra. Estas listas são geridas por entidades privadas, e as políticas de inserção de endereços são suscetíveis à falhas e interpretações erróneas do comportamento dos servidores de correio eletrónico.

**Problemas de segurança** A entrada dos servidores de correio eletrónico em listas negras deve ser uma preocupação constante dos administradores destes sistemas. Este processo pode ocorrer quando uma ou mais contas de correio eletrónico são atacadas e infetadas, passando a fazer parte de uma Rede de *SPAMMERS* normalmente enviando centenas ou milhares de correios indesejáveis.

Tipicamente, mensagens enviadas por servidores de correios eletrónicos em listas negras são liminarmente rejeitadas ou marcadas pelas ferramentas de deteção de correio indesejado. A presença de um servidor de correio eletrónico legítimo numa lista negra tem por isso consequências negativas na produtividade e reputação da instituição, que importa tomar conhecimento e resolver com a maior brevidade possível.

**Solução** Com vista a manter um controle dos endereços dos servidores responsáveis pelo envio de correio eletrónico, foi desenvolvido o módulo ***check_dnsbl***, capaz de detetar estes incidentes, reproduzindo o comportamento dos filtros *antispam*, para um endereço IP recebido como argumento. Por omissão, o módulo pesquisa em 27 das listas negras mais populares, podendo ainda ser adicionadas outras listas ou ignorado o conjunto de listas negras predefinidas. Este módulo assinala a presença do endereço IP do servidor em pelo menos uma das listas negras com o estado *CRITICAL*.

#### O módulo

**Nome** check_dnsbl.py

**Linhas de código** 164 linhas (144 sloc)

**Linguagem de programação** Python 3

**Bibliotecas python** os, sys, OptionParse fron optparse, resolver from dns, urllib.request, OrderedDict from collections, repeat from itertools e ipaddress.

**Argumentos obrigatórios** O argumento seguinte deve ser especificado quando o módulo for executado:

**-H ou –hostaddress** Usado para especificar o endereço IP do servidor de correio eletrónico.

**Argumentos opcionais** Os argumentos seguintes são invocados opcionalmente, conforme necessidade do utilizador:

**-l ou –list** Usado para adicionar uma ou mais listas negras ao conjunto de listas pré-instaladas.

**-i ou –ignore** Usado para ignorar o conjunto de listas negras pré-instaladas.

**-I ou –Ignore** , usado para ignorar uma ou mais listas negras do conjunto de listas negras pré-instaladas.

**-V ou –version** Usado para consultar a versão do módulo.

**-A ou –author** Usado para consultar os dados do autor.

**Exemplo de execução em linha de comando**

```
./check_dnsbl.py -H 194.117.42.133
```

**Status** Em resposta a execução do módulo, o Nagios exibe um dos seguintes estados:

**OK** Exibido quando o resultado da pesquisa nas listas negras indicar a não inclusão do endereço do servidor de correio eletrónico em qualquer uma delas.

**CRITICAL** Exibido quando o endereço IP do servidor de correio eletrónico for encontrado em uma ou mais listas negras.

**Sistema operativo** Linux

**Testes** Este módulo foi testado nas seguintes distribuições Linux:

Xubuntu-16-04.01

Lubuntu 16.10

Gentoo 2.2

CentOS 7

Debian 8

**Código** Anexo A.6

## 4.7 Monitorização das configurações e validade do DNSSEC

**Introdução** *Domain Name System Security Extensions* (DNSSEC) é o nome usado para designar a extensão de segurança do protocolo DNS, concebida para proteger e autenticar o tráfego DNS.

Esta extensão é usada para validar os dados por meio de assinaturas digitais, recorrendo ao uso da tecnologia de criptografia assimétrica para assegurar a autenticidade e a integridade da informação trocada entre servidores de DNS, e entre estes e as aplicações cliente [3].

O DNSSEC prevê mecanismos de segurança complementares e transparentes para o utilizador, o que proporciona a capacidade de não interferir com o normal funcionamento do protocolo DNS.

O uso do DNSSEC proporciona melhorias na confiabilidade dos utilizadores nos serviços prestados, nomeadamente:

- Melhora a fiabilidade do sistema;
- Evita a ocorrência de ataques *man-in-the middle*[3] e *cache poisoning*[4];
- Corrige fragilidades do protocolo DNS;
- Diminui a probabilidade de manipulação de informação;
- Reforça a segurança e consequentemente presta um serviço seguro.

**Problemas de segurança** A Figura 4.2 ilustra o ataque *Cache Poisoning* ao servidor DNS, que pode ser evitado usando DNSSEC devidamente configurado. Segundo dados da Verisign LABS[5], (organização de pesquisa na área de Ecossistema do DNS, Ciência de dados/aprendizagem automática da Segurança e estabilidade da Internet), atualmente as três zonas que mais usam o DNSSEC são a .com, com cerca de 677,292 domínios, seguida pela .net com 105,991, e na terceira posição a .edu com cerca de 67 domínios [36]. A Figura 4.3 representa o crescimento anual de sítios protegidos com a extensão de segurança do protocolo do protocolo de DNS.

**Solução** Para auxiliar na deteção de problemas de configuração, assinaturas expiradas ou falta da extensão de segurança, foi desenvolvido o módulo ***check_dnssec***, que monitoriza o estado das configurações do protocolo DNS e sinaliza com o estado *CRITICAL* caso sejam detetadas vulnerabilidades. O domínio a ser monitorizado é passado como argumento, e pode ser definido o servidor de DNS a ser usado, já que, por omissão é usado o servidor de DNS da google com o endereço IP *8.8.8.8*.

[3]O ataque Man-In-The-Middle (MITM), ocorre quando um atacante escuta a comunicação entre duas partes e consegue personificar uma delas. A ferramenta geralmente usada para concretizar este ataque é chamada de *sniffer*.

[4]O ataque cache poisoning, literalmente traduzido para ataque de envenenamento, ocorre quando utilizadores maliciosos conseguem comprometer servidores de DNS, redirecionando nomes de domínios para endereços de servidores comprometidos, que ativam e exploram vulnerabilidades das vitimas, como por exemplo infetar o computador com spyware.

[5]`https://www.verisign.com/`

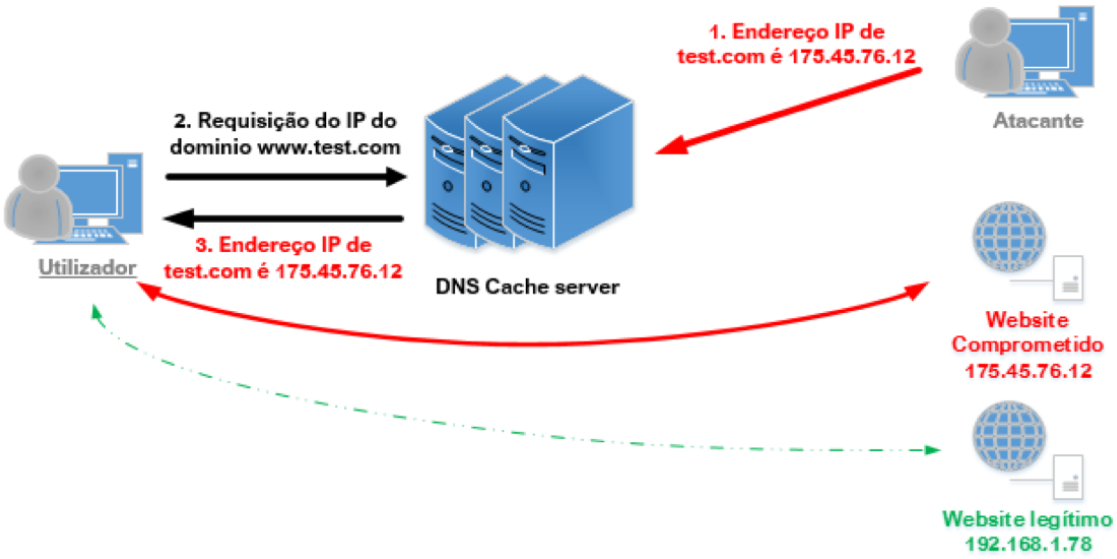


Fig. 4.2: DNS Cache Poisoning Attack [22]

**O módulo**

**Nome** check_dnssec.py

**Linhas de código** 117 linhas (105 sloc)

**Linguagem de programação** Python 3

**Bibliotecas python** os, sys, requests, OptionParse fron optparse, urllib.request e socket.

**Argumentos obrigatórios** O argumento seguinte deve ser especificado quando o módulo for executado:

**-H ou –domain** Usado para especificar nome do domínio a ser monitorizado.

**Argumentos opcionais** Os argumentos seguintes são invocados opcionalmente, conforme necessidade do utilizador:

**-d ou –dnsserver** Usado para especificar o endereço IP do servidor de DNS a ser usado, por omissão a consulta é feita ao servidor de DNS da google com o endereço IP *8.8.8.8*.

**-V ou –version** Usado para consultar a versão do módulo.

**-A ou –author** Usado para consultar os dados do autor.

**Exemplo de execução em linha de comando**

```
./check_dnssec.py -H www.ciencias.ulisboa.pt
```

**Status** Em resposta a execução do módulo, o Nagios exibe um dos seguintes estados:

**OK** Exibido quando a consulta determinar a existência do mecanismo DNSSEC devidamente configurado, e com assinaturas válidas.

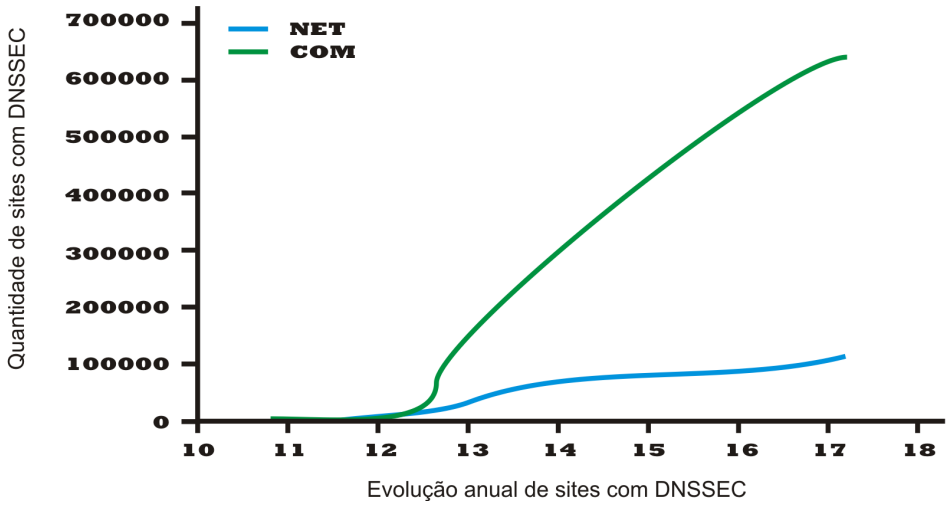


Fig. 4.3: Evolução do uso do DNSSEC de 2010 a 2017 [36]

**CRITICAL** Quando os dados retornados pela consulta determinarem debilidades no DNSSEC, nomeadamente incorreções na configuração, e/ou assinaturas inválidas, e/ou ausência de DNSSEC.

**Dependências** Bash (dig)

**Sistema operativo** Linux

**Testes** Este módulo foi testado nas seguintes distribuições Linux:

Xubuntu-16-04.01

Lubuntu 16.10

Gentoo 2.2

CentOS 7

Debian 8

**Código** Anexo A.7

## 4.8 Monitorização de ficheiros e diretorias

**Introdução** Ficheiros são agrupamentos de registos que seguem uma regra estrutural, e podem conter dados, documentos ou programas, armazenados em uma diretoria. Diretoria é um mecanismo que permite organizar os ficheiros de acordo com os assuntos a que se relacionam [26].

Vários são os ficheiros/diretorias, de sistema ou não, cujo estado deve manter-se inalterado. A sua alteração pode indicar problemas de segurança, e a sua monitorização é uma óptima solução para evitar que estas alterações passem despercebidas aos administradores.

**Problemas de segurança** Após a instalação de programas como por exemplo o Sistema de Administração de Conteúdos (*Content Management System*, CMS) WordPress, várias são as diretorias e ficheiros desta aplicação que devem manter o seu estado inalterado, e cuja alteração é um forte indício de ação maliciosa. A título de exemplo, a substituição da pagina index.html ou index.php, implica necessariamente a alteração do estado deste ficheiro, e em geral, ataques perpetrados por *defacers* passam pela alteração deste ficheiro, ou a sua substituição por um de autoria dos atacantes.

**Solução** O módulo ***check_filechange*** foi projetado para responder a esta necessidade, alertando sobre a ocorrência de um evento relacionado com a alteração do estado do ficheiro/diretoria. O módulo utiliza a ferramenta de monitorização de estado *Inotify-tools*[6] [21], recebendo como argumentos os estados e os ficheiros/diretorias a serem monitorizados, assim como o caminho do ficheiro de log onde o Inotify regista os eventos. O módulo retorna o estado *CRITICAL* caso um evento seja registado no ficheiro de log como resultado da subversão dos parâmetros definidos.

O Inotify-tools é um utilitário que prevê eventos de notificação provenientes do sistema de ficheiros, notificando modificações destes, tais como abrir, fechar, mover, renomear, apagar, criar ou alterar atributos [21]. Basicamente o módulo desenvolvido lança o Inotifywait em modo *Daemon*, e este passa a escrever num ficheiro de log a ser criado e configurado manualmente no momento da instalação. Neste ficheiro serão armazenados os registos sobre eventos dos ficheiros monitorizados. Tratando-se de um ficheiro de log, a sua localização por omissão deverá ser na diretoria /var/log/.

Cada vez que o módulo é invocado, este verifica se o Inotifywait esta a ser executado como *Daemon*. Caso não esteja, é lançado, proporcionando assim fiabilidade dos seus resultados, e tirando a responsabilidade ao administrador de verificar constantemente se a ferramenta esta a ser executada como é previsto.

### O módulo

**Nome** check_filechange.py

**Linhas de código** 114 linhas (100 sloc)

**Linguagem de programação** Python 2

[6] https://github.com/rvoicilas/inotify-tools/wiki

**Bibliotecas python** os, sys, OptionParse fron optparse, strftime fron time, time e datetime.

**Argumentos obrigatórios** Os argumentos seguintes devem ser especificados quando o módulo for executado:

**-p ou –path** Usado para especificar o caminho completo do ficheiro de log do Inotify.

**-e ou –events** Usado para especificar os eventos a serem monitorizados.

**-f ou –files** Usado para especificar as diretorias ou ficheiros a serem monitorizados.

**Argumentos opcionais** Os argumentos seguintes são invocados opcionalmente, conforme necessidade do utilizador:

**-V ou –version** Usado para consultar a versão do módulo.

**-A ou –author** Usado para consultar os dados do autor.

**Exemplo de execução em linha de comando**

```
./check_filechange -p /var/log/inotify/inotify.log -f /home/cgs/
-e access,open
```

**Status** Em resposta a execução do módulo, o Nagios exibe um dos seguintes estados:

**OK** Exibido quando nenhum evento associado aos ficheiros monitorizados for verificado.

**CRITICAL** Exibido quando um ou mais eventos associados aos ficheiros monitorizados forem verificados.

**Dependências** inotify-tools, tail, cat, e ps.

**Sistema operativo** Linux

**Testes** Este módulo foi testado nas seguintes distribuições Linux:

Xubuntu-16-04.01

Lubuntu 16.10

Gentoo 2.2

CentOS 7

Debian 8

**Código** Anexo A.8

## 4.9 Monitorização de vulnerabilidades web

**Introdução** Uma vulnerabilidade é um defeito que pode ser explorado por um atacante com o objetivo de subverter a política de segurança [10]. As vulnerabilidades podem existir por diversas razões. Exemplos comuns são a falta de tratamento pelo servidor de partes do conteúdo do pedido, *cookies* mal configurados, identificadores de sessão expostos no navegador, parâmetros passados pela URL ou num formulário de *Login*. Todos estes casos caracterizam-se por uma falha de segurança, expondo o servidor à ataques específicos.

Um *scanner* de vulnerabilidades web, é uma ferramenta que dado um determinado alvo web, é capaz de analisá-lo em busca de vulnerabilidades conhecidas. O processo consiste em testar sistematicamente o alvo em busca de vulnerabilidades suscetíveis de serem exploradas por atacantes, tais como senhas padrão, serviços inseguros escutando em portos públicos, sistemas vulneráveis e outros tipos de falhas conhecidas.

**Problemas de segurança** A existência de vulnerabilidades em um servidor web, representa um grosseiro problema de segurança. Isto porque para a maior parte das vulnerabilidades web conhecidas existe uma oferta enorme de ferramentas livres e comerciais capazes de as explorar com maior ou menor dificuldade.

As 10 vulnerabilidades de segurança mais críticas em aplicações web em 2017, segundo o *Open Web Application Security Project*, OWASP[7] (comunidade online dedicada a criar e disponibilizar de forma gratuita artigos, metodologias, documentação, ferramentas e tecnologias no campo da segurança de aplicações web), são [32]:

1. Injeção[8];
2. Autenticação e gestão de sessões fracas[9];
3. *Cross Site Scripting* (XSS)[10];
4. Falta de controlo de acesso adequado[11].
5. Configuração insegura[12];

[7] `https://www.owasp.org`

[8] **Injeção** é um problema de segurança que ocorre quando os dados não confiáveis são enviados para um interpretador como parte de um comando ou consulta. A informação maliciosa fornecida pelo atacante engana o interpretador que irá executar comandos mal intencionados ou manipular conteúdo não autorizado.

[9] **Autenticação e gestão de sessões fracas** ocorre quando as credenciais de acesso e *token* de sessão não são protegidos apropriadamente. Atacantes comprometem senhas, chaves ou *tokens* de autenticação de forma a assumir a identidade de outros utilizadores.

[10] Problemas de **XSS** ocorrem quando uma aplicação recebe dados e não os valida adequadamente. A exploração desta vulnerabilidade permite que utilizadores maliciosos executem código no navegador e consigam usurpar sessões do utilizador e redireciona-lo para sis maliciosos ou desfigurar sítios.

[11] **Falta de controlo de acesso adequado** ocorre quando uma aplicação recebe requisições e não verifica a identidade do solicitante antes de permitir o acesso. O que permitiria que um utilizador com más intenções fosse capaz de aceder a dados ou funcionalidades para quais não tem autorização

[12] **Configuração insegura** é um problema de segurança bastante frequente, uma vez que as configurações padrão geralmente são fracas, e ocorre quando não são implementadas corretamente as configurações de uma aplicação, como *frameworks*, servidor de aplicação, servidor web, base de dados, etc.

6. Armazenamento criptográfico inseguro[13];
7. Falta de proteção contra ataques[14];
8. *Cross Site Request Forgery* (CSRF)[15];
9. Uso de componentes com vulnerabilidades conhecidas[16];
10. Interface de programação de aplicações desprotegidas[17];

**Solução** Foi desenvolvido o módulo ***check_nikto***, que usa o *scan* de vulnerabilidades web NIKTO para auditar sítios em busca de vulnerabilidades. Entre as vulnerabilidades verificadas, destacam-se os *componentes mal configurados, divulgação de informação sensível, injeção (XSS/Script/HTML), negação de serviço, execução de comandos, injeção de SQL*. O módulo produz um relatório em html, e alerta para existência de vulnerabilidades conhecidas [14], retornando o estado *CRITICAL* em caso de deteção.

O domínio a ser monitorizado é passado como argumento, e opcionalmente pode ser especificado o porto de comunicação, a diretoria onde será armazenado o relatório, o nome e tempo de validade do relatório, assim como os tipos de vulnerabilidades a serem pesquisadas.

O NIKTO lista e classifica as vulnerabilidades segundo a codificação da extinta *Open Source Vulnarabilities Data Base (OSVDB)* com equivalência na *Common Vulnerabilities and Exposures(CVE).*

O plugin é estruturado de formas a executar o nikto nas seguintes condições:

1. Se não for encontrado na diretoria especificada o relatório com o nome passado como argumento através da opção *-r*;
2. Se o relatório encontrado na diretoria especificada tiver tempo de vida superior ao passado no argumento *-t*.

Por outro lado, a leitura e apresentação do relatório é feita nos seguintes casos:

1. Se o relatório tiver tamanho superior a zero bytes;

[13]**Armazenamento criptográfico inseguro** é uma vulnerabilidade que acontece quando aplicações não protegem devidamente dados sensíveis, como por exemplo, *id's*, credenciais de autenticação ou dados de cartões de credito. Um utilizador malicioso pode tirar partido desta falta de proteção para usurpar e fazer uso destas informações sensíveis. O uso de criptografia adequada é um bom principio para evitar esta falha de segurança.

[14]**Falta de proteção contra ataques** é o tipo de vulnerabilidade que ocorre quando as aplicações não têm capacidade para detetar, prevenir e responder a ataques manuais ou automáticos.

[15]**CSRF** acontece quando um atacante consegue forçar o navegador da vítima, que esteja autenticada em uma aplicação, a enviar uma requisição pré-autenticada a um servidor Web vulnerável, que por sua vez força o navegador da vítima a executar uma ação maliciosa em prol do atacante. O CSRF pode ser tão poderoso quanto a aplicação Web que ele ataca.

[16]**Uso de componentes com vulnerabilidades conhecidas** é uma vulnerabilidade que existe quando componentes como bibliotecas, frameworks, e outros módulos de software quase sempre são executados com privilégios elevados. As aplicações que utilizam componentes com vulnerabilidades conhecidas podem minar as suas defesas e permitir uma gama de possíveis ataques e impactos.

[17]**APIs desprotegidas** é atualmente uma vulnerabilidade bastante comum porque são várias as aplicações que recorrem a Interface de Programação de Aplicações para desempenharem suas funções. Essas APIs são muitas vezes desprotegidas ou com proteção inadequada e contêm numerosas vulnerabilidades.

2. Caso o tamanho do relatório seja diferente de zero, o plugin verifica se o *scan* esta concluído, e só se estiver concluído é retornado o estado *OK* se nenhuma vulnerabilidade for listada, ou o estado *CRITICAL* seguida da quantidade e os *IDs* das vulnerabilidades, caso uma ou mais vulnerabilidades sejam encontradas.

**O Módulo**

**Nome** check_nikto.py

**Linhas de código** 197 linhas (186 sloc)

**Linguagem de programação** Python 3

**Bibliotecas python** os, sys, OptionParse fron optparse, time, socket, datetime from datetime, Counter from collections, codecs, urllib.request, re e urllib.

**Argumentos obrigatórios** O argumento seguinte deve ser especificado quando o módulo for executado:

**-H ou –host** Usado para especificar o nome do domínio a ser verificado.

**Argumentos opcionais** Os argumentos seguintes são invocados opcionalmente, conforme necessidade do utilizador:

**-P ou –path** Usado para especificar a diretoria onde o relatório será armazenado.

**-p ou –port** Usado para especificar o porto de comunicação do alvo.

**-r ou –report** Usado para atribuir um nome ao relatório da verificação.

**-t ou –time** Usado para especificar tempo de validade (em dias) do relatório.

**-T ou –tuning** Usado para especificar os tipos de vulnerabilidades a serem procuradas no domínio.

**-V ou –version** Usado para consultar a versão do módulo.

**-A ou –author** Usado para consultar os dados do autor.

**Exemplo de execução em linha de comando**

```
./check_nikto.py -H ciencias.ulisboa.pt -p 80 -r ciencias
-t 2 -T 9
```

**Status** Em resposta a execução do módulo, o Nagios exibe um dos seguintes estados:

**OK** Exibido quando nenhuma vulnerabilidade for encontrada.

**CRITICAL** Exibido quando uma ou mais vulnerabilidade forem detetadas.

**Dependências** Nikto2.

**Sistema operativo** Linux

**Testes** Este módulo foi testado nas seguintes distribuições Linux:

Xubuntu-16-04.01

Lubuntu 16.10

Gentoo 2.2

CentOS 7

Debian 8

**Código** Anexo A.9

## 4.10 Monitorização de portos de comunicação

**Introdução** Porto de comunicação é o número que identifica uma aplicação à qual se destinam os dados de uma comunicação. Desta maneira, os dados são enviados diretamente para a aplicação correspondente. Uma vez que estes são os pontos de entrada e saída de informação no sistema, portos abertos fora do controlo dos administradores podem representar uma janela para que utilizadores maliciosos possam aceder ao sistema e extrair ou executar o que lhes interessar.

**Problemas de segurança** A existência de portos abertos fora do controlo do administrador representa assim um problema de segurança que deve merecer a atenção de qualquer administrador. Um atacante após ganhar acesso ao sistema pode optar por instalar uma *backdoor*, para possibilitar a sua entrada em outras ocasiões. Nesse caso, o software de *backdoor* necessitará de estar à escuta num porto específico, não utilizado por aplicações convencionais.

**Solução** O módulo ***check_open_port*** foi desenvolvido para monitorizar os portos, e alertar sobre a existência de portos abertos que não constam da lista de portos autorizados. Esta lista é passada como argumento, e indica os portos abertos sobre controlo dos administradores. É igualmente passado o endereço IP da máquina a ser monitorizada, retornando o estado *CRITICAL* caso um ou mais portos abertos sejam encontrados.

**O Módulo**

**Nome** check_open_port.py

**Linhas de código** 93 linhas (82 sloc)

**Linguagem de programação** Python 2

**Bibliotecas** sys, OptionParse fron optparse, socket e subprocess.

**Argumentos obrigatórios** Os argumentos seguintes devem ser especificados quando o módulo for executado:

**-H ou –hostaddress** Usado para especificar o endereço IP a ser verificado.

**-p ou –port** Usado para especificar os portos autorizados a estarem abertos.

**Argumentos opcionais** Os argumentos seguintes são invocados opcionalmente, conforme necessidade do utilizador:

**-V ou –version** Usado para consultar a versão do módulo.

**-A ou –author** Usado para consultar os dados do autor.

**Exemplo de execução em linha de comando**

```
./check_open_port.py -H 192.168.2.35 -p 21,25,80
```

**Status** Em resposta a execução do módulo, o Nagios exibe um dos seguintes estados:

**OK** Exibido quando nenhum porto diferente dos passados como argumento através da opção *-p* for encontrado aberto.

**CRITICAL** Exibido quando forem encontrados um ou mais portos que não constem da lista de portos passados como argumento.

**Dependências** Não aplicável.

**Sistema operativo** Linux

**Testes** Este módulo foi testado nas seguintes distribuições Linux:

Xubuntu-16-04.01

Lubuntu 16.10

Gentoo 2.2

CentOS 7

Debian 8

**Código** Anexo A.10

## 4.11 Monitorização das configurações SSL/TLS

**Introdução** O *Secure Sockets Layer* (SSL) é um protocolo criptográfico projetado para garantir autenticidade, confidencialidade e integridade na troca de dados entre aplicações cliente/servidor, e tem como sucessor o *Transport Layer Security* (TLS). Este protocolo usa certificados digitais (ficheiros eletrónicos autenticados com assinatura digital) para cifrar e proteger a troca de dados. Os certificados SSL/TLS necessitam de cuidados na sua administração e utilização. Estes podem ser revogados, estar expirados, usar protocolos criptográficos inseguros, estar mal configurados, entre outras debilidades suscetíveis de tornar a comunicação insegura.

Maior parte dos navegadores já estão adaptados para aceitar, validar e interagir com sítios que tenham certificados SSL emitidos por Autoridades de Certificação (*Certificate Authority*, CA) [18].

**Problemas de segurança** Considerando a função do protocolo SSL explanada no paragrafo anterior, torna-se evidente que o não uso deste torna a comunicação vulnerável e portanto insegura do ponto de vista da privacidade, autenticidade e confidencialidade.

Por outro lado, quando os certificados usados são revogados, encontram-se expirados, usam protocolos criptográficos inseguros para troca de chaves, ou se encontram mal configurados, tornam-se num alvo bastante fácil e apetecível para os utilizadores maliciosos. Nestas condições, os certificados podem criar uma falsa sensação de segurança em ambas as partes legítimas de uma comunicação.

**Solução** Para auxiliar na deteção dos problemas de segurança acima mencionados, foi desenvolvido o módulo ***check_ssl***. Este módulo usa o *web service* do *SSLLAB*[19], para realizar testes aos certificados em busca de vulnerabilidades. Caso estas sejam detetadas é retornado o estado *WARNING* ou *CRITICAL*, em função dos argumentos definidos, nomeadamente o nome do domínio e os valores para *warning* e *critical*. Os argumentos para as opções *critical* e *warning* se baseiam na tabela de classificação do SSLLAB, e na quantidade de dias antes da expiração do certificado.

O *SSLLAB* adotou uma classificação de segurança dos certificados que varia entre *A* e *F*[20]. Porém, recentemente foram adicionadas mais duas classificações uma destinada a casos em que existe incompatibilidade de nomes (*M*) e outra (*T*) atribuída quando o certificado é assinado por entidade privada ou é auto-assinado[21].

Caberá ao administrador de sistema definir quais classificações deverá enquadrar como *OK*, *WARNING* ou *CRITICAL*.

### O Módulo

**Nome** check_ssl.py

**Linhas de código** 227 linhas (200 sloc)

[18] Autoridade de Certificação é a entidade que detém a responsabilidade de emitir, revogar, manter e disponibilizar a informação dos certificados.

[19] `https://www.ssllabs.com/`

[20] `https://github.com/ssllabs/research/wiki/SSL-Server-Rating-Guide`

[21] `https://blog.qualys.com/ssllabs/2014/06/17/ssl-labs-new-grades-for-trust-t-and-mis`

**Linguagem de programação** Python 3

**Bibliotecas** os, sys, OptionParse fron optparse, json, urllib e urllib2, ssl, socket, datetime, time e requests.

**Argumentos obrigatórios** O argumento seguinte deve ser especificado quando o módulo for executado:

**-H ou –domain** Usado para especificar o nome do domínio.

**Argumentos opcionais** Os argumentos seguintes são invocados opcionalmente, conforme necessidade do utilizador:

**-d ou –days** Usado para especificar o número de dias antes da data de expiração a partir do qual deve ser retornado o estado *warning*.

**-c ou –critical** Usado para especificar quais classificações do ssllab deverão ser interpretadas como *critical*.

**-w ou –warning** Usado para especificar quais as classificações do ssllab deverão ser interpretadas como *warning*.

**-s ou – sleep** Usado para especificar o tempo em que o módulo deverá esperar enquanto está em execução.

**-V ou –version** Usado para consultar a versão do módulo.

**-A ou –author** Usado para consultar os dados do autor.

**Exemplo de execução em linha de comando**

```
./check_ssl.py -H www.ciencias.ulisboa.pt
```

**Status** Em resposta a execução do módulo, o Nagios exibe um dos seguintes estados:

**OK** exibido quando a classificação retornada pela consulta à API, não constar da lista passada como argumento através das opções *-w ou -c*.

**WARNING** Exibido quando a classificação retornada pela consulta à API, constar da lista passada como argumento através da opção *-w*, ou quando a quantidade de dias até a expiração do certificado for inferior ao valor passado como argumento através da opção *-d*.

**CRITICAL** Exibido quando a classificação retornada pela consulta à API, constar da lista passada como argumento através da opção *-c*.

**Dependências** *Web service* SSLLAB[22] .

**Sistema operativo** Linux

**Testes** Este módulo foi testado nas seguintes distribuições Linux:

Xubuntu-16-04.01

Lubuntu 16.10

[22] https://www.ssllabs.com/

Gentoo 2.2

CentOS 7

Debian 8

**Código** Anexo A.11

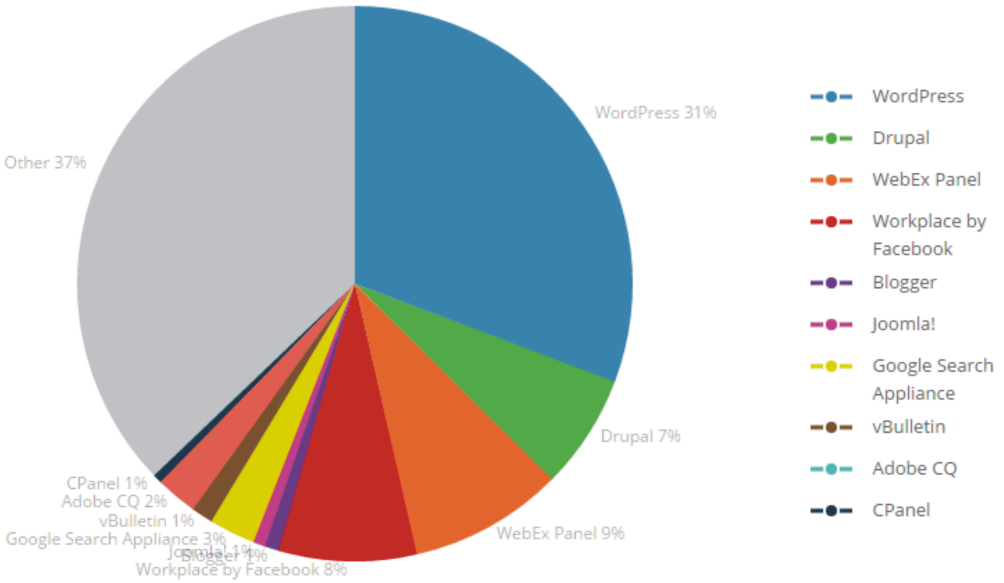


Fig. 4.4: Estatísticas de uso da tecnologia CMS pelo mundo [4]

## 4.12 Monitorização de atualizações do WordPress

**Introdução** O WordPress[23] é um Sistema de Administração de Conteúdo *(Content Management System, CMS)*, concebido para facilitar a criação e administração robusta de sítios na Internet, blogues ou aplicações, assegurando usabilidade e estética. O WordPress é um dos CMS com maior quota de mercado, como mostra a Figura 4.4.

Ele permite o uso de centenas de aplicações de terceiros (temas e plugins) que melhoram a sua flexibilidade e usabilidade, fazendo dele uma poderosa ferramenta de administração de Conteúdo web.

É um software desenvolvido por uma comunidade aberta, e por isso está em constante aperfeiçoamento, o que o torna suscetível a vulnerabilidades aplicacionais ou de projeto, que são frequentemente corrigidas por versões de segurança, disponibilizadas no seu sítio oficial na Internet. No entanto, apesar de serem sempre que se justifique disponibilizadas versões para correções de segurança, do lado do utilizador a descoberta de novas atualizações nem sempre é um processo trivial, o que leva a que muitas vezes se utilizem por largos períodos de tempo versões desatualizadas e com vulnerabilidades conhecidas publicamente.

O WordPress usa o sistema de versionamento Semântico, representado por três grupos numéricos separados por ponto (*4.7.3*), onde o incremento de cada um deles tem o seguinte significado:

- O primeiro grupo é denominado Versão principal (***Major Version***). Representa a base do software, sendo incrementado sempre que ocorra melhorias que impliquem alteração na compatibilidade da API, atualmente identificada pelo número 4;
- O segundo grupo é denominado Versão menor (***Minor Version***). É incrementado quando são feitas alterações da interface, que impliquem mudanças ou melhorias

[23] `https://wordpress.org/`

no comportamento, ou quando é adicionada uma característica que não afeta o funcionamento padrão do software, atualmente identificada pelo número 8.

- O terceiro grupo é denominado de Versão de correções (***Patch Version***). Estes dígitos são incrementados quando uma atualização de segurança (correção de *bugs*) for disponibilizada com o único objetivo de corrigir falhas e vulnerabilidades. Tendo a versão 4.8 sido disponibilizada a pouco menos de um mês, ainda não foi lançada nenhuma versão de correção de erros.

**Problemas de segurança** Sistemas desatualizados estão sujeitos a serem atacados por via da exploração de vulnerabilidades já corrigidas com atualizações de segurança, e por isso publicamente conhecidas.

Por exemplo, o recente ciberataque à escala global com o *Ransomware WannaCry*, ocorrido em Maio de 2017, provou uma vez mais não ser trivial manter os sistemas atualizados, uma vez que este ataque explorou uma vulnerabilidade em sistemas operativos Windows corrigida dois meses antes com a atualização *MS-17-010* [1], e mesmo assim várias foram as organizações espalhadas pelos quatro cantos do planeta apanhadas desprotegidas.

**Solução** O módulo ***check_wp_update*** foi desenvolvido para lidar com o problema de descoberta de atualizações do CMS WordPress. Recebe como argumento o caminho completo do ficheiro version.php, de onde obtém informações sobre a versão do software instalado, e usa preferencialmente o *web service*[24] para obter a última versão disponível no sítio oficial do WordPress. Alerta com o estado *CRITICAL* sempre que uma nova versão ou atualização seja detetada.

Adicionalmente é feito um *parse* ao conteúdo html da pagina de *download* do WordPress[25], sendo que esta opção é usada apenas no caso pouco provável de falha na invocação do *web service*.

O módulo considera *critical* qualquer versão que não seja a mais recente, pois as versões do WordPress, têm no mínimo correções de erros. Ou seja, todas as versões principais além de trazerem melhorias no desempenho e novas funcionalidades, têm correções de bugs e melhorias na segurança, o que torna um ótimo princípio atualizar o CMS para a verão mais recente logo que for disponibilizada.

Além disso, jogam a favor o facto de problemas de compatibilidades que em muitos casos fazem com que os administradores tenham reticencias na hora de prosseguir com a atualização, serem contornáveis, uma vez que as versões do WordPress são amplamente testadas antes de serem disponibilizadas, e serem facilmente encontradas soluções para possíveis incompatibilidades.

No entanto, isso não dispensa que se façam cópias de segurança antes de avançar para a atualização.

### O Módulo

**Nome** check_wp_update.py

[24] `https://api.wordpress.org/core/version-check/1.7/`
[25] `https://wordpress.org/download`

**Linhas de código** 152 linhas (136 sloc)

**Linguagem de programação** Python 3

**Bibliotecas python** os, sys, OptionParse fron optparse, json, re, request e urllib.

**Argumentos obrigatórios** O argumento seguinte deve ser especificado quando o módulo for executado:

**-p ou –path** Usado para especificar o caminho completo para o ficheiro version.php.

**Argumentos opcionais** Os argumentos seguintes são invocados opcionalmente, conforme necessidade do utilizador:

**-V ou –version** Usado para consultar a versão do módulo.

**-A ou –author** Usado para consultar os dados do autor.

**Exemplo de execução em linha de comando**

```
./check_wp_update.py -p /var/www/html/wp-includes/version.php
```

**Status** Em resposta a execução do módulo, o Nagios exibe um dos seguintes estados:

**OK** Exibido quando a versão retornada pelo *web service* for igual a versão instalada.

**CRITICAL** Exibido quando a versão retornada pelo *web service* for superior a versão instalada.

**Dependências** *Web service* do WordPress.

**Sistema operativo** Linux

**Testes** Este módulo foi testado nas seguintes distribuições Linux:

Xubuntu-16-04.01

Lubuntu 16.10

Gentoo 2.2

CentOS 7

Debian 8

**Código** Anexo A.12

# Capítulo 5

# Produção, Avaliação e Discussão

## 5.1 Passagem para ambiente de produção

Esta secção é reservada a abordagem dos resultados esperados e em alguns casos obtidos após passagem dos módulos desenvolvidos para ambiente de produção em Ciências.

Tendo em conta que os módulos desenvolvidos visam alertar sobre problemas de segurança, a implementação destes veio acelerar os processos de deteção e resposta a incidentes de segurança, pelo que se espera que em caso de eventos relacionados com os módulos aqui abordados possamos ter resposta mais eficiente.

#### Distribuição dos módulos

Depois de desenvolvidos e testados exaustivamente em ambiente de teste, houve a necessidade de preparar os módulos para que fossem facilmente configurados em ambiente de produção. Uma vez que o sistema de monitorização de Ciências alberga dezenas de máquinas, este processo afigurou-se uma tarefa pouco trivial.

A solução para tornar esta tarefa mais amigável foi a conversão dos módulos desenvolvidos em pacotes instaláveis, deixando assim de haver a necessidade de instalar manualmente as dependências de cada módulo.

**Pacotes python** A primeira opção foi converter os módulos em pacotes python usando a ferramenta distutils. O distutils baseia-se essencialmente no ficheiro setup.py que importa os módulos do distutils, acerta os parâmetros e invoca a função setup() com os parâmetros necessários (Anexo B.1). O ficheiro resultante é multiplataforma e de simples instalação, bastando descompactar, e na diretoria dos pacotes executar o comando `# python setup.py install`, instalando assim o módulo e todas as dependências especificadas.

**Pacotes RPM** A segunda solução, foi concebida pensando exclusivamente no sistema de monitorização de Ciências, e para tal foram criados pacotes RPM (*Red Hat Package Manager*) a fim de tirar partido do gestor de pacotes *yum*[1], sobre o qual assenta a quase

[1] O yum é um gestor de pacotes RPM *Package Management da Red Hat*, que permite o download, instalação, configuração, atualização e remoção de software de forma simplificada, preservando as dependências entre softwares.

totalidade dos pacotes Linux geridos.

Nesta opção, os parâmetros são todos definidos num ficheiro .spec (Anexo B.2). Tal como na solução anterior são especificadas as dependências mínimas, e estas são resolvidas no ato da instalação dos módulos, caso não estejam disponíveis na máquina em questão.

Especialmente na conservação dos módulos em pacotes RPM, foram passados parâmetros que permitiram instalar as dependências mínimas de cada plugin, assim como o agente nagios-nrpe-server, já com as suas configurações básicas, tais como, endereço IP do servidor Nagios e comandos básicos para execução remota do módulo.

## 5.2 Avaliação

Esta secção é reservada para a apresentação dos resultados obtidos em ambiente de testes e de produção, analisando a capacidade de deteção de eventos e consequente notificação. Os testes a que os módulos foram submetidos tiveram em atenção não só as funcionalidades, mais também o tempo de resposta dos plugins, tendo em conta que estes não devem exceder os 10 segundos, conforme recomendações da comunidade Nagios [13].

Os módulos desenvolvidos foram testados em várias plataformas Linux, tais como, *Xubuntu-16-04.01, Lubuntu 16.10, Gentoo 2.2, CentOS 7 e Debian 8*, enquanto que em ambiente de produção estão instalados em plataformas Linux CentOS 7.

No âmbito dos testes houve necessidade de forçar cenários que simulassem situações que fizessem os módulos retornarem todos os estados possíveis. Os resultados obtidos foram bastante animadores, deixando indicadores que permitiram passá-los para ambiente de produção em Ciências.

Foram também submetidos ao repositório oficial do Nagios, onde após avaliação pela equipa responsável pela administração, foram aprovados, estando por isso publicados[2] e disponíveis para serem usados dentro dos padrões a que estão licenciados. Além disso, os módulos estão também publicados no repositório público github[3].

**Os módulos check_app, check_synflood, check_filechange e check_open_port** foram instalados no sistema de Ciências, estando no momento da escrita desta dissertação a monitorizar cerca de 182 servidores, 160 dos quais virtuais, não tendo até ao momento detetado qualquer situação anormal no funcionamento dos equipamentos monitorizados.

**O módulo check_apache_status** foi igualmente instalado e encontra-se a monitorizar cerca de **32** servidores web, não tendo até ao momento detetado qualquer situação anormal no funcionamento dos equipamentos monitorizados.

**O módulo check_defacement** com características reativas encontra-se igualmente instalado e a monitorizar o principal sítio de Ciências e embora não tendo até ao momento detetado qualquer situação anormal foi considerado pela equipa de administração como um dos mais úteis.

[2] `https://exchange.nagios.org/directory/Plugins/Security/`
[3] `https://github.com/rc48807`

**Os módulos check_dns, check_dnssec e check_ssl** estão em produção a monitorizar os **7** domínios mais relevantes. No caso do check_ssl alguns dos domínios monitorizados estão representados na Figura 5.1, não tendo até ao momento detetado qualquer situação anormal.

| Service | Status | Duration | Attempt | Status Information |
|---|---|---|---|---|
| [redacted] SSL Rating | OK | 3d 1h 23m 6s | 1/3 | OK - SSLLABS grade for https://[redacted].pt is A+ |
| Webmail SSL Rating | OK | 3d 1h 4m 57s | 1/3 | OK - SSLLABS grade for https://webmail.[redacted].pt is B |
| Suporte.[redacted] SSL Rating | OK | 2d 18h 41m 22s | 1/3 | OK - SSLLABS grade for https://suporte.[redacted].pt is A |
| Webpages.[redacted] SSL Rating | OK | 3d 1h 41m 49s | 1/3 | OK - SSLLABS grade for https://webpages.[redacted].pt is A |
| Webpages.[redacted] SSL Rating | OK | 3d 1h 36m 23s | 1/3 | OK - SSLLABS grade for https://webpages.[redacted].pt is A |

Fig. 5.1: Estado *"OK"* para os certificados

**O módulo check_dnsbl** encontra-se instalado e a monitorizar cerca de **3** servidores responsáveis pelo envio de correio eletrónico, sem qualquer deteção relevante até ao momento da escrita desta dissertação.

**Módulo check_nikto** com a função de pesquisar periodicamente vulnerabilidades web num domínio, em ambiente de produção este módulo foi capaz de identificar uma vulnerabilidade de *Clickjacking*, que embora não seja critica já foi corrigida, e a seguir descrevemos:

**Clickjacking** Cabeçalho HTTP X-frame-options em falta;

**Descrição** *Clickjacking*, vulnerabilidade também conhecida como "*UI redress attack*", é uma vulnerabilidade web que afeta vários navegadores web, tais como Firefox, Chrome, Internet Explorer, Opera e Safari. Ela existe quando é possível um atacante usar camadas transparentes para enganar um utilizador, fazendo com que este clique em um objeto (link ou botão) invisível pensando que está a clicar no objeto que deseja, quando na verdade a sua ação esta a ser usada para ações maliciosas [6].

**Soluções** Maior parte das vulnerabilidades resultam de implementações incorretas no código fonte da linguagem de programação. Diferente deste padrão, o *clickjacking* geralmente resulta da combinação do mau uso de algumas características dos recursos do HTML e CSS e a interação do utilizador com elementos transparentes.

Para os utilizadores é possível tomar alguns cuidados, tais como, manter o navegador atualizado e evitar clicar em anúncios de origem desconhecida. Enquanto que do lado dos desenvolvedores a utilização de quebra de *frames* e o uso de cabeçalho *HTTP X-frame-options* [19] são algumas das soluções adotadas.

**Módulo check_wp_update** foi capaz de alertar os administradores sobre a disponibilização de uma nova atualização de segurança do WordPress, e consequente desatualização da versão instalada.

**Descrição** O módulo detetou e gerou notificação quando foi disponibilizada a versão 4.7.4 do WordPress, momento em que a versão em uso era a 4.7.3. A Figura 5.2 representa o estado retornado pelo Nagios antes da deteção, enquanto que a Figura 5.3 representa o estado exibido no momento da deteção.

| Service | Status | Duration | Attempt | Status Information |
|---|---|---|---|---|
| WordPress Updates | OK | 2d 1h 20m 59s | 1/3 | The latest stable version WordPress 4.7.3 available in wordpress.org is installed |

Fig. 5.2: Estado *"OK"* para o CMS WordPress

**Soluções** A solução para este problema passou por tomar as providencias necessárias em véspera de atualização de um gestor de conteúdo web, e atualizá-lo.

| Service | Status | Last Check | Duration | Attempt | Status Information |
|---|---|---|---|---|---|
| CPU Usage | OK | 05-04-2017 15:53:21 | 34d 9h 12m 14s | 1/3 | CPU Usage : 4 % |
| HTTP | OK | 05-04-2017 15:59:19 | 34d 9h 6m 30s | 1/3 | HTTP OK: HTTP/1.1 302 Found - 312 bytes in 0.041 second response time |
| Home Partition | OK | 05-04-2017 16:02:13 | 98d 4h 51m 16s | 1/3 | SNMP OK - /home at 6% with 26.24 of 27.93 GB free |
| Memory Usage | OK | 05-04-2017 16:00:24 | 34d 9h 20m 26s | 1/3 | Memory usage : 3.37 GB used for a total of 3.85 GB (87%) with 1.65 MB in buffer and 958.19 MB in |
| Root Partition | OK | 05-04-2017 15:58:39 | 34d 9h 21m 57s | 1/3 | SNMP OK - / at 27% with 7.31 of 10.05 GB free |
| Updates | OK | 05-03-2017 16:13:04 | 0d 23h 53m 26s | 1/3 | CHECK_UPDATES OK - no updates available |
| WordPress Updates | CRITICAL | 05-03-2017 16:19:01 | 12d 23h 51m 28s | 3/3 | Version outdated, has installed WordPress 4.7.3, but is available in wordpress.org the version 4.7.4 |

Fig. 5.3: Estado *"CRITICAL"* para o CMS WordPress

### Resultados esperados

Não é expectável, nem desejável que eventos de segurança ocorram com frequência, e quando tiverem que ocorrer, espera-se que todos os mecanismos de segurança do sistema estejam em pleno funcionamento, e sejam capazes de detetar e impedir que o ataque seja bem sucedido. Estes mecanismos vão desde *firewalls* à Sistemas de Deteção de Intrusos (*Intrusion Detection System*, IDS), e como o sistema de monitorização não é uma ferramenta de primeira linha, é expectável que os resultados venham a ser observados ao longo do tempo e não logo após a passagem para ambiente de produção.

Nesta linha de pensamento, e após resultados satisfatórios em ambiente de testes, são esperados bons resultados no decorrer do tempo, pelo que melhor avaliação será feita com o passar do tempo.

## 5.3 Discussão sobre a segurança do Nagios

Sendo o Nagios uma excelente ferramenta de monitorização de código aberto também suscetível a vulnerabilidades, importa realçar alguns aspetos a ter em conta para a segurança do sistema.

O Nagios requer especial atenção, e por isso é aconselhável olhá-lo como uma potencial *backdoor* no sistema [13]. Isto porque o servidor Nagios tem acesso ou permissões em *firewalls* para poder monitorizar ativos de rede de forma remota, podendo assim consultá-los para obter informações sensíveis, úteis para a monitorização. O que significa que os servidores Nagios recebem uma certa confiança dos administradores, de modo a poderem obter as necessárias informações de monitorização.

É justamente essa confiança que se dá aos servidores de monitorização que podem representar um potencial risco de segurança, quando certos cuidados não são tidos em conta. A Figura 5.4 representa a exploração do Nagios como uma *backdoor*.

Se um atacante tiver a capacidade de entrar no sistema de monitorização, este pode forjar resultados e disparar alarmes para distrair a sua atividade, esconder alarmes, obter

informações valiosas para posteriores atividades ilícitas, tais como o melhor horário para poder passar despercebido, e tantas outras informações de grande sensibilidade que podem ser encontradas neste servidor. Em caso de serem executados pelo Nagios módulos capazes de reiniciar sistemas ou serviços, o atacante pode valer-se desses privilégios para desenvolver suas ações maliciosas.

É ainda bastante importante que se mantenha cifrada a comunicação do servidor para os equipamentos monitorizados, assim como a comunicação destes com o servidor, afim de evitar que atacantes possam escutar o tráfego e usarem essa informação a seu favor. Supondo que o atacante consiga escutar as respostas dos módulos *check_cpu*, *check_memory*, check_users, este pode analisar esta informação e ser capaz de perceber qual o melhor momento para comprometer um sistema e usar esses recursos sem ser notado.

Com base nos perigos mencionados, passamos a descrever adiante alguns aspetos a ter em conta, e que podem deixar o nosso servidor de monitorização e o sistema em geral mais seguro quando a solução de monitorização usada for o Nagios [13].

### Melhores práticas de segurança

**Usar um servidor de monitorização dedicado** É vivamente recomendado que se use um servidor de monitorização dedicado, e que este seja considerado e protegido como um dos principais servidores do sistema, merecendo igual atenção aos demais servidores, principalmente a nível de segurança [13].

É igualmente aconselhável que se mantenha os serviços em execução no mínimo e bloqueie o acesso a ele via TCP *wrappers*, *firewalls*, etc. Tendo em atenção que o servidor Nagios tem permissão para comunicar com todos os servidores monitorizados, sendo assim permitido pelas *firewalls* que circule tráfego do servidor Nagios para os demais ativos de rede e vice-versa, o que pode ser um risco de segurança.

**Não executar o Nagios como *root*** A execução do Nagios como *root*, além de não ser necessária, representa um grande risco de segurança. Por isso é vivamente recomendado que não se opte por esta via. É ainda possível configurar o Nagios para desativar privilégios após a inicialização, e se for mesmo incontornável a necessidade de privilégios *root*, neste caso aconselha-se a optar pelo *sudo* [13].

**Permissões de leitura e escrita na diretoria dos resultados das verificações** A diretoria onde são armazenados os resultados das verificações do Nagios, deve merecer especial atenção. É fundamental que o acesso a ela e todo seu conteúdo seja reservado, certificando-se que apenas o Nagios tem permissões para ler e escrever [13].

Se outros utilizadores diferentes do Nagios e Root tiverem permissões para esta diretoria, o risco desta ser comprometida passa a ser elevado, e acontecendo pode possibilitar a alteração dos resultados a favor do utilizador malicioso, gerando falsos positivos e/ou falsos negativos, que nitidamente seriam prejudiciais para a segurança da rede.

**Bloquear o ficheiro de comando externo** Ao ativar comandos externos, devemos certificar-nos que são definidas permissões apropriadas na diretoria /usr/local/nagios/-

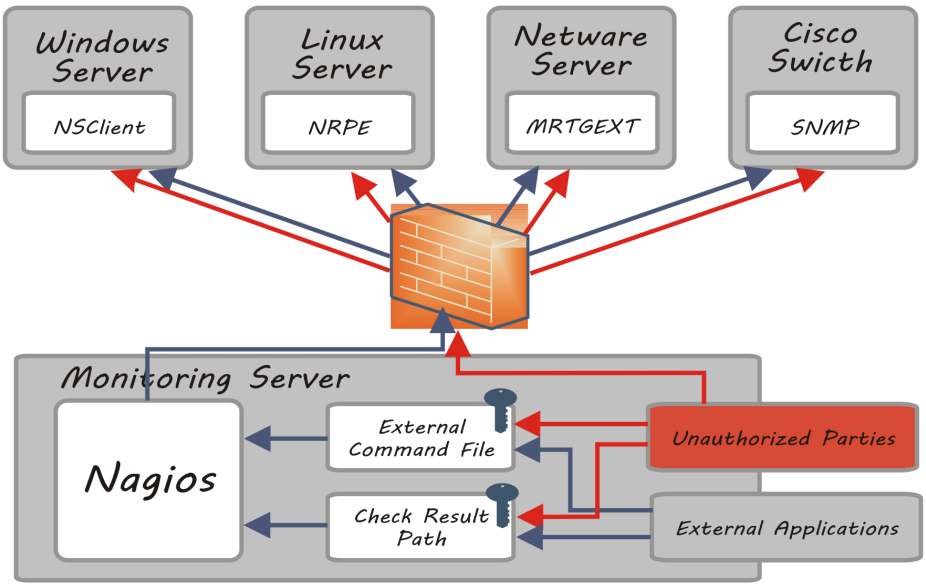


Fig. 5.4: Intrusão por meio da comunicação entre servidor e agente Nagios [13]

ver/rw [13].

Apenas é desejável que o utilizador Nagios, httpd, apache2 ou www-data tenham permissões para escrever no ficheiro de comando. No entanto, essas permissões são aceitáveis se o Nagios é executado numa máquina dedicada, não sendo aconselhável caso o Nagios esteja instalado numa máquina multi-utilizador, permitir que o utilizador do servidor web tenha permissões de escrita no ficheiro de comando, o que traduzir-se-ia num problema de segurança.

**Autenticação nas *Common Gateway Interface* CGI** É recomendado que o acesso às CGIs seja permitido mediante autenticação. Sabendo as permissões padrão após autenticação, devem ser autenticados contactos específicos para direitos adicionais conforme necessário [13].

**Implementar medidas de segurança, CGI avançadas** Considerar a implementação de medidas de segurança aprimoradas para as CGIs, pode ajudar a garantir que as credencias do utilizador usadas para aceder à interface web não são intercetadas por terceiros [13].

**Usar caminhos completos nas definições de comandos** Ao definir comandos, é importante certificar-se de especificar o caminho completo para quaisquer *script* ou binários [13].

**Ocultar informações sensíveis com $USERn$ Macros** No ato de configuração do Nagios, é possível definir a ocultação de informação sensível, como nomes de utilizadores, palavras passe, etc. Ao ser necessário especificar nomes de utilizadores e pa-

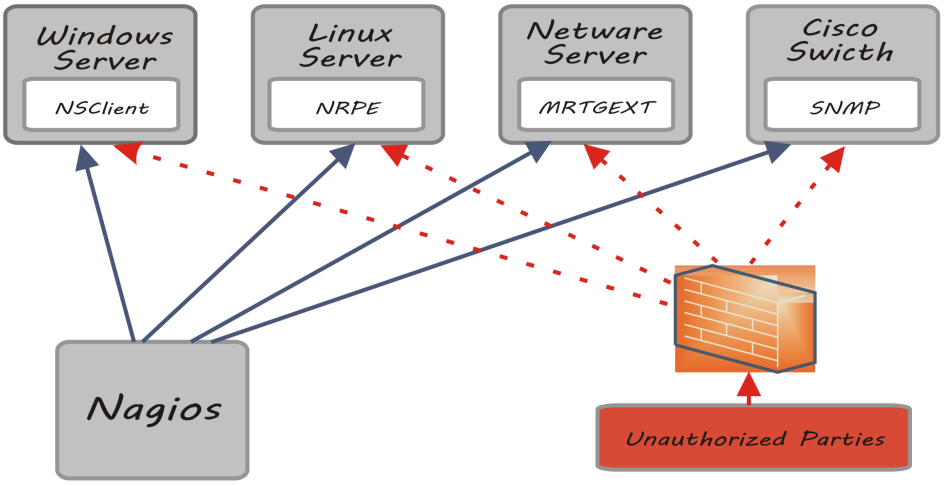


Fig. 5.5: Comunicação segura entre servidor e agentes Nagios [13]

lavras passe, é recomendado que se faça recorrendo ao uso de uma macro $USERn$ para ocultá-la. $USERn$ macros são definidos em um ou mais ficheiros de recurso [13].

**Evitar caracteres perigosos** Usando a diretiva illegal_macro_output_chars pode-se remover caracteres perigosos das macros $HOSTOUTPUT$, $SERVICEOUTPUT$, $HOSTPERFDATA$ e $SERVICEPERFDATA$ antes que eles sejam usados em notificações [13].

Caracteres perigosos são aquelas que possam ser interpretadas pela shell, abrindo assim um furo de segurança. Um exemplo disso é a presença de plicas (‘) nas macros $HOSTOUTPUT$, $SERVICEOUTPUT$, $HOSTPERFDATA$ e/ou $SERVICEPERFDATA$, o que poderia permitir que um atacante executasse um comando arbitrário como utilizador Nagios (sendo esta uma óptima razão para não executar o Nagios como root).

**Acesso seguro à agentes remotos** É importante que o acesso aos agentes remotos do Nagios (NRPE, NSClient, NSClient++, SNMP, etc.), sejam bem controlados (Figura 5.5). Tipicamente seria bastante constrangedor se cada um que quisesse pudesse comunicar com um agente remoto, dar instruções e receber informações das verificações [13].

**Canais de comunicação seguros** Tal como referido no inicio desta secção, a troca de informações de monitorização não cifradas representa uma vulnerabilidade. É fundamental certificar-se que os canais de comunicação entre o servidor Nagios e seus agentes remotos estão cifrados sempre que possível. Este é um importante passo na direção da segurança do sistema [13]. A Figura 5.6 representa a comunicação cifrada entre os servidores e os agentes Nagios.

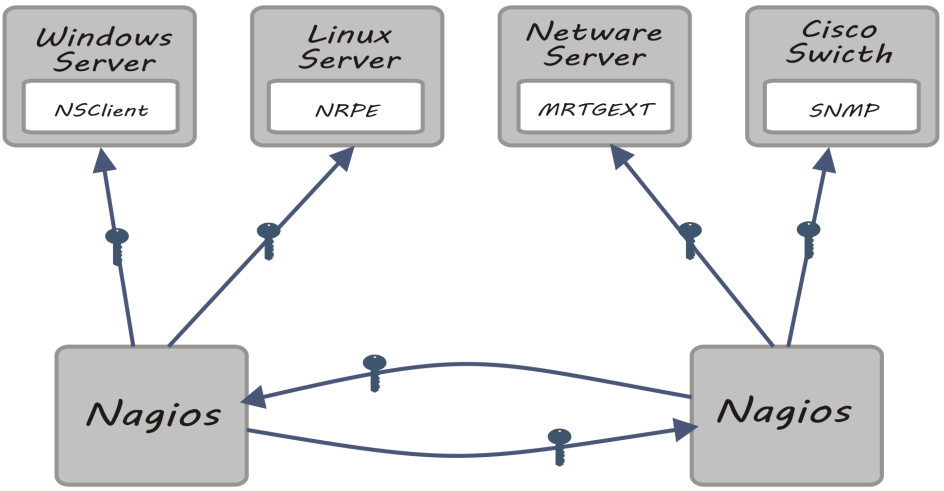


Fig. 5.6: Comunicação cifrada entre servidores e agentes Nagios [13]

# Capítulo 6

# Conclusão

Num sistema, as ferramentas de monitorização supervisionam as operações do potencialmente grande número de equipamentos conectados entre si e a Internet. Uma supervisão eficiente e eficaz é fundamental para garantir um serviço confiável e de alta qualidade. Nesta dissertação estudamos o Nagios como uma ferramenta de monitorização de código aberto com potencial para melhorar a segurança dos sistemas informáticos através do desenvolvimento de módulos focados na geração de indicadores de segurança.

Este trabalho consistiu na identificação das necessidades do recém reestruturado sistema de monitorização da Direção de Serviços Informáticos da Faculdade de Ciências, e no desenvolvimento de módulos com caraterísticas de segurança. Os módulos desenvolvidos foram exaustivamente testados, produzindo excelentes resultados que indicam ser possível incrementar a segurança das infraestruturas tecnológicas recorrendo à utilização das ferramentas de monitorização.

A inclusão destes módulos na plataforma de monitorização de Ciências permitiu um acompanhamento exaustivo, automatizado, centralizado e em tempo real de eventos de segurança, contribuindo para se ter um ambiente que combina eventos relacionados com a administração de sistema, e eventos de segurança, o que evita que uma ocorrência possa ter uma interpretação pela equipa responsável pela administração e outra interpretação diferente pela equipa responsável pela segurança do sistema.

Esta combinação de eventos em um único ambiente contribuiu para uma colaboração mais eficaz entre estas duas equipas distintas, ambas fundamentais para o correto funcionamento dos serviços. Espera-se um incremento da pro-atividade através da reação atempada a problemas, e uma melhoria do tempo de resposta a incidentes através da pronta deteção e alerta de eventos.

A título de exemplo, o módulo *check_dnsbl.py* relacionado a deteção de servidores de correio eletrónico em listas negras, reduz o tempo de resposta a incidentes e melhora a pro-atividade das equipas de administração. Esta melhoria verifica-se caso o servidor de correio eletrónico passe a fazer parte de uma rede responsável pelo envio de SPAM, uma vez que o módulo deteta logo este evento e permite que os administradores possam agir antes de serem observados problemas com o envio de correio eletrónico.

Por outro lado, o módulo *check_apache_status.py* melhora o tempo de resposta a incidentes uma vez que prováveis problemas de segurança no servidor web são descobertos através da pesquisa de indícios de comportamentos anómalos no ficheiro de log, gerando alertas aos administradores.

As ferramentas de monitorização são dotadas de funcionalidades que devidamente exploradas podem auxiliar positivamente os administradores de redes e sistemas a melhorar a segurança de suas infraestruturas.

No entanto, são necessárias algumas precauções na configuração, para evitar que os servidores de monitorização se tornem numa porta de entrada para utilizadores maliciosos. Por exemplo, uma medida importante neste campo é o tratamento dos servidores de monitorização como parte dos servidores de maior importância no sistema.

Nesta dissertação focámo-nos em explorar uma capacidade escondida nas ferramentas de monitorização, respondendo a um conjunto de necessidades de segurança já identificadas num contexto particular. Para o futuro, pensa-se aprofundar o estudo, e desenvolver módulos com potencial de aplicação genérica mais abrangente a outras realidades.

# Glossário

**backdoor** é um código malicioso usado por atacantes, que permite o seu retorno a um computador comprometido, por meio da inclusão de serviços criados ou modificados para esse fim. Normalmente esse programa é instalado de forma a passar despercebido. 32, 52, 64

**cache poisoning** é o comprometimento da segurança ou da integridade dos dados em um Sistema de Nomes de Domínios (Domain Name System DNS). 39

**Daemon** é a designação dada a ferramentas que funcionam em segundo plano, executando tarefas sem que o utilizador dê por ela. 46

**data warehouse** é um depósito de dados digitais que serve para armazenar informações detalhadas de uma organização, criando e organizando relatórios através de históricos que são depois usados pela empresa para ajudar a tomar decisões importantes com base nos fatos apresentados. 16

**defacement** termo usado para designar a atividade de desfigurar sítios na Internet. 37

**defacers** é a designação atribuída a pessoa responsável pela desfiguração de uma página na internet, prática conhecida como defacement. 37, 46

**e-learning** terminologia inglesa para designar aprendizagem a distância, recorrendo a equipamentos eletrónicos. 12

**firewalls** é um dispositivo de segurança usado para dividir e controlar o acesso entre redes de computadores. 64, 65

**hosts** podem ser considerados de duas formas diferentes: 1 - Em um sistema de dois ou mais computadores, é o computador que desempenha as funções principais ou de controle. 2 - Na Internet, é qualquer computador que funciona como ponto de partida e/ou de chegada nas transferências de dados. 24

**Login** entrada ou acesso ao sistema. 48

**moodle** é um sistema de gestão de atividades educativas que visa a criação de comunidades online em ambientes virtuais para a aprendizagem. 12

**multicast** é a comunicação na qual um quadro é enviado para um grupo específico de dispositivos ou clientes. Os clientes da transmissão multicast devem ser membros de um grupo multicast lógico para receber as informações. 18

**Plugins** são programas geralmente desenvolvidos por terceiros que podem ser adicionados a uma aplicação, de modo a prover funcionalidades adicionais. 19

**root** é o termo comummente usado para designar um utilizador com poderes de administração. 32, 65, 67

**Rootkit** é um conjunto de ferramentas usadas para esconder processos, ficheiros ou dados do sistema operativo, atacar e infetar outros computadores, roubar informações e interagir em tempo real com o sistema comprometido. 32

**scan** é a designação de ações que consistem em efetuar buscas minuciosas em redes, com o objetivo de identificar computadores ativos e recolher dados sobre eles como, por exemplo, portos abertos, serviços disponibilizados e programas instalados. 49, 50

**script** é o termo usado para designar pequenas sequencias de códigos, capazes de desempenhar uma determinada tarefa. 32

**shell** é um interpretador de linguagens de comandos que permite executar comandos interativamente ou dum ficheiro. Exemplos de shells em UNIX são a Bourne shell (sh), C shell (csh) ou Bourne-again shell (bash). 20

**software in-house** é um software que é produzido por uma entidade corporativa para fins de usá-lo dentro da organização. 24

**sudo** é um comando Linux que permite a um utilizador executar comandos com privilégios de administrador(root). 65

**switches** são equipamentos que funcionam normalmente na camada 2 do modelo OSI e tem como principal função interligar equipamentos de uma rede. 12

**web service** é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Esta tecnologia permite que novas aplicações possam interagir com aplicações já existentes e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. 54, 58, 59

**wrappers** são objetos que contêm funcionalidades que permitem manipular as variáveis de tipo primitivo. 65

# Bibliografia


[1] CERT-UE Advisory Advisory. Wannacry ransomware campaign exploiting smb vulnerability. Date: May 16 2017.

[2] Abhishek Amralkar, Mayank Gaikwad, Rohit Nerkar, Pulkit Gupta, and Mukesh Waghadhare. Monitoring tools. *TechWatch Report*, pages 88–93, 2016.

[3] Suranjith Ariyapperuma and Chris J Mitchell. Security vulnerabilities in dns and dnssec. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pages 335–342. IEEE, 2007.

[4] BuiltWith Pty Ltd [AU]. CMS Usage Statistics - statistics for websites using cms technologies. `https://trends.builtwith.com/cms`. Accessed: 2017-03-14.

[5] Michael Badger. *Zenoss Core 3. x Network and System Monitoring*. Packt Publishing Ltd, 2011.

[6] Marco Balduzzi, Manuel Egele, Engin Kirda, Davide Balzarotti, and Christopher Kruegel. A solution for the automated detection of clickjacking attacks. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 135–144. ACM, 2010.

[7] Wolfgang Barth. *Nagios: System and Network Monitoring*. No Starch Press, San Francisco, CA, USA, 2nd edition, 2008.

[8] Nagios Brazilian Community. Nagios core. `http://nagios-br.com/nagios-core`. Accessed: 2016-11-28.

[9] Douglas E Comer. Redes de computadores e internet-6ª edição. pages 28,29,447, 2016.

[10] Miguel Pupo Correia and Paulo Jorge Sousa. Segurança no software. *Lisboa: FCA*, 2010.

[11] Christos Douligeris and Aikaterini Mitrokotsa. Ddos attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, 44(5):643–666, 2004.

[12] Nagios Enterprises LLC. Nagios xi pricing. Revision: 080416.

[13] Ethan Galstad. Nagios core version 3. x documentation. *Nagios Community*, 2009.

[14] Bhadreshsinh G Gohil, Rishi K Pathak, and Axaykumar A Patel. Federated network security administration framework. 2013.

[15] Josune Hernantes, Gorka Gallardo, and Nicolas Serrano. It infrastructure-monitoring tools. *IEEE Software*, 32(4):88–93, 2015.

[16] Yih Huang, Arun Sood, and Ravi K Bhaskar. Countering web defacing attacks with system self-cleansing. In *Proceedings of 7th Word Multiconference on Systemics, Cybernetics and Informatics*, pages 12–16, 2003.

[17] Mohammed J Kabir. *Apache Server Bible*. IDG Books Worldwide, Inc., 1998.

[18] R Kenig, D Manor, Z Gadot, and D Trauner. Ddos survival handbook, 2013.

[19] Sebastian Lekies, Mario Heiderich, Dennis Appelt, Thorsten Holz, and Martin Johns. On the fragility and limitations of current browser-provided clickjacking protection schemes. *WOOT*, 12, 2012.

[20] John Levine. Dns blacklists and whitelists. Technical report, 2010.

[21] Robert Love. Kernel korner: Intro to inotify. *Linux Journal*, 2005(139):8, 2005.

[22] Steve Lynch. Attacks over dns. `http://resources.infosecinstitute.com/attacks-over-dns/`. Accessed: 2017-06-15.

[23] Miroslav Matýsek, Milan Adámek, M Kubalcík, and Miroslav Mihok. Monitoring of computer networks and applications using nagios. *Advances in Data Networks, Communications, Computers and Materials Monitoring*, pages 63–67, 2012.

[24] Sophon Mongkolluksamee, Panita Pongpaibool, and Chavee Issariyapat. Strengths and limitations of nagios as a network monitoring solution. In *Proceedings of the 7th International Joint Conference on Computer Science and Software Engineering (JCSSE 2010)*, volume 7, 2009.

[25] Sérgio Manuel Maia Torres Moreira. *Monitorização de Redes e Sistemas Informáticos*. PhD thesis, Faculdade de Ciências da Universidade do Porto, 2014.

[26] Fernando Pereira and Rui Gerreiro. *Linux Curso Completo*. FCA Editora, 7th edition, 2012.

[27] MA Pervilä. Using nagios to monitor faults in a self-healing environment. In *Seminar on Self-Healing Systems, University of Helsinki*, pages 1–6. Citeseer, 2007.

[28] Márcio Pessoa. Nagios guia de referência rápida. pages 1–2, 2010.

[29] Ravikant Sahu and Samta Gajhbhiye. Implementation and empirical analysis of it monitoring system using open source software based on nagios core. 2014.

[30] Max Schubert, Derrick Bennett, Jonathan Gines, Andrew Hay, and John Strand. *Nagios 3 enterprise network monitoring: including plug-ins and hardware devices*. Syngress, 2008.

[31] Sitio.pt. Estatísticas de internet portuguesa. https://www.siteo.pt/estatisticas-de-internet-portuguesa. Accessed: 2017-05-30.

[32] OWASP The Open Web Application Security Project. Owasp top 10 - 2017 rc1 - the ten most critical web application security risks. pages 5–17, 2017.

[33] Ciências ULisboa. Apresentação da direção de serviços informáticos. https://ciencias.ulisboa.pt/pt/unidade. Accessed: 2016-11-28.

[34] Ciências ULisboa. Apresentação da faculdade de ciências. https://ciencias.ulisboa.pt/pt/faculdade. Accessed: 2016-11-28.

[35] Ciências ULisboa. Estatísticas da faculdade de ciências. https://ciencias.ulisboa.pt/pt/estatisticas. Accessed: 2016-11-28.

[36] Verisign Inc [US]. DNSSEC - estatística dos domínios protegidos com dnssec. http://scoreboard.verisignlabs.com/. Accessed: 2017-03-19.

[37] Hsiu-Lien Yeh, Yan-Fu Chen, Tsung-Tai Yeh, Pei-Chi Huang, Shin-Hao Liu, Hsin-Wen Wei, and Tsan-sheng Hsu. A monitoring system based on nagios for data grid environments. In *International Conference on Grid Computing and Applications*, 2011.

[38] Lihua Yuan, Krishna Kant, Prasant Mohapatra, and Chen-Nee Chuah. Dox: A peer-to-peer antidote for dns cache poisoning attacks. In *Communications, 2006. ICC'06. IEEE International Conference on*, volume 5, pages 2345–2350. IEEE, 2006.

# Apêndice A

# Código Fonte dos Módulos

## A.1 Código fonte do módulo check_apache_status.py

```
#! /usr/bin/env python
'''
* Description:
* This file contains the check_apache_status plugin
* Script to check the server accesses, and warn if an abnormal number
* of occurrence of 4xx code family
*
* Creation date: 24/02/2017
* Date last updated: 18/04/2017
*
* License: GPL
* Copyright (c) 2017 DI-FCUL, Raimundo Chipongue
*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program.  If not, see <http://www.gnu.org/licenses/>.
*************************************************************************

'''

import os
import sys
from optparse import OptionParser
from time import strftime
import time
import datetime

__author__ = "\nAuthor: Raimundo Henrique da Silva Chipongue\nE-mail:
```

```
    fc48807@alunos.fc.ul.pt, chipongue1@gmail.com\nInstitution: Faculty
     of Science of the University of Lisbon\n"
__version__= "1.0.0"

# define exit codes
ExitOK = 0
ExitWarning = 1
ExitCritical = 2
ExitUnknown = 3

def checklogs(opts):
    critical = [i for i in opts.critical.split(",")]
    warning = [i for i in opts.warning.split(",")]
    try:
        critical_total = int(critical[0])
        critical_ip = int(critical[1])
    except:
        print('Review value of critical options, these must be a
    intiger separated by ":"')
        sys.exit(ExitUnknown)
    try:
        warning_total = int(warning[0])
        warning_ip = int(warning[1])
    except:
        print('Review value of warning options, these must be a intiger
     separated by ":"')
        sys.exit(ExitUnknown)
    if opts.number:
        totalerror = int(os.popen("tail -n %s %s | grep 404 | wc -l"%(
    opts.number,opts.path)).read())
        qtdd = str(os.popen("tail -n %s %s | grep 404 | awk '{print $1
    }' | sort | uniq -c | sort $1 |awk '{print $1}' | sort -nr | awk '{
    print $1}'"%(opts.number, opts.path)).read())
        ips = str(os.popen("tail -n %s %s | grep 404 | awk '{print $1}'
     | sort | uniq -c | sort -nr $1 | sort -nr | awk '{print $2}'"%(
    opts.number,opts.path)).read())
        os.popen("tail -n %s %s | grep 404 > %s"%(opts.number,opts.path
    ,opts.output)).read()

    else:
        totalerror = int(os.popen("cat %s | grep 404 | wc -l"%opts.path
    ).read())
        qtdd = str(os.popen("cat %s | grep 404 | awk '{print $1}' |
    sort | uniq -c | sort $1 |awk '{print $1}' | sort -nr | awk '{print
     $1}'"%opts.path).read())
        ips = str(os.popen("cat %s | grep 404 | awk '{print $1}' | sort
     | uniq -c | sort -nr $1 | sort -nr | awk '{print $2}'"%opts.path).
    read())
        os.popen("cat %s %s | grep 404 > %s"%(opts.number,opts.path,
    opts.output)).read()
    num_ip = []
    IPs = []

    if totalerror != 0:
        newitemip = [i for i in ips.split("\n")]
        IPs.extend([i for i in newitemip])
```

```
77          IPs.pop(-1)
78          newitemnum = [i for i in qtdd.split("\n")]
79          num_ip.extend([i for i in newitemnum])
80          num_ip.pop(-1)
81          num = 0
82          for numero in num_ip:
83              if int(numero) >= critical_ip or totalerror >=
    critical_total:
84                  print("Critical - Were found %s errors 404, the IP
    Address %s, did %s acess attempts, see %s"%(totalerror, IPs[num],
    numero,opts.output))
85                  sys.exit(ExitCritical)
86              elif int(numero) >= warning_ip or totalerror >=
    warning_total:
87                  print("Warning - Were found %s errors 404, the IP
    Address %s, did %s acess attempts, see %s"%(totalerror, IPs[num],
    numero,opts.output))
88                  sys.exit(ExitWarning)
89              elif int(numero) < warning_ip or totalerror < warning_total
    :
90                  print("Ok - Were found only %s errors 404, the IP
    Address %s, did %s acess attempts, see %s"%(totalerror, IPs[num],
    numero,opts.output))
91                  sys.exit(ExitOK)
92              else:
93                  num = num +1
94      else:
95          print('No 404 error were found in file log "%s"'%opts.path)
96          sys.exit(ExitOK)
97
98  def main():
99      parser = OptionParser("usage: %prog [options] ARG1 ARG2 FOR EXAMPLE
    : -c 300,100 -w 200,70 -p /var/log/apache2/access.log -n 2000")
100     parser.add_option("-p","--path", dest="path",
101                       help="Enter the full path to the apache logs file
    , i.e. -p /var/log/apache2/access.log")
102     parser.add_option("-o","--output", dest="output", default="/tmp/
    apache.txt",
103                       help="Enter the full path to save txt file
    contined all find line, i.e. -p /var/log/apache2/apache.txt"+
104                       "default is /tmp, but this folder is not secure
    to save this file")
105     parser.add_option("-c","--critical", dest="critical", default=False
    , type=str,
106                       help="Enter the number of all attemps you
    considered critical and number of attemps of a single IP are
    considered critical, separated by ':'")
107     parser.add_option("-w","--warning", dest="warning", default=False,
    type=str,
108                       help="Enter the number of all attemps you
    considered warning and number of attemps of a single IP are
    considered warning, separated by ':'")
109     parser.add_option("-n","--number", dest="number", default=False,
    type=int,
110                       help="Enter the number of records that you want
    to check")
```

```
111     parser.add_option("-V","--version", action="store_true", dest="
        version", help="This option show the current version number of the
        program and exit")
112     parser.add_option("-A","--author", action="store_true", dest="
        author", help="This option show author information and exit")
113     (opts, args) = parser.parse_args()
114
115     if opts.author:
116         print(__author__)
117         sys.exit()
118     if opts.version:
119         print("check_apache404.py %s"%__version__)
120         sys.exit()
121     if not opts.critical:
122         parser.error('Please, this program requires critical values
        arguments, these must be a intiger separated by ",")
123     elif not opts.warning:
124         parser.error('Please, this program requires warning values
        arguments, these must be a intiger separated by ",")
125     if opts.path:
126         if not os.path.exists(opts.path):
127             parser.error("Please, this program requires to specify a
        valid path file.")
128         else:
129             checklogs(opts)
130     else:
131         parser.error("Please, this program requires to specify a valid
        path file.")
132
133 if __name__ == '__main__':
134     main()
```

## A.2 Código fonte do módulo check_app.py

```
#! /usr/bin/env python
'''
* Description:
* Nagios check_app plugin
* Script to check for the existence of new applications installed
*
* Creation date: 17/01/2017
* Date last updated: 19/03/2017
*
* License: GPL
* Copyright (c) 2017 DI-FCUL, Raimundo Chipongue
*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program.  If not, see <http://www.gnu.org/licenses/>.
*************************************************************************
'''

import os
import sys
import difflib
from optparse import OptionParser

__author__ = "\nAuthor: Raimundo Henrique da Silva Chipongue\nE-mail:
    fc48807@alunos.fc.ul.pt, chipongue1@gmail.com\nInstitution: Faculty
     of Science of the University of Lisbon\n"
__version__= "1.0.0"

# define exit codes
ExitOK = 0
ExitWarning = 1
ExitCritical = 2
ExitUnknown = 3

def checkapp(opts):
    path = opts.storefolder
    if opts.path:
        folder_paths = opts.path
        folder_paths = folder_paths.replace(","," ")
    else:
        folder_paths = "/usr/bin/ /sbin/ /bin/ /usr/sbin/ /usr/local/
    bin/"

```

```
50      path_exist = ("%sapp1.txt" %path)
51      if not os.path.exists(path_exist):
52          if os.path.exists(path):
53              os.popen("find %s | sort > %sapp1.txt" %(folder_paths, path
    )).read()
54          else:
55              os.makedirs(path)
56              os.popen("find %s | sort > %sapp1.txt"%(folder_paths, path)
    ).read()
57
58      diff_num = int(os.popen("find %s | sort | diff - %sapp1.txt | wc -l
    "%(folder_paths, path)).read())
59      os.popen("find %s | sort | diff - %sapp1.txt > %sappdiff.txt"%(
    folder_paths, path, path)).read()
60      if diff_num:
61          diff_num = diff_num -1
62          print("%s recently installed applications, see %sappdiff.txt"
    %(diff_num,path))
63          sys.exit(ExitCritical)
64      else:
65          print("Not found any recently installed applications")
66          sys.exit(ExitOK)
67
68  def main():
69      parser = OptionParser("usage: %prog This plugin doesn't require any
     argument for this works.")
70      parser.add_option("-V","--version", action="store_true", dest="
    version", help="This option show the current version number of the
    program and exit")
71      parser.add_option("-A","--author", action="store_true", dest="
    author", help="This option show author information and exit")
72      parser.add_option("-p","--path", dest="path", default=False,
73                        help="Specify all bin folder you need to
    monitoring, Ex. check_app.py -p /usr/bin/,/sbin/,/bin/,/usr/sbin/,/
    usr/local/bin/, if this argument is not used, the program uses the
    default folders. Whenever you change the folders to be monitored,
    restart the file /tmp/appdiff.txt")
74      parser.add_option("-s","--storefolder", dest="storefolder", default
    ="/tmp/",
75                        help="Specify the full path of the folder where
    you save the file with information of installed applications." +
76                        "By default this is /tmp/ folder")
77      (opts, args) = parser.parse_args()
78      if not os.path.exists(opts.storefolder):
79          parser.error("Please, this program requires to specifay a valid
     and private folder to store temp file")
80      if opts.author:
81          print(__author__)
82          sys.exit()
83      if opts.version:
84          print("check_app.py %s"%__version__)
85          sys.exit()
86      checkapp(opts)
87
88  if __name__ == '__main__':
89      main()
```

## A.3 Código fonte do módulo check_synflood.py

```
#! /usr/bin/env python
'''
* Description:
* This file contains the check_synflood plugin
* Script to check the server accesses, and warn if an abnormal number
* of requests, indicating probable attempt to DDOS
*
* Creation date: 14/01/2017
* Date last updated: 19/03/2017
*
* License: GPL
* Copyright (c) 2017 DI-FCUL, Raimundo Chipongue
*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program.  If not, see <http://www.gnu.org/licenses/>.
*************************************************************************
'''

import os
import sys
from optparse import OptionParser

__author__ = "\nAuthor: Raimundo Henrique da Silva Chipongue\nE-mail:
    fc48807@alunos.fc.ul.pt, chipongue1@gmail.com\nInstitution: Faculty
     of Science of the University of Lisbon\n"
__version__= "1.0.0"

# define exit codes
ExitOK = 0
ExitWarning = 1
ExitCritical = 2
ExitUnknown = 3

connection=int(os.popen("netstat -antu | grep SYN_RECV | awk '{print $5
    }' | cut -d: -f1 | sort | uniq -c | sort -rn | grep -v 127.0.0.1 |
    wc -l").read())
def check(opts):
    critical = opts.crit
    warning = opts.warn
    if critical and warning:
        if connection > critical:
            print('CRITICAL - The host has %s active connections' %
```

```
    connection)
49              sys.exit(ExitCritical)
50          elif connection > warning:
51              print('WARNING - The host has %s active connections' %
    connection)
52              sys.exit(ExitWarning)
53          else:
54              print("OK - The host has %s active connections" %connection
    )
55              sys.exit(ExitOK)
56      else:
57          print("UNKNOWN - The number of active connections is unknown")
58          sys.exit(ExitUnknown)
59  def main():
60      parser = OptionParser("usage: %prog [options] ARG1 ARG2 FOR EXAMPLE
    : -c 300 -w 200")
61      parser.add_option("-c","--critical", type="int", dest="crit", help=
    "the value if consider very heigth connection in web server")
62      parser.add_option("-w","--warning", type= "int", dest="warn", help=
    "the value if consider heigth connection in web server")
63      parser.add_option("-V","--version", action="store_true", dest="
    version", help="This option show the current version number of the
    program and exit")
64      parser.add_option("-A","--author", action="store_true", dest="
    author", help="This option show author information and exit")
65      (opts, args) = parser.parse_args()
66
67      if opts.author:
68          print(__author__)
69          sys.exit()
70      if opts.version:
71          print("check_ddos.py %s"%__version__)
72          sys.exit()
73
74      if opts.crit and opts.warn:
75          if opts.crit < opts.warn:
76              print("Critical value < Warning value, please check you
    config")
77              sys.exit(ExitCritical)
78      else:
79          parser.error("Please, this program requires -c and -w arguments
    , for example: -c 300 -w 200")
80          sys.exit(ExitCritical)
81
82      check(opts)
83
84  if __name__ == '__main__':
85      main()
```

## A.4 Código fonte do módulo check_defacement.py

```
#! /usr/bin/env python3
'''
* Description:
* This file contains the check_defacement plugin
* Script to check access to the server, and if the existence of
* specific words in the html content of a page that indicate
* probable defacement
*
* Creation date: 24/01/2017
* Date last updated: 19/03/2017
*
* License: GPL
* Copyright (c) 2017 DI-FCUL, Raimundo Chipongue
*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program.  If not, see <http://www.gnu.org/licenses/>.
*************************************************************************

'''

import os
import sys
from optparse import OptionParser
import urllib.request
import re
import urllib

__author__ = "\nAuthor: Raimundo Henrique da Silva Chipongue\nE-mail:
    fc48807@alunos.fc.ul.pt, chipongue1@gmail.com\nInstitution: Faculty
     of Science of the University of Lisbon\n"
__version__= "1.0.0"

# define exit codes
ExitOK = 0
ExitWarning = 1
ExitCritical = 2
ExitUnknown = 3

def verify(opts):
    Ignore = []
    if opts.Ignore:
        Ig = [i for i in opts.Ignore.split(",")]
        Ignore.extend([i for i in Ig])
```

```
51          Ignore = sorted(list(set(Ignore)))
52      try:
53          req = urllib.request.Request(opts.url)
54          resp = urllib.request.urlopen(req,timeout=2)
55      except:
56          print("Internet error or page not found, error 404, url: %s"%
    opts.url)
57          sys.exit(ExitUnknown)
58
59      path_tmp = opts.path
60      if opts.ignore:
61          keywords = []
62      else:
63          keywords = ['Anonymous', 'Hacked', 'Hacker', 'hack', 'h4ck3d',
64                      'attack','Hacktivist','virus','defaced',
65                      'Gary McKinnon','LulzSec','Lulz Security','lulz',
66                      'Adrian Lamo','Astra','MafiaBoy','YTCracker','414s'
    ,
67                      'FinnSec Security','Chaos Computer Club','Cicada
    3301',
68                      'Croatian Revolution Hackers','Cult of the Dead Cow
    ',
69                      'cDc','Decocidio','Digital DawgPound','Global kOS '
    ,
70                      'globalHell','Goatse Security','GoatSec','Hacking
    Team',
71                      'Hacking','Hackweiser','Honker Union', 'L0pht',
                'Mazafaka', 'milw0rm','NCPH','OurMine','Syrian
    Electronic Army',
72                      'TeaMp0isoN','UGNazi','bl@ck dr@gon','Lizard Squad'
    ,
73                      'Tarh Andishan','ArabAttack','k4zuk3','kdms team',
74                      'Dragonfly']
75
76      if opts.keyword:
77          newitem = [i for i in opts.keyword.split(",")]
78          keywords.extend([i for i in newitem])
79          keywords = sorted(list(set(keywords)))
80      else:
81          keywords = (sorted(keywords))
82      try:
83          os.popen("wget %s –O %sdefacement.html > /dev/null 2>&1"%(opts.
    url, path_tmp)).read()
84      except:
85          print("Unable to connect")
86          sys.exit(ExitUnknown)
87      if os.path.exists("%sdefacement.html"%path_tmp):
88          if not os.stat("%sdefacement.html"%path_tmp).st_size:
89              print("Can't read the url: %s"%opts.url)
90              sys.exit(ExitUnknown)
91      else:
92          print("Can't read the url: %s"%opts.url)
93          sys.exit(ExitUnknown)
94
95      keywords = sorted(list(set(keywords) - set(Ignore)))
96
```

```
97      path = ("%sdefacement.html"%path_tmp)
98      word_macth = []
99      num = 0
100     for i in range(0,len(keywords)):
101         text = str(keywords[num])
102         macth = int(os.popen("grep -i '%s' %s | wc -l"%(text, path)).
        read())
103         num = num +1
104         if macth != 0:
105             word_macth.extend([text])
106
107     os.popen("rm -fr %sdefacement.html"%path_tmp).read()
108
109     if word_macth != []:
110         print('Dangerous words were found, verify that the following %s
         words "%s", are legitimate in the following web page: %s'%(len(
        word_macth),(", ".join(word_macth)),opts.url))
111         sys.exit(ExitCritical)
112     else:
113         print("No dangerous words are found")
114         sys.exit(ExitOK)
115
116 def main():
117     parser = OptionParser("usage: %prog [options] ARG1 ARG2 FOR EXAMPLE
        : -U https://www.ciencias.ulisboa.pt -K hacker,hacked,anonymous")
118     parser.add_option("-U","--url", dest="url",
119                       help="Specify the full url you want to check, i.e
        . -U https://www.ciencias.ulisboa.pt")
120     parser.add_option("-K","--keyword", dest="keyword", default=False,
121                       help="Put the words you want to search, i.e. -K
        hacker,hacked,anonymous")
122     parser.add_option("-V","--version", action="store_true", dest="
        version", help="This option show the current version number of the
        program and exit")
123     parser.add_option("-A","--author", action="store_true", dest="
        author", help="This option show author information and exit")
124     parser.add_option("-i","--ignore", action="store_true", dest="
        ignore",
125                       help="Use this option to ignore all pre-installed
         suspect words")
126     parser.add_option("-I","--Ignore", dest="Ignore",default=False,
        type=str,
127                       help="Use this option to ignore one or multiple
        pre-installed suspect words")
128     parser.add_option("-p","--path", dest="path", default="/tmp/",
129                       help="Specify the full path of the folder where
        you store temp file." +
130                       "By default this is /tmp/ folder")
131     (opts, args) = parser.parse_args()
132
133     if opts.author:
134         print(__author__)
135         sys.exit()
136     if opts.version:
137         print("check_defacement.py %s"%__version__)
138         sys.exit()
```

```
139     if opts.ignore:
140         if not opts.keyword:
141             parser.error("When using -i option, you need to specify at
    least one suspect word.")
142     if opts.url:
143         verify(opts)
144     else:
145         parser.error("Please, this program requires url arguments.")
146
147 if __name__ == '__main__':
148     main()
```

## A.5 Código fonte do módulo check_dns.py

```
#! /usr/bin/env python3
'''
* Description:
* This file contains the check_dns plugin
* Script to check DNS changes
*
* Creation date: 10/02/2017
* Date last updated: 19/03/2017
*
* Nagios check_dns plugin
*
* License: GPL
* Copyright (c) 2017 DI-FCUL, Raimundo Chipongue
*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program.  If not, see <http://www.gnu.org/licenses/>.
*************************************************************************

'''

import os
import sys
import urllib.request
from optparse import OptionParser
import ipaddress
import socket

__author__ = "\nAuthor: Raimundo Henrique da Silva Chipongue\nE-mail:
    fc48807@alunos.fc.ul.pt, chipongue1@gmail.com\nInstitution: Faculty
     of Science of the University of Lisbon\n"
__version__= "1.0.0"

# define exit codes
ExitOK = 0
ExitWarning = 1
ExitCritical = 2
ExitUnknown = 3

def check_connectivity():
    try:
        urllib.request.urlopen('http://194.210.238.163', timeout=2)
        return True
    except urllib.request.URLError:
```

```
51          return False
52
53  def dns_f(opts):
54      if check_connectivity():
55
56          address_found =[]
57          iplist = []
58          domain = opts.hostname
59          domain = domain.replace("https://", "")
60          domain = domain.replace("http://", "")
61          domain = domain.replace("www.", "")
62          try:
63              answers = os.popen("dig +short %s @%s"%(domain, opts.
    dnsserver)).read()
64          except:
65              print("Connot query dns")
66              sys.exit(ExitUnknown)
67
68          if answers != "" or answers != 0:
69              address_found.extend([i for i in answers.split("\n")])
70              error = 0
71              n = -1
72              for ip in address_found:
73                  n = n +1
74                  try:
75                      ipaddress.ip_address(address_found[n])
76                      iplist.extend([i for i in str(address_found[n]).
    split(",")])
77                  except:
78                      error = error + 1
79
80          hostaddress = []
81          hostaddress.extend([opts.hostaddress])
82          try:
83              match = list(set(hostaddress).intersection(iplist))
84              match_item = str(match[0])
85          except:
86              match_item = 0
87          if opts.hostaddress == match_item:
88              print("The IP Address %s corresponds to the domain name %s"
     %(match_item, domain))
89              sys.exit(ExitOK)
90          else:
91              print("The IP Address %s does not match the domain name %s"
    %(opts.hostaddress, domain))
92              sys.exit(ExitCritical)
93      else:
94          print("Error, check you internet connection")
95          sys.exit(ExitUnknown)
96
97  def main():
98      parser = OptionParser("usage: %prog [options] ARG1 ARG2 FOR EXAMPLE
    : -I 192.168.0.1 -H domain.com")
99      parser.add_option("-H","--hostname", dest="hostname", help="Specify
     the Domain Name you want to check")
100     parser.add_option("-I","--hostaddres", dest="hostaddress", help="
```

```
    Specify the IP you want to check")
101     parser.add_option("-d","--dnsserver", type=str, default="8.8.8.8",
    dest="dnsserver",
102                       help="Specify the DNS server you need to use for
    check DNSSEC, for example: -d 127.0.0.1, default value is 8.8.8.8")
103     parser.add_option("-V","--version", action="store_true", dest="
    version", help="This option show the current version number of the
    program and exit")
104     parser.add_option("-A","--author", action="store_true", dest="
    author", help="This option show author information and exit")
105     (opts, args) = parser.parse_args()
106
107     if opts.author:
108         print(__author__)
109         sys.exit()
110     if opts.version:
111         print("check_dns.py %s"%__version__)
112         sys.exit()
113     if opts.hostname and opts.hostaddress:
114         try:
115             ip = ipaddress.ip_address(opts.hostaddress)
116         except:
117             parser.error("This application requires a valid IP Address.
    ")
118     else:
119         parser.error("Usage: %s -H <hostname> -I <IP Address>"%sys.argv
    [0])
120     dns_f(opts)
121
122 if __name__ == '__main__':
123     main()
```

## A.6 Código fonte do módulo check_dnsbl.py

```
#! /usr/bin/env python3
'''
* Description:
* This file contains the check_dnsbl plugin
* Script to check if the IP Address has a blacklist
*
* Creation date: 24/01/2017
* Date last updated: 19/03/2017
*
* License: GPL
* Copyright (c) 2017 DI-FCUL, Raimundo Chipongue
*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program.  If not, see <http://www.gnu.org/licenses/>.
*****************************************************************************
'''

import os
import sys
import dns.resolver
import urllib.request
from dns import resolver
from optparse import OptionParser
from collections import OrderedDict
from itertools import repeat
import ipaddress

__author__ = "\nAuthor: Raimundo Henrique da Silva Chipongue\nE-mail:
    fc48807@alunos.fc.ul.pt, chipongue1@gmail.com\nInstitution: Faculty
     of Science of the University of Lisbon\n"
__version__= "1.0.0"

# define exit codes
ExitOK = 0
ExitWarning = 1
ExitCritical = 2
ExitUnknown = 3

def check_connectivity():
    try:
        urllib.request.urlopen('http://194.210.238.163', timeout=2)
        return True
```

```
51
52      except urllib.request.URLError:
53          return False
54
55  def open_bl(opts):
56      #newitem = str(opts.blist)
57      Ignore = []
58      if opts.Ignore:
59              Ig = [i for i in opts.Ignore.split(",")]
60              Ignore.extend([i for i in Ig])
61              Ignore = sorted(list(set(Ignore)))
62
63      if opts.ignore:
64          blacklist = []
65
66      else:
67          blacklist = ["zen.spamhaus.org", "spam.abuse.ch", "cbl.abuseat.
    org","virbl.dnsbl.bit.nl",
68                  "dnsbl.inps.de", "ix.dnsbl.manitu.net", "dnsbl.sorbs.net
    ", "bl.spamcannibal.org",
69                  "bl.spamcop.net", "xbl.spamhaus.org", "pbl.spamhaus.org"
    , "dnsbl-1.uceprotect.net",
70                  "dnsbl-2.uceprotect.net", "dnsbl-3.uceprotect.net", "db.
    wpbl.info", "safe.dnsbl.sorbs.net",
71                   "b.barracudacentral.org", "access.redhawk.org", "dnsbl.
    justspam.org","dnsbl.sorbs.net",
72                    "noservers.dnsbl.sorbs.net","rhsbl.sorbs.net","sbl.
    spamhaus.org","xbl.spamhaus.org",
73                    "pbl.spamhaus.org","dnsbl.cobion.com","dyna.spamrats.
    com"]
74
75      if opts.blist:
76          newitem = [i for i in opts.blist.split(",")]
77          blacklist.extend([i for i in newitem])
78          blacklist = sorted(list(set(blacklist) - set(Ignore)))
79          return sorted(list(set(blacklist)))
80      else:
81          blacklist = sorted(list(set(blacklist) - set(Ignore)))
82          return sorted(blacklist)
83
84  def dns_f(opts):
85      if check_connectivity():
86          myIP = opts.host
87          if not myIP:
88              myIP = (os.popen("dig +short myip.opendns.com @resolver1.
    opendns.com").read())
89              myIP = myIP.replace("\n", "")
90
91          blacklist = open_bl(opts)
92          blacklisted = []
93          noblacklisted = []
94          timeout = []
95          nonameserver = []
96          noanswer = []
97          number = 0
98          for bl in blacklist:
```

```
99              try:
100                 my_resolver = dns.resolver.Resolver()
101                 query = '.'.join(reversed(str(myIP).split("."))) + "."
    + bl
102                 my_resolver.timeout = 1
103                 my_resolver.lifetime = 1
104                 answers = my_resolver.query(query, "A")
105                 if answers:
106                     answer_txt = my_resolver.query(query, "TXT")
107                     blacklisted.append(str(bl))
108                     number = number + 1
109
110             except dns.resolver.NXDOMAIN:
111                 noblacklisted.append(str(bl))
112
113             except dns.resolver.Timeout:
114                 timeout.append(str(bl))
115
116             except dns.resolver.NoNameservers:
117                 nonameserver.append(str(bl))
118
119             except dns.resolver.NoAnswer:
120                 noanswer.append(str(bl))
121
122         if blacklisted:
123             print("IP %s LISTED in %s blacklist: %s" %(myIP, number, ',
     '.join(blacklisted)))
124             sys.exit(ExitCritical)
125
126         else:
127             print("IP %s NOT LISTED in dns blacklist"%myIP)
128             sys.exit(ExitOK)
129     else:
130         print("Error, check you internet connection")
131         sys.exit(ExitUnknown)
132
133 def main():
134     parser = OptionParser("usage: %prog -H <IP address> and -l <anylist
    .com,anylist.org>, black list you have to check")
135     parser.add_option("-H","--hostaddress", dest="host", help="Specify
    the IP address you want to check")
136     parser.add_option("-l","--list", dest="blist", default=False, type=
    "string", help="If you heve an list to add, please enter -l <
    anylist.com,anylist.org>")
137     parser.add_option("-V","--version", action="store_true", dest="
    version", help="This option show the current version number of the
    program and exit")
138     parser.add_option("-A","--author", action="store_true", dest="
    author", help="This option show author information and exit")
139     parser.add_option("-i","--ignore", action="store_true", dest="
    ignore",
140                       help="Use this option to ignore all pre-installed
     blacklists")
141     parser.add_option("-I","--Ignore", dest="Ignore",default=False,
    type=str,
142                       help="Use this option to ignore one or multiple
```

```
    pre-installed blacklist")
143
144     (opts, args) = parser.parse_args()
145     if opts.author:
146         print(__author__)
147         sys.exit()
148     if opts.version:
149         print("check_dnsbl.py %s"%__version__)
150         sys.exit()
151     if opts.ignore:
152         if not opts.blist:
153             parser.error("When using -i option, you need to specify at
    least one blacklist.")
154     if opts.host:
155         try:
156             ip = ipaddress.ip_address(opts.host)
157         except ValueError:
158             parser.error("Incorrect IP Address.")
159         dns_f(opts)
160     else:
161         dns_f(opts)
162
163 if __name__ == '__main__':
164     main()
```

## A.7 Código fonte do módulo check_dnssec.py

```
#! /usr/bin/env python3

'''
* Description:
* This file contains the check_dnssec plugin
* Script to check without a particular domain has a DNSSEC signature.
* Creation date: 14/01/2017
* Date last updated: 19/03/2017
*
* License: GPL
* Copyright (c) 2017 DI-FCUL, Raimundo Chipongue
*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program.  If not, see <http://www.gnu.org/licenses/>.
*****************************************************************************
'''

import os
import sys
import requests
import urllib.request
from optparse import OptionParser
import socket

__author__ = "\nAuthor: Raimundo Henrique da Silva Chipongue\nE-mail:
    fc48807@alunos.fc.ul.pt, chipongue1@gmail.com\nInstitution: Faculty
     of Science of the University of Lisbon\n"
__version__= "1.0.0"

# define exit codes
ExitOK = 0
ExitWarning = 1
ExitCritical = 2
ExitUnknown = 3

def testdnssec(opts):
    domain = opts.domain
    domain = domain.replace("https://", "")
    domain = domain.replace("http://", "")
    domain = domain.replace("www.", "")
    if domain:
        try:
```

```
51              socket.gethostbyname_ex(domain)
52              num = True
53          except:
54              num = False
55          if not num:
56              try:
57                  domain = ("www.%s"%domain)
58                  socket.gethostbyname_ex(domain)
59              except:
60                  print('Unable to resolve "%s"'%domain)
61                  sys.exit(ExitUnknown)
62
63          dig_requests = os.popen('dig @%s %s +noall +comments +dnssec'%(
    opts.dnsserver, domain)).read()
64          if "ad;" in dig_requests:
65              ad_flag = 1
66          else:
67              ad_flag = 0
68          if "SERVFAIL," in dig_requests:
69              status_error = 1
70          else:
71              status_error = 0
72          if "NOERROR," in dig_requests:
73              status_noerror = 1
74          else:
75              status_noerror = 0
76
77          if ad_flag == 1 and status_noerror == 1:
78              print('The domain "%s" is safe because it use a valid
    DNSSEC.'%domain)
79              sys.exit(ExitOK)
80          elif status_error == 1:
81              print('The domain "%s" uses DNSSEC, but this is invalid or
    misconfigured.'%domain)
82              sys.exit(ExitWarning)
83          elif status_noerror == 1 and ad_flag == 0:
84              print('Domain "%s" does not use DNSSEC' %domain)
85              sys.exit(ExitCritical)
86          else:
87              print("Can't read the result")
88              sys.exit(ExitUnknown)
89      else:
90          print ("Impossible to check domains")
91          sys.exit(ExitUnknown)
92
93  def main():
94      parser = OptionParser("usage: %prog [options] ARG1 FOR EXAMPLE: -H
    www.ciencias.ulisboa.pt")
95      parser.add_option("-H","--domain", type=str,
96                        dest="domain", help="Domain name for check DNSSEC
    , for example: -H www.ciencias.ulisboa.pt")
97      parser.add_option("-d","--dnsserver", type=str, default="8.8.8.8",
    dest="dnsserver",
98                        help="Specify the DNS server you need to use for
    check DNSSEC, for example: -d 127.0.0.1, default value is 8.8.8.8")
99      parser.add_option("-V","--version", action="store_true", dest="
```

```
    version",
100                     help="This option show the current version number
     of the program and exit")
101     parser.add_option("-A","--author", action="store_true", dest="
    author",
102                     help="This option show author information and
    exit")
103
104     (opts, args) = parser.parse_args()
105     if opts.author:
106         print(__author__)
107         sys.exit()
108     if opts.version:
109         print("check_dnssec.py %s"%__version__)
110         sys.exit()
111     if not opts.domain:
112         parser.error("Please, this program requires domain arguments,
    for example: -H www.ciencias.ulisboa.pt.")
113
114     testdnssec(opts)
115
116 if __name__ == '__main__':
117     main()
```

## A.8 Código fonte do módulo check_filechange.py

```
#! /usr/bin/env python
'''
* Description:
* This file contains the check_defacing plugin
* Script to monitor files and directories, alerting you whenever an
    event occurs
*
* Creation date: 24/02/2017
* Date last updated: 19/03/2017
*
* License: GPL
* Copyright (c) 2017 DI-FCUL, Raimundo Chipongue
*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program.  If not, see <http://www.gnu.org/licenses/>.
*****************************************************************************
'''

import os
import sys
from optparse import OptionParser
from time import strftime
import time
import datetime

__author__ = "\nAuthor: Raimundo Henrique da Silva Chipongue\nE-mail:
    fc48807@alunos.fc.ul.pt, chipongue1@gmail.com\nInstitution: Faculty
    of Science of the University of Lisbon\n"
__version__= "1.0.0"

# define exit codes
ExitOK = 0
ExitWarning = 1
ExitCritical = 2
ExitUnknown = 3

def checklogs(opts):

    path_files = opts.files
    files = []
    files.extend([i for i in opts.files.split(",")])
    for i in range(0,len(files)):
```

```
50          if not os.path.exists(files[i]):
51              print("The specified file or directory does not exist: %s"%
    files[i])
52              sys.exit(ExitUnknown)
53
54      path_files = path_files.replace(",", " ")
55      processesname = 'inotifywait -mrd'
56      tmp = os.popen("ps -Af").read()
57      proccount = tmp.count(processesname)
58      if proccount > 0:
59          process = True
60      else:
61          process = False
62      if process == False:
63          timefmt = "%d/%m/%y %H:%M"
64          format_ = "%T %w %e %f"
65          try:
66              os.popen("inotifywait -mrd --timefmt '%s' --format '%s' %s
    -e %s -o %s"%(timefmt,format_,path_files, opts.events, opts.path))
67          except:
68              print("Could not execute the inotify")
69              sys.exit(ExitUnknown)
70
71      totalerror = int(os.popen("cat %s | wc -l"%opts.path).read())
72
73      if totalerror:
74          print('Critical - Were found in inotify log file %s alerts, see
     the file "%s"'%(totalerror, opts.path))
75          sys.exit(ExitCritical)
76      else:
77          print('Ok - No change were found in inotify log file "%s"'%opts
    .path)
78          sys.exit(ExitOK)
79
80  def main():
81      parser = OptionParser("usage: %prog [options] ARG1 ARG2 FOR EXAMPLE
    : -p"+
82                            " /var/log/inotify/inotify.log -f /tmp/,/home
    /cgs/ -e access,modify,attrib")
83      parser.add_option("-p","--path", dest="path", default="/var/log/
    inotify/inotify.log",
84                        help="Enter the full path to the inotify logs
    file, i.e. -p /var/log/inotify/inotify.log")
85      parser.add_option("-f","--files", dest="files", default=False, type
    ="string",
86                        help="Use this option to specify the files or
    directories you need to monitore,"+
87                        " separated by comma ',', for ex. '-f /tmp/,/home
    /cgs/Documents/wp_version.php'")
88      parser.add_option("-e","--events", dest="events", default="access,
    modify,attrib,open,move,create,delete", type="string",
89                        help="Use this option to specify the events you
    need to monitore, default values is"+
90                        " 'access, modify, attrib, open, move, create,
    delete'")
91      parser.add_option("-V","--version", action="store_true", dest="
```

```
    version", help="This option show the current version"+
92                      " number of the program and exit")
93      parser.add_option("-A","--author", action="store_true", dest="
    author", help="This option show author information and exit")
94      (opts, args) = parser.parse_args()
95
96      if opts.author:
97          print(__author__)
98          sys.exit()
99      if opts.version:
100         print("check_apache404.py %s"%__version__)
101         sys.exit()
102     if not opts.files:
103         parser.error("Please, this program requires to specify file or
    directory to monitore.")
104
105     if opts.path:
106         if not os.path.exists(opts.path):
107             parser.error("Please, this program requires to specify a
    valid log file.")
108         else:
109             checklogs(opts)
110     else:
111         parser.error("Please, this program requires to specify a valid
    path file.")
112
113 if __name__ == '__main__':
114     main()
```

## A.9 Código fonte do módulo check_nikto.py

```
#! /usr/bin/env python3
'''
* Description:
* This file contains the check_nikto plugin
* Script to scan websites looking for web vulnerabilities
*
* Creation date: 05/03/2017
* Date last updated: 19/03/2017
*
* License: GPL
* Copyright (c) 2017 DI-FCUL, Raimundo Chipongue
*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program.  If not, see <http://www.gnu.org/licenses/>.
*****************************************************************************
'''

import socket
from datetime import datetime
from collections import Counter
import time
import codecs
import sys
import urllib.request
import re
import urllib
from optparse import OptionParser
import os

__author__ = "\nAuthor: Raimundo Henrique da Silva Chipongue\nE-mail:
    fc48807@alunos.fc.ul.pt, chipongue1@gmail.com\nInstitution: Faculty
     of Science of the University of Lisbon\n"
__version__= "1.0.0"

# define exit codes
ExitOK = 0
ExitWarning = 1
ExitCritical = 2
ExitUnknown = 3

def check_connectivity():
    '''
```

```
51      Check if the internet conection is up
52      '''
53      try:
54          urllib.request.urlopen('https://google.com', timeout=1)
55          return True
56      except urllib.request.URLError:
57          return False
58
59  def version(opts):
60      if check_connectivity():
61          report = ("%s.html"%opts.report)
62          time_to_check = 60 * 60 * 24 * opts.time
63          domain = opts.host
64          domain = domain.replace("https://", "")
65          domain = domain.replace("http://", "")
66          domain = domain.replace("www.", "")
67          if domain:
68              try:
69                  socket.gethostbyname_ex(domain)
70                  num = True
71              except:
72                  num = False
73                  if not num:
74                      try:
75                          domain = ("www.%s"%domain)
76                          socket.gethostbyname_ex(domain)
77                      except:
78                          print('Unable to resolve "%s"'%domain)
79
80          if os.path.exists("%s%s"%(opts.path,report)):
81              file_ts = float(os.path.getatime("%s%s"%(opts.path,report))
    )
82              current_ts = float(time.time())
83              dif = current_ts - file_ts
84              if dif > time_to_check:
85                  if opts.tuning:
86                      tuning = opts.tuning.replace(",", "")
87                      os.popen("rm %s%s"%(opts.path,report))
88                      os.popen('nikto -Tuning %s -h %s -output %s%s -
    Format html -port %s'%(tuning, domain, opts.path, report, opts.port
    )).read()
89                  else:
90                      os.popen("rm %s%s"%(opts.path,opts.report))
91                      os.popen('nikto -h %s -output %s%s -Format html -
    port %s'%(domain, opts.path, report, opts.port)).read()
92          else:
93              if opts.tuning:
94                  tuning = opts.tuning.replace(",", "")
95                  os.popen('nikto -Tuning %s -h %s -output %s%s -Format
    html -port %s'%(tuning, domain, opts.path, report, opts.port)).read
    ()
96              else:
97                  os.popen('nikto -h %s -output %s%s -Format html -port %
    s'%(domain, opts.path, report, opts.port)).read()
98          try:
99              size = os.path.getsize("%s%s"%(opts.path,report))
```

```
100         except:
101             print("Unable to read the file, verify that it exists and
    the nagios user has permissions of writing and reading in this path
    .")
102             sys.exit(ExitUnknown)
103         if size > 0:
104             try:
105                 file = codecs.open("%s%s"%(opts.path,report))
106             except:
107                 print("report error")
108                 sys.exit(ExitUnknown)
109             osvdb = []
110             end_scan = False
111             for line in file:
112                 x = re.search('>OSVDB-[0-9]?\d', line)
113                 scan_summary = re.search('Scan Summary', line)
114                 if scan_summary:
115                     end_scan = True
116                 if x:
117                     osvdb.extend(re.findall(r'>(.*?)<', str(line)))
118             if end_scan:
119                 num = (len(osvdb))
120                 if num != len(set(osvdb)):
121                     class MyCounter(Counter):
122                         def __str__(self):
123                             return ("\n".join('{} {}'.format(k, v) for
    k, v in self.items()))
124
125                     vuln = MyCounter(Counter(osvdb))
126                     vuln = str(vuln).replace("\n", ", ")
127
128                 else:
129                     vuln = (', '.join(osvdb))
130
131                 if num:
132                     print("Were found the folowing %s vulnerabilities %
    s, in %s, please open the file %s%s for more details."%(num, vuln,
    domain, opts.path, report))
133                     sys.exit(ExitCritical)
134                 else:
135                     print("The Nikto web scan didn't find any
    vulnerability in %s"%domain)
136                     sys.exit(ExitOK)
137             else:
138                 print("Scan in progress")
139                 sys.exit(ExitUnknown)
140         else:
141             print("Can't read file")
142             sys.exit(ExitUnknown)
143     else:
144         print('Error, check you internet connection')
145         sys.exit(ExitUnknown)
146
147 def main():
148     parser = OptionParser("usage: %prog [options] arg1 arg2 arg3 arg4.
    \nEx.: %prog -H ciencias.ulisboa.pt -p 80 -r ciencias -t 2 -T 9" )
```

```
149     parser.add_option("-H","--host", dest="host", help="Use this
    options to specify the target host")
150     parser.add_option("-P", "--path", dest="path", default="/tmp/",
    type="string",
151                       help="Use this option to specify the path to save
     the nikto report." +
152                       " By default this is /tmp folder, however this
    doesn't secure, any malicious user can change the" +
153                       " your content, and generating false positives or
     false negatives." +
154                       " If the folder is different from /tmp, you must
    assign read and write permissions to nagios.")
155     parser.add_option("-p", "--port", dest="port", default="443",
156                       help="Use this option to specify the port or
    ports", type="string")
157     parser.add_option("-r", "--report", dest="report", default="report"
    ,
158                       help="Use this option to specify the report name"
    , type="string")
159     parser.add_option("-t", "--time", dest="time", default=1,
160                       help="Use this option to specify the life time
    for report file, life time is in day.", type=float)
161     parser.add_option("-T", "--tuning", dest="tuning", default=False,
162                       help="Use this option to select the tuning mode,
    and specify the options you need to work."+
163                       "Using Tuning mode, this implies the selection of
     tests, running in short time,"+
164                       "that run in normal mode, takes a several minutes
    ."+
165                       " | 0 - File Upload"+
166                       " | 1 - Interesting File // we will get in logs"+
167                       " | 2 - Misconfiguration // Default File"+
168                       " | 3 - Information Disclousure"+
169                       " | 4 - Injection (XSS/Script/HTML)"+
170                       " | 5 - Remote File Retrievel - Inside Web Root"+
171                       " | 6 - Denial of Service // Scan for DDOS"+
172                       " | 7 - Remote File Retrieval - Server Wide"+
173                       " | 8 - Command Execution // Remote Shell"+
174                       " | 9 - SQL Injection // Scan for mysql
    vulnerabilities"+
175                       " | a - Authentication Bypass"+
176                       " | b - Software Identification"+
177                       " | c - Remote Source Inclusion"+
178                       " | x - Reverse Tuning Options",
179                       type="string")
180     parser.add_option("-V","--version", action="store_true", dest="
    version", help="This option show the current version number of the
    program and exit")
181     parser.add_option("-A","--author", action="store_true", dest="
    author", help="This option show author information and exit")
182     (opts, args) = parser.parse_args()
183     if not os.path.exists(opts.path):
184         parser.error("Please, this program requires to specify a valid
    and private folder to store temp file")
185     if opts.author:
186         print(__author__)
```

```
187             sys.exit()
188         if opts.version:
189             print("check_nikto.py %s"%__version__)
190             sys.exit()
191 
192         if not opts.host:
193             parser.error("This script requires to specify host arguments.")
194             sys.exit(ExitUnknown)
195         version(opts)
196 if __name__ == '__main__':
197         main()
```

## A.10 Código fonte do módulo check_open_port.py

```
#!/usr/bin/env python
'''
* Description:
* This file contains the check_open_port plugin to check open ports
*
* Creation date: 19/01/2017
* Date last updated: 19/03/2017
*
* License: GPL
* Copyright (c) 2017 DI-FCUL, Raimundo Chipongue
*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program.  If not, see <http://www.gnu.org/licenses/>.
*************************************************************************
'''

import socket
import subprocess
import sys
from optparse import OptionParser

__author__ = "\nAuthor: Raimundo Henrique da Silva Chipongue\nE-mail:
    fc48807@alunos.fc.ul.pt, chipongue1@gmail.com\nInstitution: Faculty
     of Science of the University of Lisbon\n"
__version__= "1.0.0"

# define exit codes
ExitOK = 0
ExitWarning = 1
ExitCritical = 2
ExitUnknown = 3

def scan(opts):
    listfound = []
    try:
        for port in range(1,7000):
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            result = sock.connect_ex((opts.host, port))
            if result == 0:
                listfound.extend([int(i) for i in ("{}".format(port)).
    split(" ")])
        return listfound
```

```
50      except:
51          print("Error, unable to scan")
52          sys.exit(ExitUnknown)
53  def getopenport(opts):
54      try:
55          authorized_ports = [int(i) for i in opts.port.split(",")]
56          authorized_ports = sorted(authorized_ports)
57      except:
58          print("Error, check list of authorized ports, i.e.: -p
    500,21,23,80,3333")
59          sys.exit(ExitUnknown)
60      all_open_ports = scan(opts)
61      unauthorized_ports = sorted((list(set(all_open_ports) - (set(
    authorized_ports)))))
62      num_anauth_open_ports = len(unauthorized_ports)
63      if num_anauth_open_ports == 0:
64          print("Not found any anauthorized port open.")
65          sys.exit(ExitOK)
66      else:
67          unauthorized_ports = ", ".join(str(x) for x in
    unauthorized_ports)
68          print("Were found the following %s unauthorized open ports: %s!
    " %(num_anauth_open_ports, unauthorized_ports))
69          sys.exit(ExitCritical)
70
71  def main():
72      parser = OptionParser("usage: %prog -H <IP address> and -p <port or
     list of ports>, that have been authorized to are open, e.i. <-p
    500,21,23,80,3333>")
73      parser.add_option("-H","--hostaddress", dest="host", help="Specify
    the IP address you want to check")
74      parser.add_option("-p","--port", dest="port", default="0", help="
    Specify the ports or list of allowed ports to be open, i.e.: <-p
    500,21,23,80,3333>")
75      parser.add_option("-V","--version", action="store_true", dest="
    version", help="This option show the current version number of the
    program and exit")
76      parser.add_option("-A","--author", action="store_true", dest="
    author", help="This option show author information and exit")
77      (opts, args) = parser.parse_args()
78      if opts.author:
79          print(__author__)
80          sys.exit()
81      if opts.version:
82          print("check_open_port.py %s"%__version__)
83          sys.exit()
84      if opts.host and opts.port:
85          try:
86              ServerIP = socket.gethostbyname(opts.host)
87          except:
88              parser.error("Incorrect IP Address.")
89          getopenport(opts)
90      else:
91          parser.error(" This program requires at least one argument.")
92  if __name__ == '__main__':
93      main()
```

## A.11 Código fonte do módulo check_ssl.py

```
#! /usr/bin/env python3
'''
* Description:
* This file contains the check_ssl plugin
* Script to check the status of ssl, through the api (https://api.
    ssllabs.com/api/v2/)
* for ssllabs (ssllabs.com)
*
* Creation date: 12/02/2017
* Date last updated: 18/03/2017
*
* License: GPL
* Copyright (c) 2017 DI-FCUL, Raimundo Chipongue
*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program.  If not, see <http://www.gnu.org/licenses/>.
*************************************************************************

'''

import sys
from optparse import OptionParser
import os
import json
import time
import urllib
import socket
import ssl
import datetime
import time
import requests

__author__ = "\nAuthor: Raimundo Henrique da Silva Chipongue\nE-mail:
    fc48807@alunos.fc.ul.pt, chipongue1@gmail.com\nInstitution: Faculty
    of Science of the University of Lisbon\n"
__version__= "1.0.0"

# define exit codes
ExitOK = 0
ExitWarning = 1
ExitCritical = 2
ExitUnknown = 3

```

```
50 def ssl_expiry_datetime(hostname):
51     ssl_date_fmt = r'%b %d %H:%M:%S %Y %Z'
52 
53     context = ssl.create_default_context()
54     conn = context.wrap_socket(socket.socket(socket.AF_INET),
       server_hostname=hostname,)
55 
56     conn.settimeout(3.0)
57 
58     conn.connect((hostname, 443))
59     ssl_info = conn.getpeercert()
60     expriration date = datetime.datetime.strptime(ssl_info['notAfter'],
       ssl_date_fmt)
61     date = []
62     ts = str(expriration date)
63     ts = ts.replace("-", ",")
64     ts = ts.replace(" ", ",")
65     ts = ts.replace(":", ",")
66     Ig = [i for i in ts.split(",")]
67     date.extend([int(i) for i in Ig])
68     dt = datetime.datetime(int(date[0]), int(date[1]), int(date[2]),
       int(date[3]), int(date[4]), int(date[5]))
69     exp_timestemp = float(time.mktime(dt.timetuple()))
70     expriration date = str(expriration date)
71     return exp_timestemp, expriration date
72 
73 def getgrade(payload, num, opts):
74 
75     API_url = 'https://api.ssllabs.com/api/v2/analyze?'
76     response = requests.get(API_url, params=payload)
77     status = response.json()["status"]
78 
79     #response = urllib2.urlopen("%s%s"%(API_url, urllib.urlencode(
       payload)))
80     #status = json.load(response)["status"]
81 
82     if status != "READY" and num == 2:
83         time.sleep(opts.sleep)
84         response = requests.get(API_url, params=payload)
85         status = response.json()["status"]
86         #response = urllib2.urlopen("%s%s"%(API_url, urllib.urlencode(
       payload)))
87         #status = json.load(response)["status"]
88     elif status == "READY":
89         response = requests.get(API_url, params=payload)
90         message = response.json()["endpoints"][0]
91         #response = urllib2.urlopen("%s%s"%(API_url, urllib.urlencode(
       payload)))
92         #message = json.load(response)["endpoints"][0]
93         if message["statusMessage"] == "No secure protocols supported":
94             print(message["statusMessage"])
95             sys.exit(ExitCritical)
96         elif message["statusMessage"] == "Ready":
97             response = requests.get(API_url, params=payload)
98             grade = response.json()["endpoints"][0]
99             #response = urllib2.urlopen("%s%s"%(API_url, urllib.
```

```
    urlencode(payload)))
100             #grade = json.load(response)["endpoints"][0]
101             grade = str((grade["grade"]))
102             return (grade)
103         else:
104             return False
105
106     else:
107         return False
108
109 def getcacheresult(opts, publish='off', startNew='off', fromCache='on',
     all='done'):
110     payload = {
111         "host": opts.domain,
112         "publish": publish,
113         "startNew": startNew,
114         "fromCache": fromCache,
115         "all": all}
116     num = 1
117     grade = getgrade(payload, num, opts)
118
119     return grade
120
121 def getnewScanresult(opts, publish="off", startNew="on", all="done",
    ignoreMismatch="on"):
122     payload = {
123         "host": opts.domain,
124         "publish": publish,
125         "startNew": startNew,
126         "all": all,
127         "ignoreMismatch": ignoreMismatch}
128     num = 2
129     grade = getgrade(payload, num, opts)
130     return grade
131
132 def scan(opts):
133     grade = (getcacheresult(opts))
134     if not grade:
135         grade = getnewScanresult(opts)
136         return grade
137     else:
138         return grade
139
140 def testssl(opts):
141     result = scan(opts)
142     expiry_datetime = (ssl_expiry_datetime(opts.domain))
143     current_date = time.time()
144     difftime = expiry_datetime[0] - current_date
145     timetoexpire = difftime/60/60/24
146
147     try:
148         grade = []
149         grade.extend([i for i in result.split(",")])
150     except:
151         grade = False
152
```

```
153     if not opts.critical or not opts.warning:
154         critical = ["E+","E-","E","F+","F-","F","T","M"]
155         warning = ["C+","C-","C","D+","D-","D"]
156 
157     else:
158         try:
159             critical=[]
160             warning = []
161             critical.extend([i for i in opts.critical.split(",")])
162             warning.extend([i for i in opts.warning.split(",")])
163         except:
164             sys.exit(ExitUnknown)
165 
166     if grade:
167         if list(set(grade).intersection(critical)):
168             if timetoexpire <= 0:
169                 print('SSLLABS grade for %s is %s, the ssl certificate
    expired on %s'%(opts.domain, grade[0], expiry_datetime[1]))
170                 sys.exit(exitcode)
171             else:
172                 print('SSLLABS grade for %s is %s, ssl certificate
    expires in %s days'%(opts.domain, grade[0], int(timetoexpire)))
173                 sys.exit(ExitCritical)
174 
175         elif list(set(grade).intersection(warning)):
176             if timetoexpire <= 0:
177                 exitcode = 2
178                 print('SSLLABS grade for %s is %s, the ssl certificate
    expired on %s'%(opts.domain, grade[0], expiry_datetime[1]))
179                 sys.exit(exitcode)
180             else:
181                 exitcode = 1
182             print('SSLLABS grade for %s is %s, ssl certificate expires
    in %s days'%(opts.domain, grade[0], int(timetoexpire)))
183             sys.exit(exitcode)
184 
185         else:
186             if timetoexpire <= 0:
187                 exitcode = 2
188                 print('SSLLABS grade for %s is %s, the ssl certificate
    expired on %s'%(opts.domain, grade[0], expiry_datetime[1]))
189                 sys.exit(exitcode)
190             elif timetoexpire <= opts.days:
191                 exitcode = 1
192             else:
193                 exitcode = 0
194             print('SSLLABS grade for %s is %s, ssl certificate expires
    in %s days'%(opts.domain, grade[0], int(timetoexpire)))
195             sys.exit(exitcode)
196 
197     else:
198         print("Can't check SSL settings in %s"%opts.domain)
199         sys.exit(ExitUnknown)
200 
201 def main():
202     parser = OptionParser("usage: %prog [options] ARG1 ARG2 ARG3 FOR
```

```
    EXAMPLE: -H www.ciencias.ulisboa.pt, -c E+,E-,E,F+,F-,F,T,M -w C+,C
    -,C,D+,D-,D")
203     parser.add_option("-H","--domain", dest="domain", default=False,
    help="Domain name for check ssl")
204     parser.add_option("-c","--critical", dest="critical", type=str,
    default=False,
205                       help="Specify all value for ssllabs grade yo
    considered critical, Ex. -c T,M,E+,E-,E,F+,F-,F")
206     parser.add_option("-w","--warning", dest="warning", default=False,
207                       help="Specify all value for ssllabs grade yo
    considered warning, Ex. -w C+,C-,C,D+,D-,D")
208     parser.add_option("-s","--sleep", dest="sleep", default=45,type=int
    ,
209                       help="Specify the number of seconds you want to
    wait, if not found the result in cache, the defoult value is 45
    seconds")
210     parser.add_option("-d","--days", dest="days", default=30,type=int,
211                       help="specify how many days before it expires
    will be considered warning, the defoult value is 30 days")
212     parser.add_option("-V","--version", action="store_true", dest="
    version", help="This option show the current version number of the
    program and exit")
213     parser.add_option("-A","--author", action="store_true", dest="
    author", help="This option show author information and exit")
214     (opts, args) = parser.parse_args()
215
216     if opts.author:
217         print(__author__)
218         sys.exit()
219     if opts.version:
220         print("check_ssl.py %s"%__version__)
221         sys.exit()
222     if opts.domain == False:
223         parser.error("Please, this program requires domain arguments, -
    H www.ciencias.ulisboa.pt")
224
225     testssl(opts)
226 if __name__ == '__main__':
227     main()
```

## A.12 Código fonte do módulo check_wp_update.py

```
#! /usr/bin/env python3
'''
* Description:
* This file contains the check_wp_update plugin
* Script to check the status of update of wordpress, based on
* information taken from the API https://api.wordpress.org/core/version-check/1.7/
* and site https://wordpresss.org
*
* Creation date: 30/10/2016
* Date last updated: 19/03/2017
*
* License: GPL
* Copyright (c) 2017 DI-FCUL, Raimundo Chipongue
*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program.  If not, see <http://www.gnu.org/licenses/>.
*****************************************************************************
'''

import sys
import urllib.request
import re
import urllib
from optparse import OptionParser
import json
import os

__author__ = "\nAuthor: Raimundo Henrique da Silva Chipongue\nE-mail: fc48807@alunos.fc.ul.pt, chipongue1@gmail.com\nInstitution: Faculty of Science of the University of Lisbon\n"
__version__= "1.0.0"

# define exit codes
ExitOK = 0
ExitWarning = 1
ExitCritical = 2
ExitUnknown = 3

def check_connectivity():
    '''
    Check if the internet conection is up
```

```
50      '''
51      try:
52          urllib.request.urlopen('https://wordpress.org/download',
    timeout=1)
53          return True
54      except urllib.request.URLError:
55          return False
56
57  def version(opts):
58      if check_connectivity():
59          wp_installed_version_path = opts.path
60          def installed_wp_version():
61              '''
62              Get the installed WordPress version
63              '''
64              for line in open(wp_installed_version_path):
65                  if "wp_version =" in line:
66                      version_number = re.search('[-+]?\d+[\.]?\d*[\.]?\d
    *', line)
67                      if version_number:
68                          installed_version = str(version_number.group())
69                          installed_version = installed_version.replace("
    -", "")
70              return(installed_version)
71
72          installed_version = installed_wp_version()
73          def current_wp_version():
74              '''
75              Get the latest stable version WordPress
76              '''
77              api_path = "https://api.wordpress.org/core/version-check
    /1.7/"
78              result = os.popen("curl -s %s "%api_path).read()
79
80              try:
81                  latest = json.loads(result)["offers"][0]
82                  current_version = (latest["version"])
83              except ValueError:
84                  return False
85
86              if current_version == installed_version:
87                  print('The latest stable version WordPress %s available
     in wordpress.org is installed'%current_version)
88                  sys.exit(ExitOK)
89              else:
90                  print('Version outdated, has installed WordPress %s,
    but is available in wordpress.org the version %s' %(
    installed_version, current_version))
91                  sys.exit(ExitCritical)
92
93          wp_current_version = current_wp_version()
94          if not wp_current_version:
95              '''
96              Get the latest stable version WordPress
97              '''
98              wp_current_version_url = "https://wordpress.org/download"
```

```
99              values = {'s':'wordpress', 'submit':'search'}
100             data = urllib.parse.urlencode(values)
101             data = data.encode('utf-8')
102             req = urllib.request.Request(wp_current_version_url, data)
103             resp = urllib.request.urlopen(req)
104             respData = resp.read()
105             text_version = re.findall(r'<strong>(.*?)</strong>', str(
    respData))
106             for eachP in text_version:
107                 version_number = re.search('[-+]?\d+[\.]?\d*[\.]?\d*',
    eachP)
108                 if version_number:
109                     current_version = str(version_number.group())
110                     current_version = current_version.replace("-", "")
111                     if current_version == installed_version:
112                         print('The latest stable version WordPress %s
    available in wordpress.org is installed' %current_version)
113                         sys.exit(ExitOK)
114                     else:
115                         print('Version outdated, has installed
    WordPress %s, but is available in wordpress.org the version %s' %(
    installed_version, current_version))
116                         sys.exit(ExitCritical)
117
118             print('Error, Connot read the current stable WordPress
    Version')
119             sys.exit(ExitUnknown)
120
121     else:
122         print('Error, check you internet connection')
123         sys.exit(ExitUnknown)
124
125 def main():
126     parser = OptionParser("usage: %prog [options] arg1. \nEx.: %prog -p
    /var/www/html/wp-includes/version.php" )
127     parser.add_option("-p", "--path", dest="path",
128                       help="specify full path of version.php in wp
    folder installation", type="string")
129     parser.add_option("-V","--version", action="store_true", dest="
    version", help="This option show the current version number of the
    program and exit")
130     parser.add_option("-A","--author", action="store_true", dest="
    author", help="This option show author information and exit")
131     (opts, args) = parser.parse_args()
132     if opts.author:
133         print(__author__)
134         sys.exit()
135     if opts.version:
136         print("check_wp-update.py %s"%__version__)
137         sys.exit()
138
139     if not opts.path:
140         parser.error("This program requires at least one argument")
141         sys.exit(ExitUnknown)
142
143     if opts.path:
```

```
144         if not os.path.exists(opts.path):
145             parser.error("Please, this program requires to specify a
    valid path file.")
146         else:
147             version(opts)
148     else:
149         parser.error("Please, this program requires to specify a valid
    path path file.")
150
151 if __name__ == '__main__':
152     main()
```

# Apêndice B

# Código Fonte dos Ficheiros setup.py e .spec

## B.1 Código fonte do ficheiro setup.py

```
from setuptools import setup, find_packages
import os
import subprocess
import sys

gcc = os.popen("whereis gcc | grep /usr/bin/gcc").read()
if not gcc:
    print("This program requires the gcc compiler, install it first and try again")
    sys.exit()

make = os.popen("whereis make | grep /usr/bin/make").read()
if not make:
    print("This program requires the make installer, install it first and try again")
    sys.exit()

make = os.popen("whereis wget | grep /usr/bin/wget").read()
if not wget:
    print("This program requires the wget utilities, install it first and try again")
    sys.exit()

python3 = os.popen("whereis python3 | grep python3.[1-9]").read()
if not python3:
    print("Downloading Python-3.5.1 to /tmp...")
    subprocess.call(["sudo wget https://www.python.org/ftp/python/3.5.1/Python-3.5.1.tgz -P /tmp/"], shell=True)
    print("Uncompress Python-3.5.1.tgz...")
    subprocess.call(["sudo tar -xzvf /tmp/Python-3.5.1.tgz -C /tmp/"], shell=True)
    print("Configure and install Python-3.5.1.tgz")
    subprocess.call(["cd /tmp/Python-3.5.1 && ./configure && make && sudo make altinstall"], shell=True)
    print("Remove temp files /tmp/Python-3.5.1 and /tmp/Python-3.5.1.
```

```
    tgz")
30      subprocess.call(["sudo rm -fr /tmp/Python-3.5.1 /tmp/Python-3.5.1.
    tgz"], shell=True)
31
32  setup(
33      # Application name:
34      name="check_ssl",
35
36      scripts=['check_ssl.py'],
37
38      # Version number (initial):
39      version="1.0.0",
40
41      # Application author details:
42      author="Raimundo Henrique da Silva Chipongue",
43      author_email="chipongue1@gmail.com",
44
45      # Include additional files into the package
46      include_package_data=True,
47
48      # Details
49      dependency_links=["https://github.com/python/cpython/tree/3.6/Lib/"
    ,"https://github.com/pypa/"],
50
51      # Packages
52      packages=find_packages(),
53
54      # License
55      license="GPL",
56
57      # Plugin description
58      description="plugin designed to warn about problems in the config
    or the validity of the digital certificates.",
59      long_description=open("README.txt").read(),
60
61      # Dependent packages (distributions)
62      install_requires=["requests"]
63      )
```

## B.2 Código fonte do ficheiro .spec

```
%define       name check_ssl
%define       version 1.0.0
%define       release 1
%define       plugins_path /usr/lib64/nagios/plugins/
%define       nrpe_file /etc/nagios/nrpe.cfg
%define       nagiosip 192.168.1.144

Name:             %{name}
Version:          %{version}
Release:          %{release}
Summary:          Nagios plugin to check ssl validity and configurations
BuildArch:      noarch
Group:            Development/Libraries
License:          GPL
URL:              http://www.ciencias.ulisboa.pt
Source0:          %{name}-%{version}.tar.gz
Vendor:       Raimundo Henrique da Silva Chipongue <chipongue1@gmail.com>
BuildRoot:        %{_tmppath}/%{name}-%{version}-%{release}-root-%(%{
    __id_u} -n)
Requires:         python34,python34-setuptools,python34-requests,nrpe

%description
check_ssl.py
plugin designed to warn about problems in the config or the validity of
     the digital certificates.

%prep
%setup -q

%install
mkdir -p $RPM_BUILD_ROOT%{plugins_path}
cp -R %{name}.py "$RPM_BUILD_ROOT%{plugins_path}"

%files
%defattr(0644,root,root)
%{plugins_path}*

%clean
rm -rf $RPM_BUILD_ROOT
%post
chown -R root:root %{plugins_path}%{name}.py
chmod -R 775 %{plugins_path}%{name}.py
sed -i '/allowed_hosts=/c\allowed_hosts=%{nagiosip}' %{nrpe_file}
sed -i '/# The following examples use hardcoded command arguments.../a\
    command[%{name}]=%{plugins_path}%{name}.py -H www.ciencias.ulisboa.
    pt' %{nrpe_file}
systemctl restart nrpe
```