



INSTITUTO POLITÉCNICO DE LEIRIA
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO
Segurança de Sistemas



PROJETO DE SEGURANÇA DE SISTEMAS

Bruno Alves 2171033, Dinis Abreu 2171199, João Lemos 2161508

Tiago Caetano 2181830, Tiago Garcia 2170959

Leiria, Janeiro de 2020

Índice

Introdução.....	3
PLANEAMENTO INICIAL.....	3
PLANEAMENTO INICIAL VS PROJETO FINAL.....	4
Certificados Digitais.....	6
Servidor Web público com HTTPS.....	6
Alojamento e SSH.....	6
Domínio.....	7
Criação do certificado digital Let's Encrypt.....	7
Secure FTP com o certificado Let's Encrypt.....	7
Autenticação por Tokens	8
DNS e HTTP seguro na rede privada	11
Servidor Web privado com HTTPS	11
Servidor DNS	11
DMZ.....	12
SSH com fail2ban.....	16
Pfsense (rede restrita dentro da rede DA SEDE).....	16
Servidor NAS (FreeNAS)	20
Servidor ADMIN (para acesso a FreeNAS)	21
Análise de Vulnerabilidades.....	21
Análise de vulnerabilidades com o Nessus	22
Conclusão	28
Pesquisas bibliográficas	29

AGRADECIMENTOS

Finalizado este projeto queremos expressar os nossos sinceros agradecimentos a todos aqueles que estiveram presentes e que de alguma maneira contribuíram para o presente relatório.

Ao nosso orientador, o Professor Doutor Miguel Frade, pela ajuda e apoio prestado. A todos, agradecemos!

INTRODUÇÃO

O presente relatório foi realizado no âmbito da Unidade Curricular de Segurança de Sistemas, pertencente ao último ano do curso de Engenharia Informática, lecionado na Escola Superior de Tecnologia e Gestão de Leiria (ESTG), do Instituto Politécnico de Leiria (IPL).

Este documento tem como propósito explicar o trabalho que foi desenvolvido e mostrar com clareza os passos até ao resultado final.

Devido a excesso de informação, haverá algumas partes do projeto que não estão exatamente como foi descrito no planeamento inicial e alterámos. Essas alterações serão abordadas neste documento, tanto o que foi alterado ou removido, como a razão de tal ter acontecido.

Este projeto visa transmitir, de forma clara, os nossos conhecimentos e de que forma os colocámos em prática no decorrer desta unidade curricular.

PLANEAMENTO INICIAL

O planeamento inicial, tal como se pode ver na figura abaixo, passava por ter uma filial e uma sede, a sede era dividida em Rede Local e na DMZ e a restante filial apenas tinha a Rede Local.

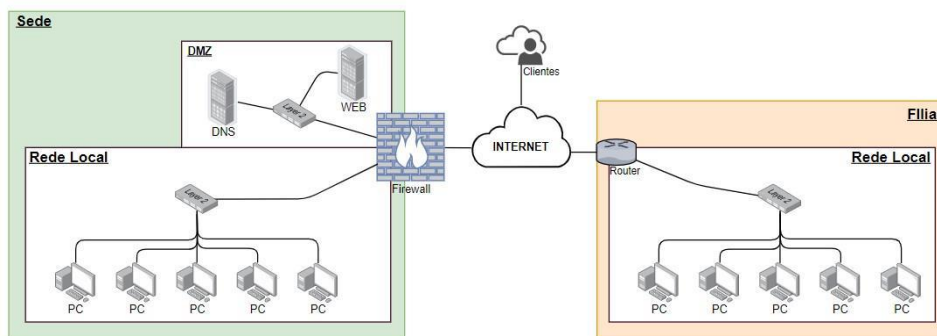


FIGURA 1 - Planeamento Inicial.

Filial

- Cinco clientes – Representados por um computador com o sistema operativo windows 7.
- Um switch layer 2 – Permite a ligação dos clientes ao router e posteriormente à internet.
- Um router – Este irá ter uma VPN site-to-site com a firewall possibilitando assim a comunicação entre os computadores e a filial.

Sede

- Uma rede local – Contém cinco clientes representados por um computador com o sistema operativo windows 7.
- Dois switch layer 2 – Permite a ligação dos clientes à firewall e posteriormente à internet.
- Uma DMZ – Proporciona o isolamento dos servidores acrescentado uma camada externa de segurança, permitindo o acesso externo mas separados da rede local.
- Um servidor WEB - Para fornecer uma página web do domínio a explicar o porquê de om606.
- Um servidor DNS – Configurando um servidor master e um slave para resolver os nomes do domínio om606.ss.pt

PLANEAMENTO INICIAL VS PROJETO FINAL

O projeto final, quer por dificuldades ou apenas porque à medida que foi sendo desenvolvido foram surgindo novas ideias, têm algumas diferenças em relação ao planeamento inicial.

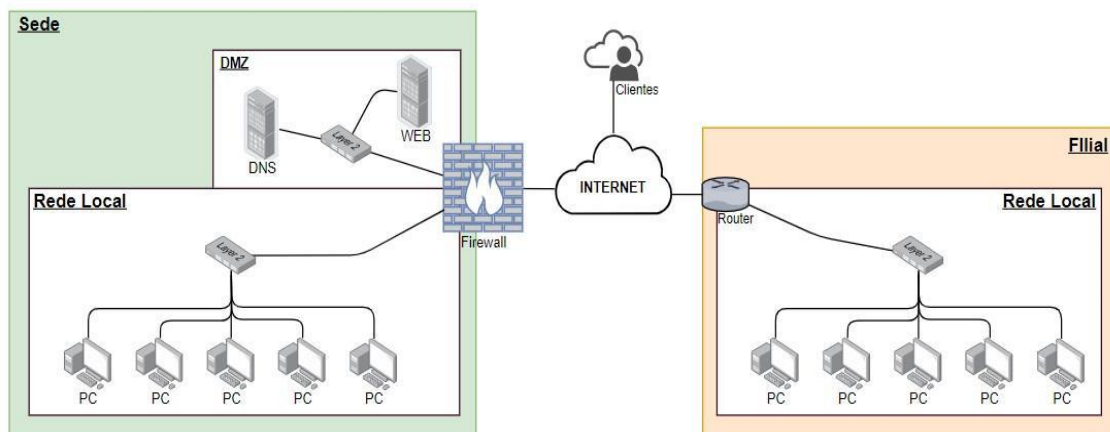


FIGURA 2 - (Figura 1) Planeamento Inicial

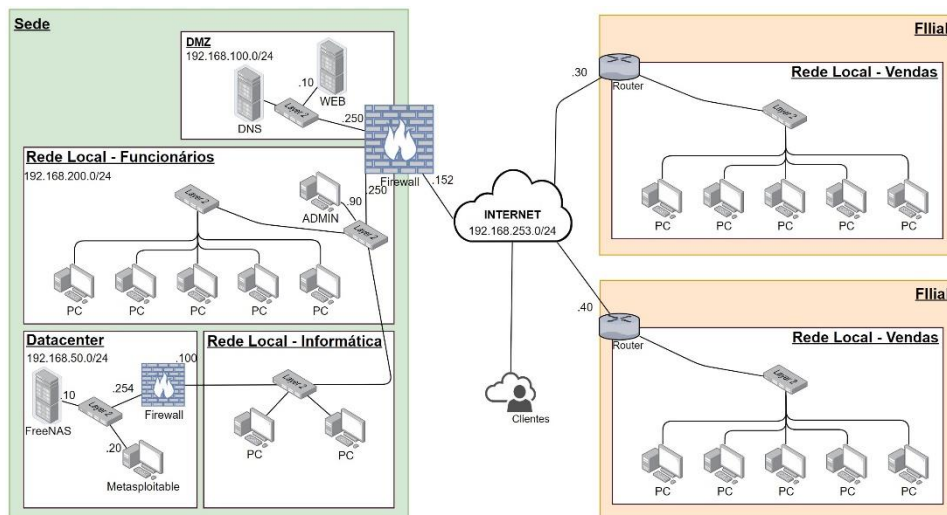


FIGURA 3 - Diagrama da implementação final

FIGURA 3 – Diagrama da implementação final.

Filial (canto inferior direito da figura 3.)

- Cinco clientes – Representados por um computador com o sistema operativo windows 7.
- Um switch layer 2 – Permite a ligação dos clientes ao router e posteriormente à internet.
- Um router – Este irá ter uma VPN site-to-site com a firewall possibilitando assim a comunicação entre os computadores e a filial.

Filial

- Cinco clientes – Representados por um computador com o sistema operativo windows 7.
- Um switch layer 2 – Permite a ligação dos clientes ao router e posteriormente à internet.
- Um router – Este irá ter uma VPN site-to-site com a firewall possibilitando assim a comunicação entre os computadores e a filial.

Sede

- Uma rede local - Funcionários – Contém cinco clientes representados por um computador com o sistema operativo windows 7.
- Uma rede local - Informática - Contém dois computadores e um switch layer 2.
- Dois switch layer 2 – Permite a ligação dos clientes à firewall e posteriormente à internet.
- Uma DMZ – Proporciona o isolamento dos servidores acrescentado uma camada externa de segurança, permitindo o acesso externo mas separados da rede local.
- Um Servidor WEB – Contém https sobre um certificado assinado localmente sem passar por uma CA.
- Um Servidor DNS – Configurado com master e slave e com DoT (DNS over TLS)

CERTIFICADOS DIGITAIS

Servidor Web público com HTTPS

Atualmente, qualquer empresa necessita de um website, seja este para comércio de produtos, seja apenas para uma página de contactos, tendo isso em conta, achámos por bem também implementar um website para a nossa empresa fictícia de projeto.

Apesar de ser por âmbito da UC, também não faz sentido uma empresa ter um website inseguro simplesmente por tornar questionáveis as suas intenções e ambições, tendo isso em conta, faz apenas sentido que o nosso website tenha HTTPS para a internet pública. Para isso é necessário ter um website com um IP público fixo e um domínio fixo (domínios dinâmicos não funcionam).

Alojamento e SSH

Para ter um servidor num IP público fixo, fizemos uso do serviço gratuito para estudantes da Azure, criámos um servidor Ubuntu 18.04 com o modelo da Azure, instalámos o nginx e configurámos o SSH do mesmo para usar chaves pré-partilhadas, desta forma só precisámos da password da chave criada (se existir, é possível criar sem password) para aceder ao servidor por SSH.

Para que possa testar irá ser anexado ao projeto uma chave privada com o nome 'prof' para que possa fazer login no servidor. Se precisar da password é 'prof'.

Desta forma evitamos problemas de autenticação por *brute-force*, visto ser um servidor aberto ao público isso é importante. Idealmente, também devíamos ter alterado o porto do SSH, para que consigamos trabalhar na escola, temos de usar o porto 22 por defeito.

Domínio

Com o servidor criado, o próximo passo para implementar HTTPS é ter um domínio fixo, para isso fizemos uso de uma conta no *name.com* com conta de estudante do GitHub ^[1] e registámos o domínio **ssproject.team** ^[2] onde também registámos o IP do nosso servidor no DNS do domínio. Como geralmente os websites utilizam o *www* para identificar a máquina web do seu domínio criamos um CNAME record (Canonical Name record) para o www.ssproject.team e um A record (Address record) para o IP do servidor. Assim, se o utilizador aceder ao *www* ou diretamente ao domínio, é sempre redirecionado para o IP do servidor Web. Isto é importante porque alguns browsers (nomeadamente o chrome) deixaram de adicionar o *www* automaticamente aos endereços URL e dessa forma não iram encontrar o endereço do servidor web.

Type	Host	Answer	TTL	Prio	Actions
<input type="checkbox"/> A	ssproject.team	52142.222.172	300	N/A	Edit Delete
<input type="checkbox"/> CNAME	www.ssproject.team	ssproject.team	300	N/A	Edit Delete

FIGURA 4 - DNS Records do domínio

Criação do certificado digital Let's Encrypt

O próximo passo para o HTTPS é o registo de um certificado digital para o domínio. Para isso fizemos uso do serviço de certificados digitais gratuitos do Let's Encrypt instalando o Certbot para nginx. Ao executar o Certbot, este faz algumas perguntas sobre o domínio e pede o endereço de email, depois vai fazer um desafio ao domínio para testar se pertence ao servidor em que executa o bot e finalmente pergunta se queremos usar apenas HTTPS, desativando assim o HTTP, ao qual respondemos que 'sim'. Com isto ficámos com a chave privada do certificado guardada no servidor que é válido por 90 dias.

Também foi necessário adicionar regras de firewall ao servidor e à rede da Azure para permitir o porto 80 e 443 para HTTP e HTTPS respetivamente.

Assim conseguimos ter HTTPS sem avisos de certificados nos browsers em qualquer computador. As configurações foram baseadas no guia da DigitalOcean "Como Proteger o Nginx com o Let's Encrypt no Ubuntu 18.04" ^[3].

Secure FTP com o certificado Let's Encrypt

Ao longo do projeto foi necessário transferir ficheiros para o servidor, fez sentido ter um servidor FTP instalando, o vsftpd, mas como já tínhamos um certificado Let's Encrypt, fazia mais sentido ainda fazer uso dele para implementar Secure FTP por SSL. Tendo o certificado digital, basta adicionar as configurações no vsftp para ativar comunicação SSL, forçar autenticação por SSL e indicar o caminho dos ficheiros do certificado (o certificado em si e a chave privada).

Para aumentar a segurança visto ser um serviço de transferência de ficheiros criámos um utilizador apenas para o FTP que só tem acesso à pasta FTP na sua diretoria Home.

Para ligar usámos o cliente FileZilla que ao conectar (**Servidor: ssproject.team Nome de utilizador e Password “ssprj”**) que da primeira vez deu-nos toda a informação que conhece do certificado para confirmar se pretendo permitir a ligação. Ao aceitar, tenho acesso à pasta `/home/ssprj/FTP` no servidor (como descrito acima).

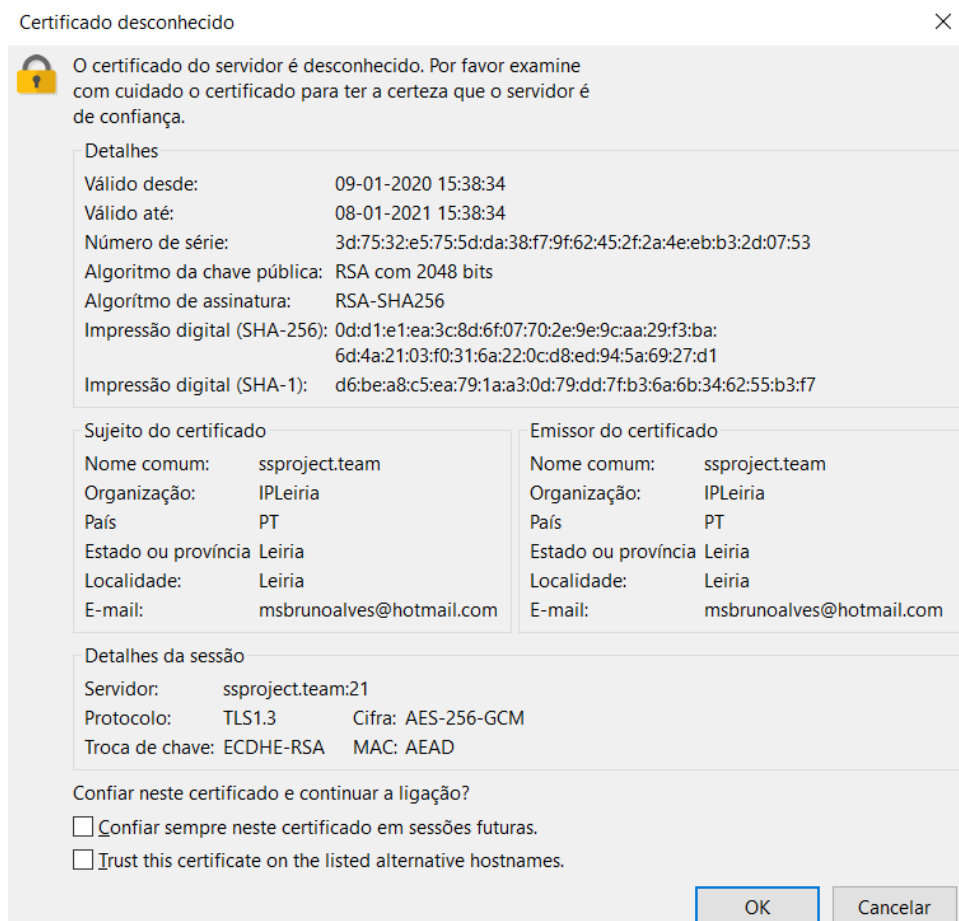


FIGURA 5 - Certificado Digital do servidor FTP

AUTENTICAÇÃO POR TOKENS

Para autenticação por tokens, temos uma implementação de outra UC (Desenvolvimento de Aplicações Distribuídas) que está disponível para experimentar no website do projeto ssproject.team. Na implementação desse trabalho usámos um método de login que recebe as credenciais do utilizador em texto simples e faz uma hash com o Bcrypt, baseado no Blowfish, para guardar na base de dados, o que significa que não guardamos passwords, guardamos apenas a hash que o Bcrypt gera usando a password como seed.

O Bcrypt que o Laravel usa (Framework da API do website) faz 10 rondas de hash por defeito, que era perfeitamente razoável há uns anos, mas de acordo com um tópico no StackExchange sobre a sua eficácia, 10 rondas já não é computacionalmente seguro, em 2014 o recomendado

para o Bcrypt eram 12 rondas, acredito que hoje já sejam mais que isso. Isto cria um problema, porque o Bcrypt, pela natureza do seu algoritmo quase duplica o tempo de execução com cada ronda, a partir do momento que essa execução começa a demorar segundos, torna-se inviável o seu uso para muitos utilizadores.

Por isso o Bcrypt já não é recomendado, num ambiente de produção seria melhor fazer uso do Argon2 que também está disponível no Laravel.

Quando o utilizador faz login, o servidor recebe as credenciais do utilizador, faz a hash dessas credenciais novamente com o Bcrypt e compara essa hash gerada no login com a hash da password guardada na base de dados. Se uma hash for diferente da outra, o login é recusado, mas se forem iguais é gerado um token, esse token é depois enviado para o utilizador e este guarda na memória local do seu browser e passa a enviar esse token nos headers HTTP dos pedidos que faz ao servidor. Assim o utilizador pode aceder às páginas protegidas e reabrir as páginas web sem que tenha de enviar novamente o seu utilizador e password.

```
'bcrypt' => [          Tiago, 2 months ago •
    'rounds' => env('BCRYPT_ROUNDS', 10),
],
```

FIGURA 6 - Configuração por defeito do Bcrypt do Laravel

```
$user = new User();
$user->name=$request->name;
$user->email=$request->email;
$user->type=$request->type;
$user->password=bcrypt($request->password);
$user->photo=$nomeFich;
$user->nif=$request->nif;
$user->save();
```

FIGURA 7 - Uso do Bcrypt no Registo

```

setToken: (state, token) => {
  state.token = token;
  sessionStorage.setItem("token", token);
  axios.defaults.headers.common.Authorization = "Bearer " + token;
},
loadTokenAndUserFromSession: state => {
  state.token = "";
  state.user = null;
  let token = sessionStorage.getItem("token");
  let user = sessionStorage.getItem("user");
  if (token) {
    state.token = token;
    axios.defaults.headers.common.Authorization = "Bearer " + token;
  } else {
    axios.defaults.headers.common.Authorization = undefined;
  }
  if (user) {
    state.user = JSON.parse(user);
  }
},

```

FIGURA 8 - Código javascript do cliente que guarda e usa tokens

Evitar que o utilizador escreva muitas vezes a password numa aplicação é importante, pois mitiga ataques de *keyloggers* que o utilizador possa ter no seu computador sem saber.

Tokens não evitam ataques de man-in-the-middle, se um utilizador malicioso conseguir obter o token de outro, consegue aceder à sua conta. Por isso é importante terminar a sessão em aplicações web com informação sensível. Além disso, não tendo https, o cliente envia as suas credenciais em texto simples, o que significa que é extremamente fácil obter passwords do cliente.

Para mitigar ataques de “*Man in the middle*” aproveitámos o servidor de HTTPS implementado para esta UC e colocámos lá o trabalho da UC de DAD, assim juntamos o melhor das duas UC, autenticação por tokens e comunicação entre o cliente e servidor.

Com HTTPS, apesar de o cliente enviar os dados dos formulários em texto simples, toda a comunicação entre o cliente e o servidor está cifrada, pelo que torna muito mais difícil roubar credenciais ao utilizador.

Para testar isto fizemos uma captura Wireshark ao website sem HTTPS (<http://167.172.49.49/>) e outra ao website com HTTPS (<http://ssproject.team/>), essas capturas vão anexadas ao projeto.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.69	167.172.49.49	HTTP	1337	GET /api/wallets/count HTTP/1.1
2	0.088539	167.172.49.49	192.168.1.69	HTTP	470	HTTP/1.1 200 OK (text/html)
3	0.253163	192.168.1.69	167.172.49.49	TCP	54	63719 → 80 [ACK] Seq=1284 Ack=417 Win=256 Len=0
4	4.940083	192.168.1.69	167.172.49.49	TCP	1426	63719 → 80 [PSH, ACK] Seq=1284 Ack=417 Win=256 Len=0
5	4.940208	192.168.1.69	167.172.49.49	HTTP	97	POST /api/login HTTP/1.1 (application/json)

FIGURA 9 - Captura Wireshark do login sem HTTP (Parte 1)

```

0000 10 13 31 20 31 2c a0 88 69 ea 17 76 08 00 45 00  ..1 1,.. i..v..E.
0010 00 53 d8 23 40 00 80 06 87 b6 c0 a8 01 45 a7 ac  .S.#@... ..E..
0020 31 31 f8 e7 00 50 84 f1 76 ce d9 8d c6 87 50 18  11...P.. v.....P.
0030 01 00 d2 7f 00 00 7b 22 65 6d 61 69 6c 22 3a 22  .....{" email":"
0040 61 64 6d 69 6e 31 40 6d 61 69 6c 2e 70 74 22 2c  admin1@m ail.pt",
0050 22 70 61 73 73 77 6f 72 64 22 3a 22 31 32 33 22  "passwor d":"123"
0060 7d  }

```

FIGURA 10 - Captura WireShark do login sem HTTPS (Parte 2)

1	0.000000	192.168.1.69	52.142.222.172	TLSv1.2	463 Application Data
2	0.076248	52.142.222.172	192.168.1.69	TLSv1.2	395 Application Data
3	0.118224	192.168.1.69	52.142.222.172	TCP	54 64577 → 443 [ACK]
150	1.001923	192.168.1.69	52.142.222.172	TCP	66 64705 → 8080 [SYN]
151	1.253747	192.168.1.69	52.142.222.172	TCP	66 64706 → 8080 [SYN]
156	2.002354	192.168.1.69	52.142.222.172	TCP	66 [TCP Retransmission]
157	2.254493	192.168.1.69	52.142.222.172	TCP	66 [TCP Retransmission]
170	2.992304	192.168.1.69	52.142.222.172	TLSv1.2	1401 Application Data
171	2.998843	192.168.1.69	52.142.222.172	TLSv1.2	1399 Application Data
176	3.057673	52.142.222.172	192.168.1.69	TLSv1.2	331 Application Data
180	3.074151	52.142.222.172	192.168.1.69	TLSv1.2	395 Application Data
184	3.113778	192.168.1.69	52.142.222.172	TCP	54 64215 → 443 [ACK]
185	3.127728	192.168.1.69	52.142.222.172	TCP	54 64222 → 443 [ACK]
198	3.308837	192.168.1.69	52.142.222.172	TCP	66 64698 → 8080 [SYN]
202	3.565557	192.168.1.69	52.142.222.172	TCP	66 64699 → 8080 [SYN]
206	3.996844	192.168.1.69	52.142.222.172	TLSv1.2	465 Application Data

FIGURA 11 - Captura WireShark do login com HTTPS

Como se pode observar nas figuras referentes às capturas do WireShark, quando temos HTTPS, nem é reconhecido como sendo um pedido HTTP, apenas sabemos o protocolo de comunicação que é TLS 1.2 e TCP, mais nada, por mais que procure não aparecem as credenciais do utilizador.

DNS E HTTP SEGURO NA REDE PRIVADA

Servidor Web privado com HTTPS

Para o servidor web privado usámos o nginx e criámos uma página simples.

Para implementar segurança criámos um certificado com o openssl sem este estar assinado por uma CA, ou seja, temos uma ligação cifrada, mas não temos conhecimento de quem emitiu esse certificado e nesse caso o browser pergunta se pretendemos prosseguir.

Para testar usámos uma VM cliente (ubuntu) e testámos no browser tanto com o ip como com o nome (www.om606.ss.pt) e verificar o correto funcionamento do DNS.

Funciona com fail2ban.

Servidor DNS

Temos configurado um domínio de DNS om606.ss.pt que tem várias máquinas, para isto usámos o bind9.

Temos 2 servidores para resolver os nomes desse domínio (master e slave) com transferência de zona entre ambos, para quando adicionarmos uma entrada no domínio esta seja apenas necessária para adicionar num dos servidores.

Para implementar segurança, implementámos Sistema de Nomes e Domínios DNS over TLS (DoT) sobre um servidor a correr nginx (funciona como proxy DoT) que recebe os pedidos de DNS enviados em modo cifrado (TLS), decifra e faz os pedidos de DNS aos servidores. As chaves foram geradas também com o openssl.

Para testar o servidor de DoT criámos uma VM cliente (ubuntu) e configurámos o stunnel para criar a ligação do cliente com o servidor. Para testar usámos o cliente nslookup e verificámos com o wireshark os pacotes enviados pela rede.

Tivemos vários problemas em como configurar o serviço de DNS de forma segura, pensámos em várias alternativas entre elas o DoH (DNS over Https), Stunnel, mas como esta questão do DNS seguro ainda é recente existe pouca informação disponível.

Ao fim de várias tentativas conseguimos por o servidor sem erros e logo depois ocorreu outro problema para testar, procurámos várias formas e chegámos à conclusão que a melhor forma seria o stubby que cria a ligação com o servidor de DNS (que neste caso faz só de proxy).

Num cenário mais real poderíamos ter criado este servidor diretamente nos servidores de DNS uma vez que são estes que resolvem os nomes, mas tivemos problemas com as versões do ubuntu em que estas não suportavam diversos módulos do nginx, por isto criámos um servidor que faz de proxy dos pedidos de DNS.

Funciona com fail2ban.

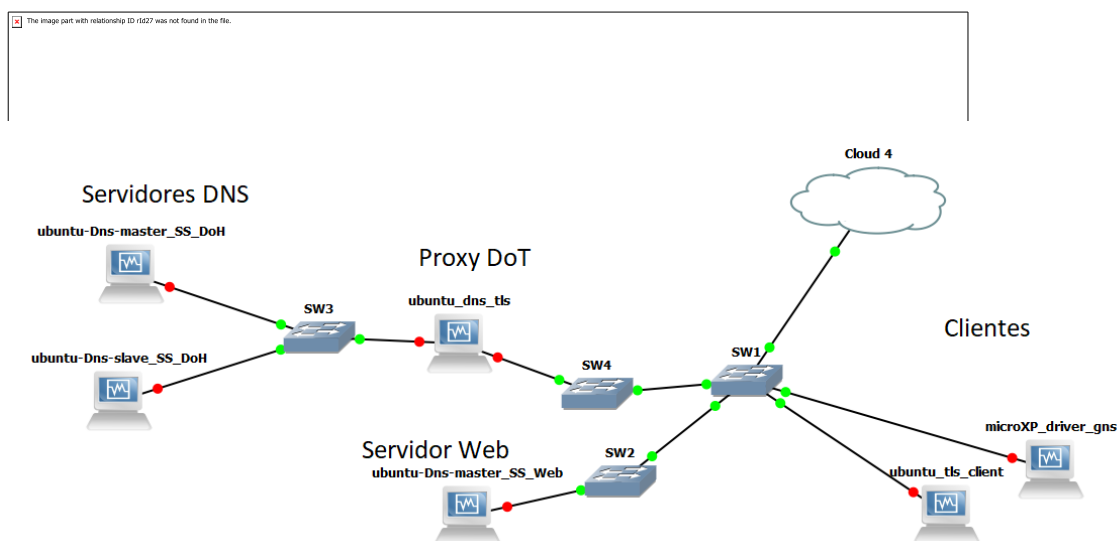


FIGURA 12 – Esquema de Dot e servidor Web interno

DMZ

A (DMZ) foi pensada inicialmente para conter o servidor WEB e DNS para ser acessível através da internet para possíveis clientes, deste modo será visualizado a página institucional, para

isso usámos uma firewall pfsense com três interfaces de rede, uma WAN, uma LAN e uma DMZ de forma a garantir o acesso, através da atualização do NAT, configurou-se o port forwarding para que todos os pedidos cheguem à interface WAN com destino ao porto 53 80 443, DNS, HTTP E HTTPS respetivamente, estes são encaminhados para o respetivo servidor, estando este da zona desmilitarizada (DMZ).

Nas imagens que seguem, encontra-se representadas as regras configuradas nas interfaces para garantir este pressuposto.

Firewall / NAT / Port Forward

Port Forward1:1OutboundNPT

Rules										
	Interface	Protocol	Source Address	Source Ports	Dest. Address	Dest. Ports	NAT IP	NAT Ports	Description	Actions
<input type="checkbox"/>	<input checked="" type="checkbox"/> WAN	TCP/UDP	*	*	WAN address	80 (HTTP)	192.168.100.10	80 (HTTP)	Servidor Web da DMZ	
<input type="checkbox"/>	<input checked="" type="checkbox"/> WAN	TCP/UDP	*	*	WAN address	443 (HTTPS)	192.168.100.10	443 (HTTPS)		
<input type="checkbox"/>	<input checked="" type="checkbox"/> WAN	TCP/UDP	*	*	WAN address	53 (DNS)	192.168.100.10	53 (DNS)		

Add Add Delete Save Separator

FIGURA 13 – Regras da Port Forward

Firewall / Rules / WAN

FloatingWANLANDMZIPsecOpenVPN

Rules (Drag to Change Order)										
	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description
<input type="checkbox"/>	0 / 10 KiB	*	Reserved Not assigned by IANA	*	*	*	*	*	*	Block bogus networks
<input type="checkbox"/>	0 / 322 KiB	IPv4 TCP/UDP	*	*	192.168.100.10	80 (HTTP)	*	none		NAT Servidor Web da DMZ
<input type="checkbox"/>	0 / 0 B	IPv4 TCP/UDP	*	*	192.168.100.10	443 (HTTPS)	*	none		NAT
<input type="checkbox"/>	0 / 0 B	IPv4 TCP/UDP	*	*	192.168.100.10	53 (DNS)	*	none		NAT

FIGURA 14 – Regras da interface WAN

Firewall / Rules / LAN

FloatingWANLANDMZIPsecOpenVPN

Rules (Drag to Change Order)										
	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description
<input type="checkbox"/>	0 / 10.66 MiB	*	*	*	LAN Address	443 80	*	*		Anti-Lockout Rule
<input type="checkbox"/>	0 / 2 KiB	IPv4 ICMP any	LAN net	*	*	*	*	none		
<input type="checkbox"/>	0 / 0 B	IPv4 TCP/UDP	LAN net	*	*	53 (DNS)	*	none		
<input type="checkbox"/>	0 / 480 KiB	IPv4 TCP/UDP	LAN net	*	*	80 (HTTP)	*	none		
<input type="checkbox"/>	0 / 18.48 MiB	IPv4 TCP/UDP	LAN net	*	*	443 (HTTPS)	*	none		

















Add Add Delete Save Separator

FIGURA 15 – Regras da interface LAN

Firewall / Rules / DMZ

Floating WAN LAN DMZ IPsec OpenVPN

Rules (Drag to Change Order)

<input type="checkbox"/>	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input type="checkbox"/>	✓ 0 / 672 B	IPv4 ICMP any	DMZ net	*	*	*	*	none			   
<input type="checkbox"/>	✓ 0 / 0 B	IPv4 TCP/UDP	DMZ net	*	! LAN net	53 (DNS)	*	none			   
<input type="checkbox"/>	✓ 0 / 13 KiB	IPv4 TCP/UDP	DMZ net	*	! LAN net	80 (HTTP)	*	none			   
<input type="checkbox"/>	✓ 1 / 2.54 MiB	IPv4 TCP/UDP	DMZ net	*	! LAN net	443 (HTTPS)	*	none			   

↑ Add

↓ Add

🗑 Delete

💾 Save

➕ Separator

FIGURA 16 – Regras da DMZ

VPN

OpenVPN

Houve a necessidade de criar esta VPN para permitir aos utilizadores que se encontram na internet aceder à rede local, esta encontra-se totalmente funcional.

Nas imagens que seguem, encontra-se representadas as regras configuradas nas interfaces para garantir este pressuposto.

VPN / OpenVPN / Servers					
Servers Clients Client Specific Overrides Wizards Client Export Shared Key Export					
OpenVPN Servers					
Interface	Protocol / Port	Tunnel Network	Crypto	Description	Actions
WAN	TCP4 / 1195	10.10.10.0/24	Crypto: AES-128-CBC/SHA256 D-H Params: 2048 bits	VPN - Utilizadores Externos (tun)	✎ 🗑

FIGURA 17 – Algoritmos OpenVPN

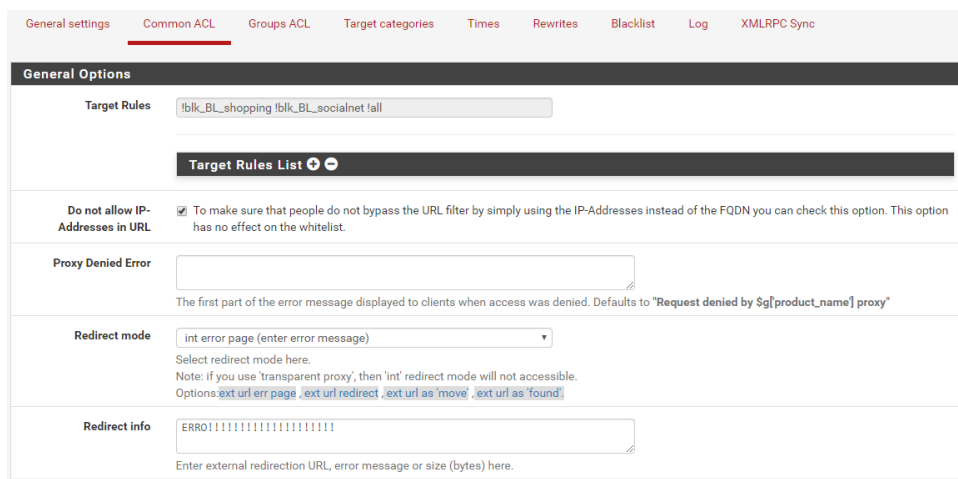
System / Certificate Manager / Certificates				
CAs Certificates Certificate Revocation				
Certificates				
Name	Issuer	Distinguished Name	In Use	Actions
webConfigurator default (5e18a55a6e906) Server Certificate CA: No Server: Yes	self-signed	O=pfSense webConfigurator Self-Signed Certificate, CN=pfSense-5e18a55a6e906 Valid From: Fri, 10 Jan 2020 16:24:58 +0000 Valid Until: Wed, 02 Jul 2025 16:24:58 +0000	webConfigurator	✎ 🗑
OpenVPN Server Certificate CA: No Server: Yes	Internal PfSense CA	ST=Leiria, O=IPLeia, L=Leiria, CN=OpenVPN, C=PT Valid From: Sat, 11 Jan 2020 17:52:47 +0000 Valid Until: Tue, 08 Jan 2030 17:52:47 +0000	OpenVPN Server	✎ 🗑
tiago User Certificate CA: No Server: No	Internal PfSense CA	CN=Segurança Sisteamas Valid From: Sat, 11 Jan 2020 18:14:52 +0000 Valid Until: Tue, 08 Jan 2030 18:14:52 +0000	User Cert	✎ 🗑

FIGURA 18 – Certificado OpenVPN utilizador

PROXY

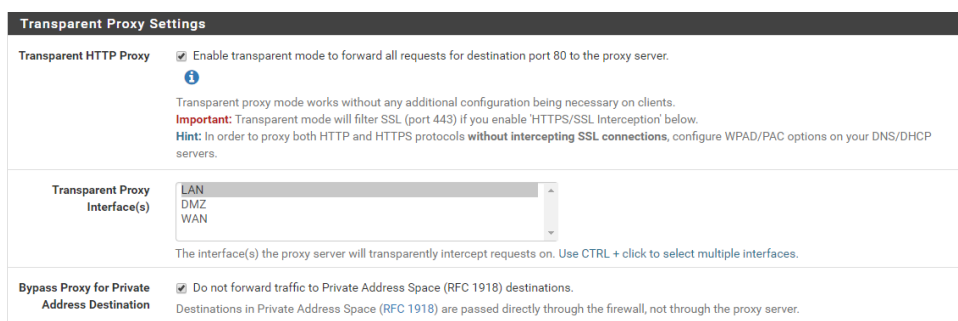
SquidGuard encontra-se incorporado na pfsense, está configurado de forma a ser um proxy transparente para os utilizadores da rede LAN.

O proxy não funciona totalmente.



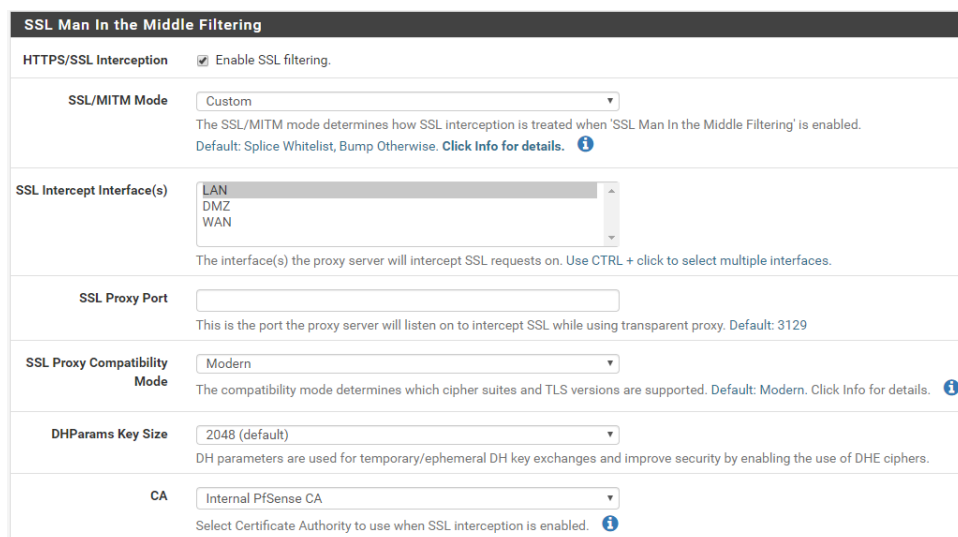
The screenshot shows the 'General Options' tab of the SquidGuard configuration. The 'Target Rules' field contains '!blk_BL_shopping !blk_BL_socialnet !all'. Below it is a 'Target Rules List' button. The 'Do not allow IP-Addresses in URL' checkbox is checked, with a note: 'To make sure that people do not bypass the URL filter by simply using the IP-Addresses instead of the FQDN you can check this option. This option has no effect on the whitelist.' The 'Proxy Denied Error' field is empty, with a note: 'The first part of the error message displayed to clients when access was denied. Defaults to "Request denied by \$g[product_name] proxy"'. The 'Redirect mode' dropdown is set to 'int error page (enter error message)', with a note: 'Select redirect mode here. Note: if you use "transparent proxy", then "int" redirect mode will not be accessible. Options: ext url err page, ext url redirect, ext url as move, ext url as found'. The 'Redirect info' field contains 'ERRO!!!!!!!!!!!!!!!!!!!!', with a note: 'Enter external redirection URL, error message or size (bytes) here.'

FIGURA 19 –SquidGuard



The screenshot shows the 'Transparent Proxy Settings' section. The 'Transparent HTTP Proxy' checkbox is checked, with a note: 'Enable transparent mode to forward all requests for destination port 80 to the proxy server.' Below it is an information icon and a note: 'Transparent proxy mode works without any additional configuration being necessary on clients. Important: Transparent mode will filter SSL (port 443) if you enable "HTTPS/SSL Interception" below. Hint: In order to proxy both HTTP and HTTPS protocols without intercepting SSL connections, configure WPAD/PAC options on your DNS/DHCP servers.' The 'Transparent Proxy Interface(s)' dropdown is set to 'LAN', with a note: 'The interface(s) the proxy server will transparently intercept requests on. Use CTRL + click to select multiple interfaces.' The 'Bypass Proxy for Private Address Destination' checkbox is checked, with a note: 'Do not forward traffic to Private Address Space (RFC 1918) destinations. Destinations in Private Address Space (RFC 1918) are passed directly through the firewall, not through the proxy server.'

FIGURA 20 – Proxy transparente



The screenshot shows the 'SSL Man in the Middle Filtering' section. The 'HTTPS/SSL Interception' checkbox is checked, with a note: 'Enable SSL filtering.' The 'SSL/MITM Mode' dropdown is set to 'Custom', with a note: 'The SSL/MITM mode determines how SSL interception is treated when "SSL Man in the Middle Filtering" is enabled. Default: Splice Whitelist, Bump Otherwise. Click Info for details.' The 'SSL Intercept Interface(s)' dropdown is set to 'LAN', with a note: 'The interface(s) the proxy server will intercept SSL requests on. Use CTRL + click to select multiple interfaces.' The 'SSL Proxy Port' field is empty, with a note: 'This is the port the proxy server will listen on to intercept SSL while using transparent proxy. Default: 3129'. The 'SSL Proxy Compatibility Mode' dropdown is set to 'Modern', with a note: 'The compatibility mode determines which cipher suites and TLS versions are supported. Default: Modern. Click Info for details.' The 'DHParams Key Size' dropdown is set to '2048 (default)', with a note: 'DH parameters are used for temporary/ephemeral DH key exchanges and improve security by enabling the use of DHE ciphers.' The 'CA' dropdown is set to 'Internal PfSense CA', with a note: 'Select Certificate Authority to use when SSL interception is enabled.'

FIGURA 21 – Certificado do proxy

SSH COM FAIL2BAN

Os servidores com acesso remoto por SSH foram configurados com o serviço fail2ban.

Para a configuração deste serviço foi necessário a criação de um ficheiro `/etc/fail2ban/jail.local` onde estão as regras que o fail2ban irá seguir. Neste ficheiro foi definido:

- “maxretry” = 5 - Número máximo de tentativas de autenticação por SSH.
- “bantime” = 10m - minutos - O tempo que um determinado IP ou hostname irá ter que esperar para poder tentar novamente uma conexão ao servidor.
- “findtime” = 10m - minutos - Se um utilizador atinge o número máximo de tentativas neste intervalo de tempo, então irá ser banido.

Quando um utilizador se tenta conectar ao servidor e é banido, esta informação fica disponível em `/var/logs/fail2ban.log`.

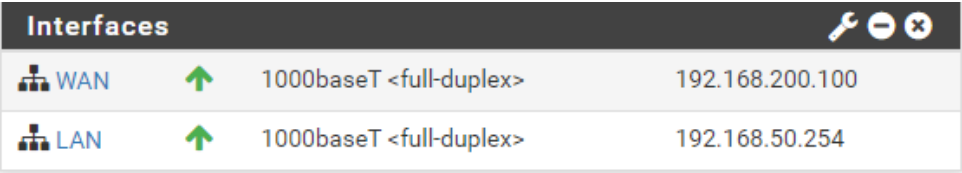
PFSENSE (REDE RESTRITA DENTRO DA REDE DA SEDE)

Foi criada uma firewall pfsense dentro da rede da sede. A criação desta pfsense deveu-se à necessidade de ter uma rede isolada, com acesso restrito, onde se encontra um servidor de

NAS e uma máquina vulnerável (mestasploitable fornecida pelo professor), que usamos para testar a nossa firewall com um sistema Nessus (descrito mais em baixo).

Interfaces

A pfsense tem duas interfaces de rede, uma para a rede da sede (WAN – 192.168.200.0/24) e outra para a rede local restrita (LAN – 192.168.50.0/24).



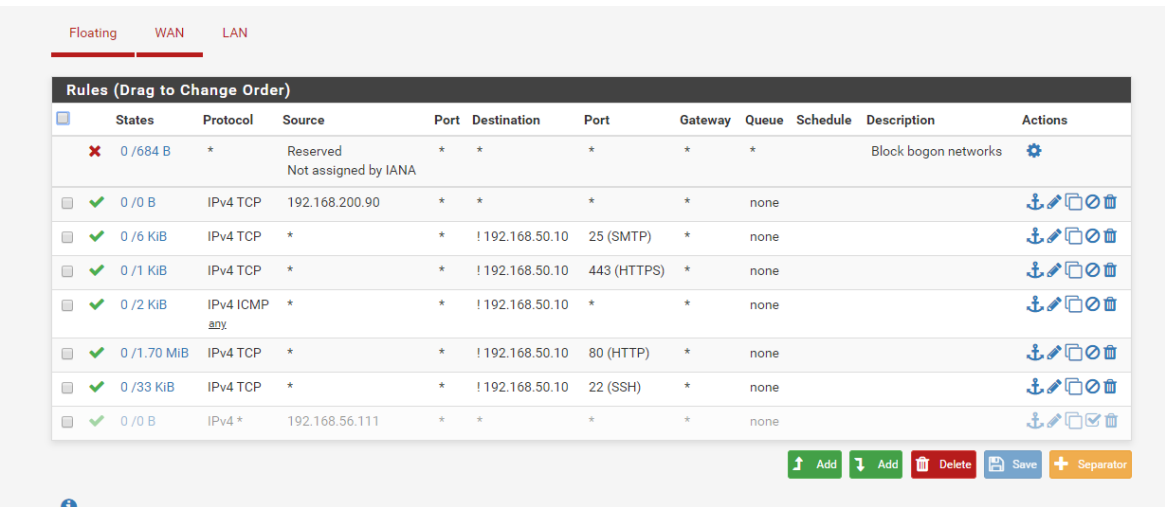
Interfaces			
WAN	↑	1000baseT <full-duplex>	192.168.200.100
LAN	↑	1000baseT <full-duplex>	192.168.50.254

FIGURA 22 – Interfaces de rede da pfsense

Regras da firewall

Nas regras da firewall, para a interface WAN, foi definido que apenas o tráfego vindo da máquina ADMIN poderia passar completamente até à rede LAN.

A firewall deixa também passar tráfego HTTP, HTTPS, SSH, ICMP e SMTP da rede WAN para a rede LAN à exceção do servidor NAS que apenas é acessível pela máquina ADMIN (192.168.200.90). Estas regras foram definidas para que pudéssemos testar o Nessus e para que o servidor de NAS ficasse com acesso exclusivo apenas às máquinas autorizadas.



Rules (Drag to Change Order)											
	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✗	0 /684 B	*	Reserved Not assigned by IANA	*	*	*	*	*		Block bogon networks	⚙️
☐	✓ 0 /0 B	IPv4 TCP	192.168.200.90	*	*	*	*	none			📌 ⚙️ 🗑️
☐	✓ 0 /6 KiB	IPv4 TCP	*	*	! 192.168.50.10	25 (SMTP)	*	none			📌 ⚙️ 🗑️
☐	✓ 0 /1 KiB	IPv4 TCP	*	*	! 192.168.50.10	443 (HTTPS)	*	none			📌 ⚙️ 🗑️
☐	✓ 0 /2 KiB	IPv4 ICMP	*	*	! 192.168.50.10	*	*	none			📌 ⚙️ 🗑️
☐	✓ 0 /1.70 MiB	IPv4 TCP	*	*	! 192.168.50.10	80 (HTTP)	*	none			📌 ⚙️ 🗑️
☐	✓ 0 /33 KiB	IPv4 TCP	*	*	! 192.168.50.10	22 (SSH)	*	none			📌 ⚙️ 🗑️
☐	✓ 0 /0 B	IPv4 *	192.168.56.111	*	*	*	*	none			📌 ⚙️ 🗑️

FIGURA 23 – Regras da firewall para a interface WAN

Na rede LAN foi definida uma política por omissão que permite que passe todo o tráfego até à interface WAN.

Rules (Drag to Change Order)											
	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	6 / 1.80 MiB	*	*	*	LAN Address	80	*	*		Anti-Lockout Rule	
<input checked="" type="checkbox"/>	0 / 311 KiB	IPv4 *	*	*	*	*	*	none			

Add
 Add
 Delete
 Save
 Separator

FIGURA 24 – Regras da firewall para a interface LAN

IDS (Intrusion Detection System) - SNORT

Configurámos um sistema de deteção de intrusão, o snort. Um sistema de deteção de intrusão que permite “detetar” eventuais ataques feitos a uma rede, quer sejam scans ou até análises feitas por ferramentas de deteção de vulnerabilidades como o Nessus. Quando estes ataques são detetados, é gerado um alerta. Depois cabe ao administrador da rede decidir o que fazer com a informação do alerta. No nosso caso ficámos pelos alertas e não configurámos o sistema de prevenção que permite fazer bloqueios a listas de possíveis atacantes.

Para a sua configuração foi necessária a instalação do Package do snort na pfsense.

Foram criadas duas interfaces onde o snort iria atuar, uma para a WAN e outra para a LAN.

Interface Settings Overview						
Interface	Snort Status	Pattern Match	Blocking	Barnyard2 Status	Description	Actions
<input checked="" type="checkbox"/> WAN (em0)		AC-BNFA	DISABLED	DISABLED	WAN	
<input checked="" type="checkbox"/> LAN (em1)		AC-BNFA	DISABLED	DISABLED	LAN	

Delete

FIGURA 25 – Imagem com as duas interfaces configuradas no snort

Para configurar cada uma das interfaces foi necessário indicar onde queríamos registar os logs dos alertas criados pelo snort.

The screenshot displays the Snort configuration web interface. At the top, there are tabs for 'LAN Settings', 'LAN Categories', 'LAN Rules', 'LAN Variables', 'LAN Preprocs', and 'LAN Barnyard2'. The 'LAN Settings' tab is active, showing two sections: 'General Settings' and 'Alert Settings'.

General Settings:

- Enable:** A checkbox labeled 'Enable interface' is checked.
- Interface:** A dropdown menu shows 'LAN (em1)' selected. Below it, text reads: 'Choose the interface where this Snort instance will inspect traffic.'
- Description:** A text input field contains 'LAN'. Below it, text reads: 'Enter a meaningful description here for your reference.'
- Snap Length:** A text input field contains '1518'. Below it, text reads: 'Enter the desired interface snaplen value in bytes. Default is 1518 and is suitable for most a'.

Alert Settings:

- Send Alerts to System Log:** A checkbox is checked. Text below reads: 'Snort will send Alerts to the firewall's system log. Default is Not Checked.'
- System Log Facility:** A dropdown menu shows 'LOG_AUTH' selected. Below it, text reads: 'Select system log Facility to use for reporting. Default is LOG_AUTH.'
- System Log Priority:** A dropdown menu shows 'LOG_ALERT' selected. Below it, text reads: 'Select system log Priority (Level) to use for reporting. Default is LOG_ALERT.'
- Block Offenders:** A checkbox is unchecked. Text below reads: 'Checking this option will automatically block hosts that generate a Snort alert'.

FIGURA 26 – Configurar interface de snort

É possível dizer ao snort que pretendemos que este detete scans feitos à rede.

The screenshot displays the 'Portscan Detection' configuration section in the Snort web interface. It includes the following settings:

- Enable:** A checkbox labeled 'Use Portscan Detection to detect various types of port scans and sweeps. Default is Not Checked.' is checked.
- Protocol:** A dropdown menu shows 'all' selected. Below it, text reads: 'Choose the Portscan protocol type to alert for (all, tcp, udp, icmp or ip). The default is all.'
- Scan Type:** A dropdown menu shows 'all' selected. Below it, text reads: 'Choose the Portscan scan type to alert for. The default is all.'

Below the Scan Type dropdown, there is a detailed explanation of the scan types:

- PORTSCAN: one->one scan; one host scans multiple ports on another host.
- PORTSWEEP: one->many scan; one host scans a single port on multiple hosts.
- DECOY_PORTSCAN: one->one scan; attacker has spoofed source address inter-mixed with real scanning address.
- DISTRIBUTED_PORTSCAN: many->one scan; multiple hosts query one host for open services.
- ALL: alerts for all of the above scan types.

FIGURA 27 – Configurar o snort para detetar scan a portos nas máquinas da rede

Também é possível definir as listas de regras que pretendemos subscrever, ou seja, o tipo de “ataques” a que o snort irá responder (detetar). No nosso caso configurámos os seguintes:

- DOS
- Malware
- Scan
- Telnet
- Trojan

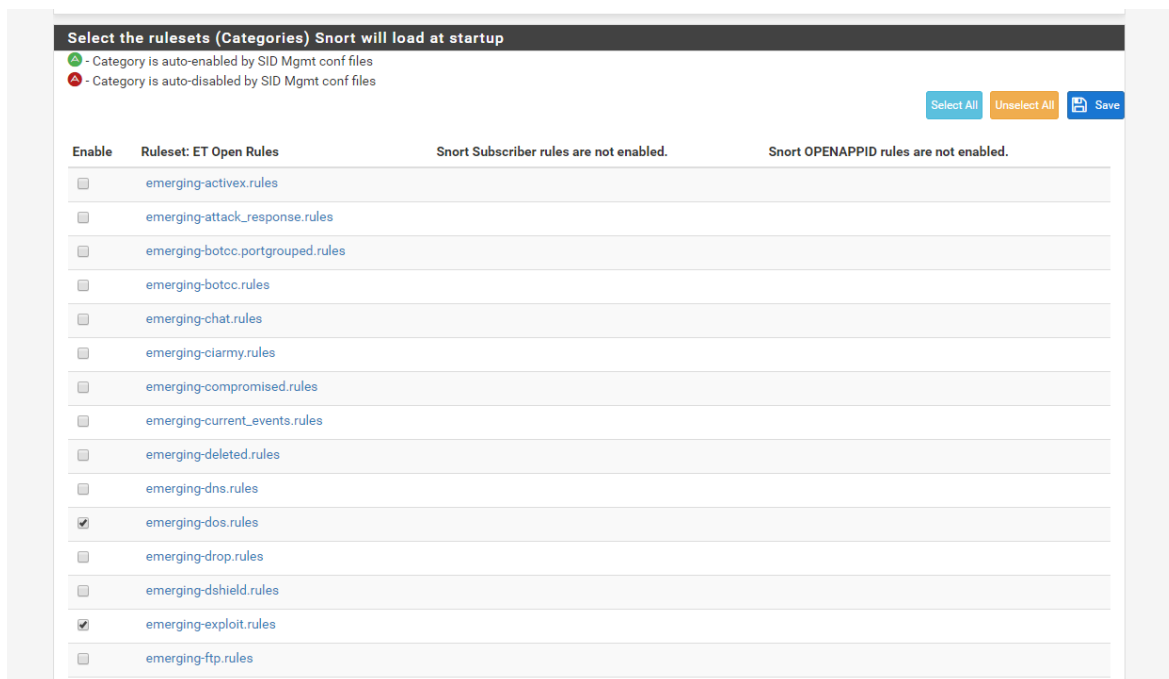


FIGURA 28 – Algumas das listas sobre as regras a serem interpretadas pelo snort.

SERVIDOR NAS (FREENAS)

Para criação do servidor NAS decidimos optar pela solução FreeNAS. Este servidor foi criado dentro da rede protegida pela pfsense descrita acima.

Foi criada uma máquina virtual com o sistema operativo FreeNAS com uma interface de rede e com 3 discos de armazenamento, sendo um deles para o sistema operativo e os outros dois para o armazenamento de ficheiros.

Após a criação da máquina foi criada uma *Pool* à qual demos o nome de RAID. Esta *Pool* consiste em 2 discos com RAID 1 (mirror). Será nesta *Pool* que os utilizadores irão colocar os seus ficheiros. Depois de criada a *Pool* RAID foi necessário configurar uma partilha NFS da mesma. Nesta partilha limitámos o acesso apenas a um host, neste caso, o servidor ADMIN (IP:192.168.200.90).

RAID ✓ HEALTHY: 12.46 MiB (0%) Used / 7.25 GiB Free								
Name ↕	Type ↕	Used ↕	Available ↕	Compression ↕	Compression Ratio ↕	Readonly ↕	Dedup ↕	Comments ↕
> RAID	dataset	12.46 MiB	7.25 GiB	lz4	9.24x	false	off	

FIGURA 29– Pool “RAID”

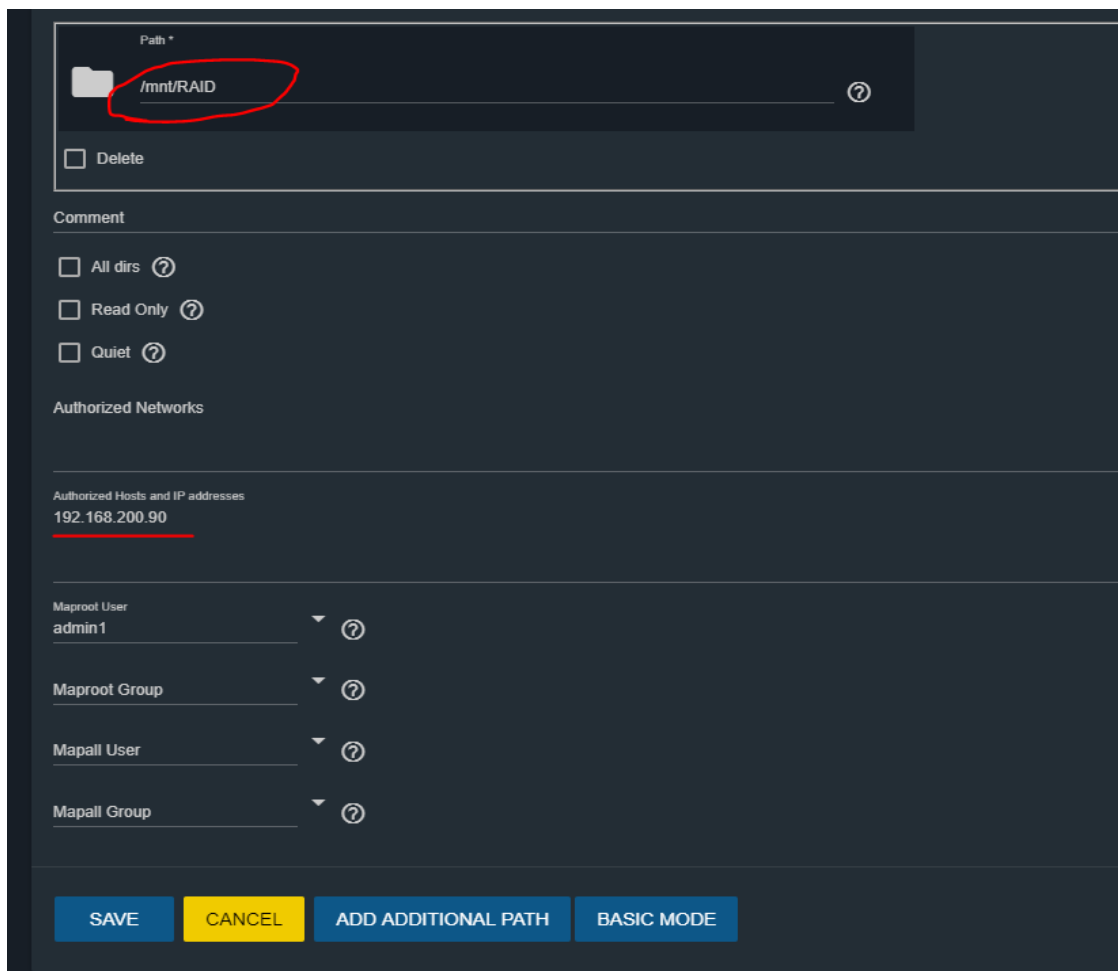


FIGURA 30 – Configuração da partilha NFS

SERVIDOR ADMIN (PARA ACESSO A FREENAS)

O servidor ADMIN encontra-se na rede da sede.

Este é o único servidor que tem acesso ao servidor NAS descrito em cima. Neste servidor é possível fazer mount à diretoria remota /mnt/RAID, configurada no servidor NAS. Para este mount foi adicionada a linha “192.168.50.10:/mnt/RAID /home/ubuntu/WORK nfs defaults 0 0” no ficheiro /etc/fstab da máquina ADMIN (user:ubuntu), posteriormente é apenas necessário fazer “mount ~/WORK/” e está feita a ligação entre a pasta WORK da máquina ADMIN e a pasta /mnt/RAID do servidor NAS.

ANÁLISE DE VULNERABILIDADES

Análise de vulnerabilidades com o Nessus

Sabemos que a segurança de uma rede empresarial é extremamente importante, pequenas falhas na rede podem-se tornar em milhões de euros perdidos e pode até mesmo acabar com empresas. Mas como podemos ter a certeza que a rede da nossa empresa é o mais segura possível? Para isso existem ferramentas de software como o Nmap e o Nessus.

O Nessus, além de um port scanner (como o Nmap), é um scanner de vulnerabilidades. É um Software de código fechado e muito caro para pesquisa de vulnerabilidades. É um scan bastante intrusivo e violento no que diz respeito à quantidade de pesquisas e tráfego gerado e o seu custo afasta utilizadores comuns, mas a informação que este gera pode ter muito mais valor que esse custo para uma empresa.

Utilizámos para este scan a versão de teste, o NESSUS Essentials, é limitado e oferece apenas 16 IPs de teste, mas para o nosso caso de estudo é mais do que suficiente.



FIGURA 31 – Versões do nessus

Tendo o servidor do Nessus instalado num computador local podemos começar a efetuar scans, é uma instalação demorada e tende a falhar pela primeira vez, mas eventualmente funciona corretamente.

Para ter um caso de estudo interessante e que nos permita ter uma conclusão certa, decidimos fazer uso da máquina virtual “Metasploitable” que nos foi disponibilizado para a UC.

Sendo isto uma versão de teste e um software que se pode tornar extremamente complexo e poderoso nas mãos certas (ou erradas), limitámos a nossa pesquisa ao template “Basic Network Scan”, que apesar de “Basic” dá-nos muita informação para digerir sobre a nossa rede. É uma pesquisa que demora alguns minutos a concluir numa máquina local e demora mais ainda em máquinas remotas (devido ao ping).

Após a conclusão do scan, o Nessus, encontrou uma lista de 113 vulnerabilidades distribuídas da seguinte forma:

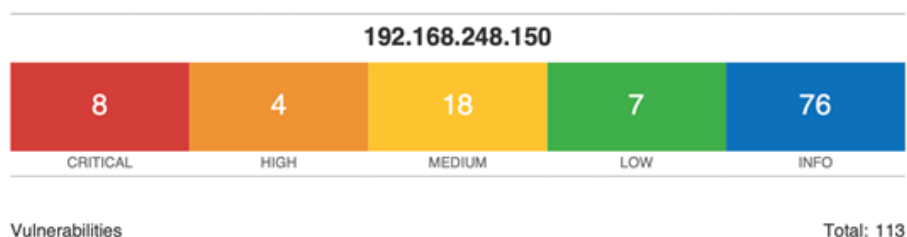


FIGURA 32 - Vulnerabilidades do Metasploitable

Um número que imediatamente é alarmante, é de facto dos piores casos possíveis sendo esta uma máquina feita para ter vulnerabilidades. As vulnerabilidades que o NESSUS considerou como críticas relevam-se extremamente críticas para um ambiente empresarial, ou para um ambiente que contenha informação sensível. Tendo inclusive encontrado problemas como o servidor VNC ter a “password”, é de facto crítico.

Vulnerabilities				Total: 113
SEVERITY	CVSS	PLUGIN	NAME	
CRITICAL	10.0	51988	Bind Shell Backdoor Detection	
CRITICAL	10.0	32314	Debian OpenSSH/OpenSSL Package Random Number Generator Weakness	
CRITICAL	10.0	32321	Debian OpenSSH/OpenSSL Package Random Number Generator Weakness (SSL check)	
CRITICAL	10.0	11356	NFS Exported Share Information Disclosure	
CRITICAL	10.0	33850	Unix Operating System Unsupported Version Detection	
CRITICAL	10.0	46882	UnrealIRCd Backdoor Detection	
CRITICAL	10.0	61708	VNC Server 'password' Password	
CRITICAL	10.0	10203	rexecd Service Detection	

FIGURA 33 - Vulnerabilidades críticas do Metasploitable

O NESSUS, além de nos alertar para as vulnerabilidades fornece-nos um painel de leitura com mais informação, apresenta sugestões para a sua correção. Decidimos então começar por algumas vulnerabilidades detetadas.

Bind Shell Backdoor Detection

CRITICAL Nessus Plugin ID 51988

Synopsis

The remote host may have been compromised.

Description

A shell is listening on the remote port without any authentication being required. An attacker may use it by connecting to the remote port and sending commands directly.

Solution

Verify if the remote host has been compromised, and reinstall the system if necessary.

FIGURA 34- Exemplos de mais informação sobre as vulnerabilidades

NFS Exported Share Information Disclosure

CRITICAL Nessus Plugin ID 11356

Synopsis

It is possible to access NFS shares on the remote host.

Description

At least one of the NFS shares exported by the remote server could be mounted by the scanning host. An attacker may be able to leverage this to read (and possibly write) files on remote host.

Solution

Configure NFS on the remote host so that only authorized hosts can mount its remote shares.

FIGURA 35- Exemplos de mais informações sobre vulnerabilidades.

Os backdoors sem sombra de dúvida que é seguro afirmar que são as maiores dores de cabeça para um administrador do sistema, uma vez que estes invalidam todas as barreiras de segurança implementadas. Conforme o Nessus reporta uma solução, deveremos tentar perceber de imediato se o sistema já se tornou vulnerável e reinstalar com máxima urgência todo o sistema.

O NESSUS retornou como vulnerabilidades altas, as seguintes.

HIGH	7.5	34460	Unsupported Web Server Detection
HIGH	7.5	10205	rlogin Service Detection
HIGH	7.5	10245	rsh Service Detection
HIGH	7.1	20007	SSL Version 2 and 3 Protocol Detection

FIGURA 36- Vulnerabilidades Altas detetadas

Protocolos de comunicação que não devem ser utilizados numa organização. Os portos devem ser fechados, de forma a impedir este tipo de comunicação. Além disso, as credencias de SSH devem ser melhoradas.

Como vulnerabilidades médias, o NESSUS retornou as seguintes:

MEDIUM	6.8	90509	Samba Badlock Vulnerability
MEDIUM	6.4	51192	SSL Certificate Cannot Be Trusted
MEDIUM	6.4	57582	SSL Self-Signed Certificate
MEDIUM	5.8	42263	Unencrypted Telnet Server
MEDIUM	5.0	12085	Apache Tomcat Default Files
MEDIUM	5.0	12217	DNS Server Cache Snooping Remote Information Disclosure
MEDIUM	5.0	11213	HTTP TRACE / TRACK Methods Allowed

FIGURA 37- Vulnerabilidades médias detetadas.

MEDIUM	5.0	42256	NFS Shares World Readable
MEDIUM	5.0	57608	SMB Signing not required
MEDIUM	5.0	15901	SSL Certificate Expiry
MEDIUM	5.0	45411	SSL Certificate with Wrong Hostname
MEDIUM	5.0	42873	SSL Medium Strength Cipher Suites Supported (SWEET32)
MEDIUM	4.3	90317	SSH Weak Algorithms Supported
MEDIUM	4.3	89058	SSL DROWN Attack Vulnerability (Decrypting RSA with Obsolete and Weakened eNcryption)
MEDIUM	4.3	26928	SSL Weak Cipher Suites Supported
MEDIUM	4.3	81606	SSL/TLS EXPORT_RSA <= 512-bit Cipher Suites Supported (FREAK)
MEDIUM	4.3	78479	SSLv3 Padding Oracle On Downgraded Legacy Encryption Vulnerability (POODLE)
MEDIUM	4.0	52611	SMTP Service STARTTLS Plaintext Command Injection

FIGURA 38- Mais vulnerabilidades médias detetadas.

Pelos conhecimentos que temos, conseguimos imediatamente perceber que uma grande parte delas, basta impedirmos comunicação em determinados portos e deixa de existir a vulnerabilidade, ainda assim, analisámos umas quantas para percebermos que tipo de vulnerabilidades se tratavam. O caso da SMB, que é por exemplo já conhecido por nós como vulnerabilidade permitindo a comunicação nos portos 445 e 139.

Além disso, como vulnerabilidade baixa, recolhemos diversa informação no que diz respeito ao protocolo de SSH, e sobre as chaves e encriptação.

LOW	2.6	70658	SSH Server CBC Mode Ciphers Enabled
LOW	2.6	71049	SSH Weak MAC Algorithms Enabled
LOW	2.6	31705	SSL Anonymous Cipher Suites Supported
LOW	2.6	65821	SSL RC4 Cipher Suites Supported (Bar Mitzvah)
LOW	2.6	83875	SSL/TLS Diffie-Hellman Modulus <= 1024 Bits (Logjam)
LOW	2.6	83738	SSL/TLS EXPORT_DHE <= 512-bit Export Cipher Suites Supported (Logjam)
LOW	2.6	10407	X Server Detection

FIGURA 39 - Vulnerabilidades baixas detetadas.

O NESSUS recolheu ainda uma quantidade de informação, que disponibilizámos como anexo. Mas bastou esta informação para perceber que se tratava de um sistema imensamente vulnerável, conseguimos estudar os ataques possíveis, os portos que deveríamos atacar.

Após testarmos as regras, vimos que as configurações estão certas com o que pretendíamos, voltamos então a fazer um scan utilizando o NESSUS, nas mesmas condições. A máquina passou a estar protegida por uma das firewalls implementadas no projeto.

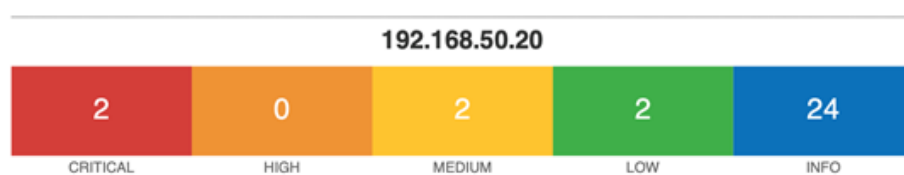


FIGURA 40 - Gravidade das vulnerabilidades detetadas ATRÁS da firewall.

Após o scan ser completado, passámos de 113 vulnerabilidades para 30 vulnerabilidade.

Podemos imediatamente concluir que se trata de uma redução ENORME, é seguro dizer que a máquina passou a estar mais segura, mas vamos então perceber quais as vulnerabilidades que o Nessus considerou.

Vulnerabilities				Total: 30
SEVERITY	CVSS	PLUGIN	NAME	
CRITICAL	10.0	32314	Debian OpenSSH/OpenSSL Package Random Number Generator Weakness	
CRITICAL	10.0	33850	Unix Operating System Unsupported Version Detection	
MEDIUM	5.0	11213	HTTP TRACE / TRACK Methods Allowed	
MEDIUM	4.3	90317	SSH Weak Algorithms Supported	
LOW	2.6	70658	SSH Server CBC Mode Ciphers Enabled	
LOW	2.6	71049	SSH Weak MAC Algorithms Enabled	

FIGURA 41 - Exemplos de vulnerabilidades detetadas.

Foram 6 as vulnerabilidades reconhecidas e as restantes informações que voltámos a disponibilizar como anexo.

Podemos ver que nenhuma delas é uma vulnerabilidade real. O SSH, é facilmente protegido com regras de permissão apenas para um determinado IP, para o administrador da rede passámos a preocupar-nos com 3 vulnerabilidades.

Uma delas diz respeito à falta de conhecimento do NISSUS no que diz respeito à versão de UNIX que estamos a usar. Logo, não é um perigo de segurança.

Passamos então a analisar uma delas

Debian OpenSSH/OpenSSL Package Random Number Generator Weakness

CRITICAL Nessus Plugin ID 32314

Synopsis

The remote SSH host keys are weak.

Description

The remote SSH host key has been generated on a Debian or Ubuntu system which contains a bug in the random number generator of its OpenSSL library.

The problem is due to a Debian packager removing nearly all sources of entropy in the remote version of OpenSSL.

An attacker can easily obtain the private part of the remote key and use this to set up to decipher the remote session or set up a man in the middle attack.

Solution

Consider all cryptographic material generated on the remote host to be guessable. In particular, all SSH, SSL and OpenVPN key material should be re-generated.

See Also

Plugin Details

Severity: Critical

ID: 32314

File Name: ssh_debian_weak.nasl

Version: 1.20

Type: remote

Family: Gain a shell remotely

Published: 2008/05/14

Updated: 2018/11/15

Dependencies: 10267

Risk Information

Risk Factor: Critical

FIGURA 42 - Exemplo de mais informação da vulnerabilidade do OpenSSH.

Após percebermos que tipo de vulnerabilidade estava retratado, percebemos que se tratava de uma vulnerabilidade gerada na criação de chaves remotas. Facilmente impedimos a partilha de chaves, permitindo apenas o ssh ao administrador do sistema, é uma vulnerabilidade que deixa de existir.

Esta comparação do Nessus entre o mesmo sistema passando ou não entra uma firewall, permite-nos sem dúvida reconhecer a importância das firewalls e ainda assim, sabemos que a configuração de uma firewall poderá ser gráfica e simples (como no caso da

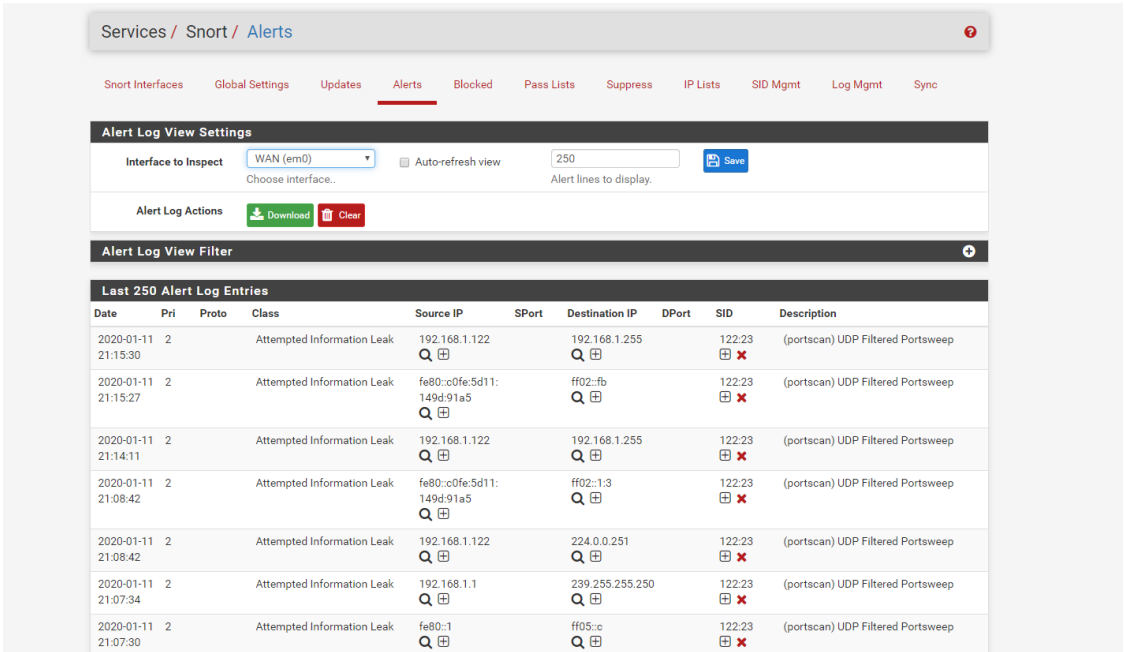
PFSense). Pelo que a implementação de uma firewall num tipo de rede com informação que não deverá ser publica é um ‘must have’.

Permite-nos, não só proteger toda a informação que guardamos como garantir a integridade de todos os sistemas, podendo ainda detetar intrusões como descrito no tópico das firewalls.

Além destas vulnerabilidades é ainda possível usar o Nessus para detetar coisas como o Spectre e o Meltdown, vulnerabilidades de processadores x86, no entanto isso já se torna um pouco complexo para o nosso caso de estudo.

Deteção do scan do Nessus – Snort (pfsense)

Enquanto era efetuada a análise de vulnerabilidades à máquina METASPLOITABLE protegida pela pfsense, o IDS snort foi registando um vasto número de alertas, tanto na interface WAN (onde se encontrava a máquina com o Nessus), como na interface LAN.



The screenshot shows the 'Alerts' tab in the Pfsense web interface. At the top, there's a navigation bar with 'Services / Snort / Alerts'. Below it, a sub-menu includes 'Snort Interfaces', 'Global Settings', 'Updates', 'Alerts' (selected), 'Blocked', 'Pass Lists', 'Suppress', 'IP Lists', 'SID Mgmt', 'Log Mgmt', and 'Sync'. The main content area has an 'Alert Log View Settings' section with a dropdown for 'Interface to Inspect' set to 'WAN (em0)', an 'Auto-refresh view' checkbox, and a text input for 'Alert lines to display' set to '250'. Below this is an 'Alert Log Actions' section with 'Download' and 'Clear' buttons. The 'Alert Log View Filter' section is empty. The 'Last 250 Alert Log Entries' table shows several entries for 'Attempted Information Leak' from various source IPs to destination IP 192.168.1.255 on port 122.23, all marked as '(portscan) UDP Filtered Portsweep'.

Date	Pri	Proto	Class	Source IP	SPort	Destination IP	DPort	SID	Description
2020-01-11 21:15:30	2		Attempted Information Leak	192.168.1.122	Q	192.168.1.255	122.23	⊞ ✖	(portscan) UDP Filtered Portsweep
2020-01-11 21:15:27	2		Attempted Information Leak	fe80::c0fe5d11:149d91a5	Q	ff02::fb	122.23	⊞ ✖	(portscan) UDP Filtered Portsweep
2020-01-11 21:14:11	2		Attempted Information Leak	192.168.1.122	Q	192.168.1.255	122.23	⊞ ✖	(portscan) UDP Filtered Portsweep
2020-01-11 21:08:42	2		Attempted Information Leak	fe80::c0fe5d11:149d91a5	Q	ff02::1:3	122.23	⊞ ✖	(portscan) UDP Filtered Portsweep
2020-01-11 21:08:42	2		Attempted Information Leak	192.168.1.122	Q	224.0.0.251	122.23	⊞ ✖	(portscan) UDP Filtered Portsweep
2020-01-11 21:07:34	2		Attempted Information Leak	192.168.1.1	Q	239.255.255.250	122.23	⊞ ✖	(portscan) UDP Filtered Portsweep
2020-01-11 21:07:30	2		Attempted Information Leak	fe80::1	Q	ff05::c	122.23	⊞ ✖	(portscan) UDP Filtered Portsweep

FIGURA 43 – Alertas apresentados pelo snort (pfsense)

Visto que o número de alertas é muito grande, é enviado em anexo um ficheiro de alertas do snort para cada interface, WAN e LAN, que se encontra na pasta pfsense_INTERNAL.

CONCLUSÃO

Tal como se pode ver neste relatório, mesmo tendo em conta a componente de configuração de redes, o cenário idealizado inicialmente foi na sua maioria implementado. Infelizmente, por dificuldades que nos apareceram ao longo da realização do projeto, a conceção e implementação ficou muito aquém do que era esperado e necessário para que a implementação corresse de forma mais favorável.

No entanto, mesmo tendo tudo isso em conta, este projeto permitiu que os conhecimentos adquiridos ao longo do semestre nesta unidade curricular fossem postos à prova.

Este projeto serviu também para explorar novas funcionalidades e serviços, descobrindo como se implementa coisas como comunicação segura com HTTPS e TLS e como se usam ferramentas como o Nessus que podem ser usadas para ter uma rede segura em que um Administrador de Sistemas se pode orgulhar.

PESQUISAS BIBLIOGRÁFICAS

- [1] - <https://www.name.com/partner/github-students>
- [2] - <https://ssproject.team>
- [3] - <https://www.digitalocean.com/community/tutorials/como-proteger-o-nginx-com-o-let-s-encrypt-no-ubuntu-18-04-pt>
- [4] - <https://markandruth.co.uk/2017/10/20/easily-setup-a-secure-ftp-server-with-vsftpd-and-letsencrypt>
- [5] - <https://turbofuture.com/internet/How-to-Set-Up-an-Intrusion-Detection-System-Using-Snort-on-pfSense-20>
- [6] - <https://www.iperiusbackup.net/en/freenas-how-to-install-and-configure-it-for-a-backup-to-nas/?fbclid=IwAR3792j6feRwBrslpAyle2xcuPljH2wUeSjhWji6wT8rPecLanPafQ9FGHQ>
- [7] - https://www.youtube.com/watch?v=Z0cDqF6HAXs&fbclid=IwAR1iPoBaNzdi_L8FafNJGC45AQu_bwa3zB3vo1o9b78Q0QUrzsoi6Zug8fXo
- [8] - https://www.youtube.com/watch?v=Z0cDqF6HAXs&fbclid=IwAR2nhRAVx-BAowZkuBUVTFFXT2Mmc0F6k6LABxMR_mc_1bc4TAz0CE3r6Rk
- [9] - <https://www.youtube.com/watch?v=EPz2bbfUb2U>
- [10] - <https://www.youtube.com/watch?v=DthbnPLBbRA>
- [11] - <https://www.youtube.com/watch?v=6s5wvmlESfo>
- [12] - <https://laravel.com/docs/6.x/configuration>
- [13] - <https://laravel.com/docs/5.8/installation>
- [14] - <https://laravel.com/docs/4.2/configuration>
- [15] - <https://www.snort.org>
- [16] - <https://paginas.fe.up.pt/~mgi98020/pgr/snort.htm>
- [17] - <https://www.tenable.com/downloads/nessus>

- [18] - <https://resources.infosecinstitute.com/a-brief-introduction-to-the-nessus-vulnerability-scanner/#gref>
- [19] - <https://www.youtube.com/watch?v=eeTZZN5U858>
- [20] - <https://www.youtube.com/watch?v=LXgZtmlt5T4>
- [21] - https://www.youtube.com/watch?v=Ma2G_9PXS5I