YNOV AIX-EN-PROVENCE

TP ICMP

Sécurité des SI

RIGONNAX Mickael – MONIER Sebastien 25/04/2018

Table des matières

1.		Objectif	2
2.		Analyse	2
	a.	Attaque ICMP Smurf	2
		Contexte	2
		Problématique	3
		Réalisation	3
	b.	Attaque ICMP Cover Channel Ptunnel	7
		Contexte	7
		Problématique	7
		Fonctionnement	7
	c.	Attaque ICMP Covert Channels Dissimulation1	10
		Contexte 1	10
		Problématique	10
		Fonctionnement	10
3.		Conclusion1	1

1. Objectif

L'objectif de ce TP est de comprendre le fonctionnement du protocole ICMP, connaitre ses faiblesses et savoir comment les exploiter.

Ce TP se déroule en deux parties :

- Attaque ICMP Smurf : cette attaque consiste à flooder la connexion d'un utilisateur ou d'un service avec des requêtes ICMP afin de réduire sa connexion voire la saturer et empêcher l'accès au réseau.
- Attaque ICMP Cover channel : cette attaque consiste à faire passer des informations sur le réseau en les dissimulant grâce au protocole ICMP.

2. Analyse

a. Attaque ICMP Smurf

Contexte

Ce TP a été réalisé en deux temps, une première fois sur un réseau privé avec un partage de connexion d'un smartphone en phase de test. Et une seconde fois en condition réelle sur le réseau de l'école Ynov d'Aix-en-Provence.

Voici les schémas correspondant aux deux infrastructures utilisées.

Réseau privé :

PC de l'attaquant OS Parrot 192.168.43.100	
PC de la victime OS Ubuntu 192.168.43.193	Broadcast 192.168.43.255/24

Réseau Privé

Réseau Ynov :



Dans chacun des TP deux machines ont été utilisée, une machine attaquante sous Parrot OS, et une seconde victime sous Ubuntu.

Problématique

Pour ces deux réalisations, la problématique est la suivante : Comment mettre une machine en déni de service avec ICMP ?

Réalisation

Le but de ce TP est de créer un déni de service sur le PC de la victime, pour ce faire nous utilisons le protocole ICMP. Le fonctionnement est très simple, il suffit d'envoyer des requêtes ICMP sur l'adresse de broadcast du réseau en se faisant passer pour la victime. L'envoie d'une requête sur le broadcast va entrainer une réponse de l'ensemble des utilisateurs du réseau vers la machine victime, qui va être surchargée de paquet.

En plus de ce mécanisme nous avons utilisé l'outil Hping3 qui nous permet d'envoyer un grand nombre de requête, de spoofer l'adresse de la victime et d'envoyer des paquets lourds et difficiles à traiter.

Fonctionnement schématisé :



Nous avons découpé cette attaque en deux parties. Dans la première partie un poste attaquant envoi des requêtes ICMP directement à un poste cible, cela n'empêchait pas l'accès au réseau mais en ralentissait grandement l'utilisation.

Pour cette partie nous utilisons un téléphone mobile afin d'être dans un réseau isolé. Le plan d'adressage de ce réseau est en 192.168.43.0/24 et l'IP du poste cible est 192.168.43.193



Dans cette commande nous pouvons voir la destination de l'adresse qui est celle de broadcast du réseau du smartphone. Le -a suivi de l'IP correspond à l'option spoof, dans notre cas, celle du PC de la victime. Pour finir le -d indique la taille des paquets à envoyer en bytes.

Après avoir lancé l'attaque, la victime a toujours accès à internet mais son temps de réponse augmente fortement.

6	64 bytes	from	8.8.8.8:	icmp_seq=33	ttl=53	time=48.9 ms
6	4 bytes	from	8.8.8.8:	icmp_seq=34	ttl=53	time=50.0 ms
6	4 bytes	from	8.8.8.8:	icmp_seq=35	ttl=53	time=48.5 ms
6	4 bytes	from	8.8.8.8:	icmp seq=36	ttl=53	time=57.3 ms
6	4 bytes	from	8.8.8.8:	icmp seq=37	ttl=53	time=48.4 ms
6	4 bytes	from	8.8.8.8:	icmp seq=38	ttl=53	time=61.6 ms
6	4 bytes	from	8.8.8.8:	icmp seq=39	ttl=53	time=59.1 ms
6	4 bytes	from	8.8.8.8:	icmp seq=40	ttl=53	time=263 ms
6	4 bytes	from	8.8.8.8:	icmp seq=41	ttl=53	time=181 ms
6	4 bytes	from	8.8.8.8:	icmp seq=42	ttl=53	time=54.0 ms
6	4 bytes	from	8.8.8.8:	icmp seq=43	ttl=53	time=54.5 ms
б	4 bytes	from	8.8.8.8:	icmp seq=44	ttl=53	time=244 ms
6	4 bytes	from	8.8.8.8:	icmp_seq=45	ttl=53	time=380 ms
6	4 bytes	from	8.8.8.8:	icmp seq=46	ttl=53	time=604 ms
6	4 bytes	from	8.8.8.8:	icmp seq=47	ttl=53	time=1168 ms
6	4 bytes	from	8.8.8.8:	icmp seq=48	ttl=53	time=348 ms
6	4 bytes	from	8.8.8.8:	icmp_seq=49	ttl=53	time=461 ms
6	4 bytes	from	8.8.8.8:	icmp_seq=50	ttl=53	time=520 ms
6	4 bytes	from	8.8.8.8:	icmp_seq=51	ttl=53	time=535 ms
6	4 bytes	from	8.8.8.8:	icmp seq=52	ttl=53	time=736 ms
6	4 bytes	from	8.8.8.8:	icmp seq=53	ttl=53	time=549 ms
6	4 bytes	from	8.8.8.8:	icmp_seq=54	ttl=53	time=469 ms
6	4 bytes	from	8.8.8.8:	icmp seq=55	ttl=53	time=505 ms
6	4 bytes	from	8.8.8.8:	icmp seq=56	ttl=53	time=474 ms
6	4 bytes	from	8.8.8.8:	icmp seq=57	ttl=53	time=767 ms
6	4 bytes	from	8.8.8.8:	icmp_seq=58	ttl=53	time=258 ms
6	4 bytes	from	8.8.8.8:	icmp seq=59	ttl=53	time=176 ms
6	4 bytes	from	8.8.8.8:	icmp seq=60	ttl=53	time=37.1 ms
6	4 bytes	from	8.8.8.8:	icmp seq=61	ttl=53	time=325 ms
6	4 bytes	from	8.8.8.8:	icmp seq=62	ttl=53	time=233 ms
6	4 bytes	from	8.8.8.8:	icmp seg=63	ttl=53	time=254 ms
6	4 bytes	from	8.8.8.8:	icmp seq=64	ttl=53	time=84.7 ms

Sur Wireshark nous avons aussi remarqué que les trames partent en continu de la victime vers le broadcast :

Арр	lications 👻 Places 👻	🚾 Wireshark 👻	Tue 09:19		,**	1 💉 🕬 🗋 🗸				
	k		*eth0							
File	<u>E</u> dit <u>V</u> iew <u>G</u> o <u>C</u> apt	ure <u>A</u> nalyze <u>S</u> tatistics	Telephony <u>W</u> ireless <u>T</u> ools	<u>H</u> elp						
	🔳 🔬 🔘 🗖 🚺	🖹 🏹 🔍 🔶	, + + ⊊ 🗐	ଇ ପ	<u>T</u>					
A	Apply a display filter <ctrl-></ctrl-> Expression +									
No.	Time	Source	Destination	Protocol	Length Info					
	3483 225.267210353	3 192.168.43.193	192.168.43.255	IPv4	1514 Fragmented IP	protocol (prot				
	3484 225.321679966	6 192.168.43.193	192.168.43.255	IPv4	1514 Fragmented IP	protocol (prot				
	3485 225.519618804	4 192.168.43.193	192.168.43.255	IPv4	1514 Fragmented IP	protocol (prot				
	3486 225.535995271	1 192.168.43.193	192.168.43.255	IPv4	1514 Fragmented IP	protocol (prot				
	3487 225.552215083	3 192.168.43.193	192.168.43.255	IPv4	1514 Fragmented IP	protocol (prot				
	3488 225, 566424736	6 192.168.43.193	192.168.43.255	IPv4	1514 Fragmented IP	protocol (prot				
	3489 225.579000301	1 192.168.43.193	192.168.43.255	IPv4	1514 Fragmented IP	protocol (prot				
	3490 225,605315188	8 192.168.43.193	192.168.43.255	TPv4	1514 Fragmented IP	protocol (prot				
	3491 225 618454384	4 192.168.43.193	192.168.43.255	TPv4	1514 Fragmented IP	protocol (prot				
	3492 225 65404700	2 192 168 43 193	192.168.43.255	TPv4	1514 Fragmented TP	protocol (prot				
	3/93 225 6667/95/	9 192 168 /3 193	192 168 /3 255	TPv4	1514 Fragmented TP	protocol (prot				
	3/19/ 225 6798621/	1 192 168 /3 193	192 168 /3 255	TPv/	151/ Fragmented TP	protocol (prot				
	3/95 225 695/9873	5 192 168 /3 193	192 168 /3 255	TPVA	151/ Fragmented TP	protocol (prot				
	3496 225 72144921	4 102 168 42 103	102 168 43 255	TDVA	1514 Fragmented TP	protocol (prot				
	2407 225 749400495	5 102 169 42 102	102 169 42 255	TDv4	1514 Fragmented TP	protocol (prot				
	2400 225 760642040	5 102 160 42 102	102 169 42 255	IFV4 TDv4	1514 Fragmented IF	protocol (prot				
	3450 225.70004201	0 100 160 40 100	102 169 42 255	IFV4	1514 Fragmented IF	protocol (prot				
	3499 225.805454538	b 192.100.43.193	192.100.43.255	IPV4	1514 Pragmented IP	protocor (prot				
	3500 225.936421468	5 192.108.43.193	192.108.43.255	TCP	1514 2857 → 0 [<non< th=""><th>e>] Seq-1 win-</th></non<>	e>] Seq-1 win-				
	3501 225.959769530	9 192.108.43.193	192.108.43.255	IPV4	1514 Fragmented IP	protocol (prot				
	3502 226.004339092	2 192.168.43.193	192.168.43.255	IPV4	1514 Fragmented IP	protocol (prot				
	3503 226.01949428	7 192.168.43.193	192.168.43.255	IPV4	1514 Fragmented IP	protocol (prot				
	3504 226.106315825	5 192.168.43.193	192.168.43.255	IPV4	1514 Fragmented IP	protocol (prot				
	3505 226.198942120	6 192.168.43.193	192.168.43.255	IPV4	1514 Fragmented IP	protocol (prot				
•						•				
	nternet Protocol V	ersion 4, Src: 192	.168.43.193, Dst: 192.1	68.43.255		A				
▼ Di	ata (1480 bytes)									
	Data: 58585858585	8585858585858585858585	858585858585858585858							
	[Length: 1480]					T				
04f	0 58 58 58 58 58	58 58 58 58 58 58	3 58 58 58 58 58 58 XXXX	XXXX XXXXX	XXX					
050	0 58 58 58 58 58	58 58 58 58 58 58	3 58 58 58 58 58 58 XXXX	XXXX XXXXXX	XXX	▼				
0	wireshark_eth0_201	L80410085133_CKQa0V	V	Pack	kets: 3505 · Displayed: 3505 (100.0%) Profile: Default				

L'attaque ne permet pas de bloquer totalement l'accès du poste de la victime, car il y a seulement deux hôtes dans ce réseau.

Afin de bloquer totalement l'accès d'un poste nous avons reproduit exactement la même attaque mais cette fois-ci sur le réseau de l'école Ynov. Le réseau comporte au moins un cinquantaines d'étudiants.

Le même utilitaire est utilisé dans cette partie, dans ce cas la victime est le poste qui a pour IP 10.13.7.161



Après avoir lancé l'attaque nous observons l'impact dans WireShark, et nous découvrons que la victime reçoit des paquets ICMP de l'ensemble des machines du réseau.

Capture Wireshark :

	4449	750.538459	749 10.	13.0.6	4		10.13	.1.77	1	ECMP	1042	Echo	(ping)	reply	1d=0x820c,	seq=35584/139,	ttl=64
	4449	750.538471	513 10.	13.5.3	1		10.13	.1.77	1	[CMP	1042	Echo	(ping)	reply	1d=0x820c,	seq=41475/930,	ttl=64
	4449	750.538472	962 10.	13.5.3	1		10.13	.1.77	1	ECMP	1042	Echo	(ping)	reply	id=0x820c,	seq=42499/934,	ttl=64
	4449	750.538474	132 10.	13.5.3	1		10.13	.1.77	1	ECMP	1042	Echo	(ping)	reply	id=0x820c,	seq=42755/935,	ttl=64
	4449	750.551944	366 10.	13.0.6	4		10.13	.1.77	1	ECMP	1042	Echo	(ping)	reply	id=0x820c,	seq=40963/928,	ttl=64
	4449	750.551956	105 10.	13.0.6	4		10.13	.1.77	1	ECMP	1042	Echo	(ping)	reply	id=0x820c,	seq=41219/929,	ttl=64
	4449	750.551958	231 10.	13.0.6	4		10.13	.1.77	1	ECMP	1042	Echo	(ping)	reply	id=0x820c,	seq=41475/930,	ttl=64
	4449	750.551961	562 10.	13.0.6	4		10.13	.1.77	1	CMP	1042	Echo	(ping)	reply	id=0x820c,	seq=42499/934,	tt1=64
	4449	758.551963	121 10.	13.5.3	1		10.13	.1.77	1	CMP	1042	Echo	(ping)	reply	1d=0x820c,	seq=43011/936,	tt1=64
	4449	750.551965	170 10.	13.5.3	1		10.13	.1.77	1	[CMP	1042	Echo	(ping)	reply	1d=0x820c,	seq=45315/945,	tt1=64
	4449	750.551966	967 10.	13.0.6	4		10.13	.1.77	1	[CMP	1042	Echo	(ping)	reply	1d=0x820c,	seq=42755/935,	ttl=64
	4449	750.551968	578 10.	13.0.6	4		10.13	.1.77	1	ICMP	1042	Echo	(ping)	reply	1d=0x820c,	seq=43011/936,	ttl=64
	4449	750.621797	372 10.	13.0.6	4		10.13	.1.77	1	ECMP	1042	Echo	(ping)	reply	id=0x820c,	seq=45827/947,	ttl=64
	4449	750.621810	791 10.	13.0.6	4		10.13	.1.77	1	[CMP	1042	Echo	(ping)	reply	id=0x820c,	seq=46595/950,	ttl=64
	4449	750.621812	617 10.	13.5.3	1		10.13	.1.77	1	ECMP	1042	Echo	(ping)	reply	id=0x820c,	seq=47107/952,	ttl=64
	4449	750.631801	154 10.	13.5.3	1		10.13	.1.77	1	ECMP	1042	Echo	(ping)	reply	id=0x820c,	seg=48387/957,	ttl=64
	4449	750.631814	180 10.	13.0.6	4		10.13	.1.77	1	CMP	1042	Echo	(ping)	reply	id=0x820c,	seq=46851/951,	ttl=64
	4449	758.632788	918 10.	13.0.6	4		10.13	.1.77	1	[CMP	1042	Echo	(ping)	reply	id=0x820c,	seq=47107/952,	ttl=64
	4449	758.632712	578 10.	13.0.6	4		10.13	.1.77	1	ICMP.	1042	Echo	(ping)	reply	1d=0x820c,	seg=47363/953,	tt1=64
	4449	750.698464	122 10.	13.0.6	4		10.13	.1.77	1	CMP	1042	Echo	(ping)	reply	1d=0x820c,	seg=48131/956,	tt1=64
	4449	750.838460	195 10.	13.5.3	1		10.13	.1.77	1	CMP	1042	Echo	(ping)	reply	1d=0x820c,	seg=48643/958,	tt1=64
	4449	750.838471	182 10.	13.5.3	1		10.13	.1.77	1	ECMP	1042	Echo	(ping)	reply	1d=0x820c,	seg=48899/959,	tt1=64
	4449	750.838473	034 10.	13.5.3	1		10.13	.1.77	1	CMP	1042	Echo	(ping)	reply	id=0x820c,	seg=49155/960,	tt1=64
	4449	750.838474	243 10.	13.5.3	1		10.13	.1.77	1	CMP	1042	Echo	(ping)	reply	id=0x820c.	seg=49667/962.	tt1=64
	4449	750.838475	645 10.	13.5.3	1		10.13	.1.77	1	CMP	1042	Echo	(ping)	reply	id=0x820c.	seg=50179/964,	tt1=64
	4449	750.838477	000 10.	13.0.6	4		10.13	.1.77	1	CMP	1042	Echo	(ping)	reply	id=0x820c.	seg=48643/958,	tisl=64
	4449	750.838478	297 10.	13.0.6	4		10.13	.1.77	1	CMP	1042	Echo	(ping)	reply	id=0x820c,	seg=48899/959,	t:==64
	4449	750.838479	757 10.	13.0.6	4		10.13	.1.77	1	CMP	1042	Echo	(ping)	reply	1d=0x820c.	seg=49155/960.	tt1=64
	4449	750.838480	988 10.	13.0.6	4		10.13	.1.77	1	CMP	1042	Echo	(ping)	reply	1d=0x820c.	seg=49667/962.	tt1=64
	4449	758.838482	249 10.	13.5.3	1		10.13	1.77	1	CMP	1042	Echo	(ping)	reply	1d=0x820c	seg=50947/967.	tt1=64
	4449	758,838483	753 10.	13.0.6	â.		10.13	.1.77	÷	CMP	1042	Echo	(ping)	reply	1d=0x820c.	seg=50179/964.	tt1=64
	4449	758,838485	047 10.	13.0.6	á l		10.13	.1.77	-	CMP	1042	Echo	(ping)	reply	1d=0x820c.	seg=50435/965.	tt1=64
	4449	758,838486	181 10.	13.0.6	á.		10.13	.1.77	ī	CMP	1642	Echo	(ping)	reply	1d=0x820c.	seg=59691/966.	tt1=64
	4449	758,838488	729 10.	13.0.6	ā.		10.13	.1.77	-	CMP	1642	Echo	(ping)	reply	id=0x820c.	seg=50947/967.	tt1=64
	4449	758.846983	156 10.	13.0.6	á l		10.13	.1.77	i	CMP	1642	Echo	(ping)	reply	id=0x820c.	seg=51203/968.	tt1=64
	4449	758.846993	535 10.	13.5.3	1		10.13	1.77	-	CMP	1642	Echo	(ping)	reply	id=0x820c	seq=51203/968.	tt1=64
	4449	758.846995	082 10.	13.5.3	1		10.13	.1.77	-	CMP	1042	Echo	(ping)	reply	id=0x820c	seg=51971/971.	tt1=64
	4449	758.846996	545 10	13.5.3	1		10.13	1.77	-	CMP	1642	Echo	(ning)	renly	1d=0x820c	seq=52483/973	tt1=64
	4449	758 846997	769 10	13.5.3	1		10.13	1.77	-	CMP	1642	Echo	(ning)	renly	1d=0x820c	seg=52995/975	tt1=64
	4449	758 846998	955 10	13 0 6	â		10 13	1 77	-	CMP	1642	Echo	(ning)	renly	1d=0x820c	seg=51459/969	tt1=64
	Erame	9470: 60 b	tes on	wire i	(480 h	its).	60 hv	tes cant	tured (486	hits) o	n int	erfac	e A				
	Etheri	net II Src	Annle	42.92	10 (1	8.08-1	56.43	93·1c)	Dst Hone	aiPr 16	aa·Rf	(hA)	10.41.1	16.33.81	n .		
	 Interr 	net Protoco	Versi	on 4 5	Src 1	0.13	8 183	Det - 16	13 7 161	01/1_101	aa.oi	(00.	101411	10.00.01	,		
	 Interr 	net Control	Mossan	e Prote	ncol		012007	DOCI 1									
	- 20000	ice concret	neosug		0002												
	0000 10	40 44 40	- 07 17		0.40	00.4-	00.00	45.00		140 5							
	0000 00	10 41 10 8	a 81 D8	5 68 5	0 43	93 10	00 07	45 88	A	VCE.							
	0010 00	20 88 79 6	6 00 46	9 81 0	15 DC	ea eo	00 07	ea eo	·e····g·	····g.							
	0020 07	a1 00 00 2	T 39 54	1 0D C	3 01	38 58	28 28	58 58	/91.		6						
	0030 56	38 38 38 8	0 00 00	9 69 6	0 00	89 88			****								
	0 7	wlan0: <live< td=""><td>capture</td><td>in progr</td><td>ress></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></live<>	capture	in progr	ress>												
I	diff Man		ten de s	and a file			Damak	Tamaina		Do no	at Tax	-		_	Descel Termine		Inc. make
1	🖉 Men	u 💽 Cer	tre de c	controle		<u> </u>	Parrot			Parr	ot ler				Parrot lermina		las root o

Afin de vérifier si l'attaque a été un succès nous tentons de rejoindre le site google.fr depuis le poste de la victime, et effectivement l'accès au réseau est impossible. Le nombre de paquets envoyés à la victime est tellement important que la carte réseau ne peut plus les gérer et met donc le poste en déni de service.

Recherche internet :



b. Attaque ICMP Cover Channel Ptunnel

Contexte

Ce TP a été réalisé sur le réseau de l'école Ynov d'Aix-en-Provence. Plusieurs machines ont été manipulées, comme le poste de l'attaquant qui est une machine sous Kali Linux, le firewall de l'école et une machine qui sert de serveur Proxy hébergée dans le cloud.

Problématique

La problématique est la suivante, comment réussir à communiquer avec un serveur bloqué par le Proxy/Firewall ?

Pour répondre à cette question, nous avons installé et utilisé PTunnel qui est un proxy ICMP et qui permet d'encapsuler le trafic HTTP dans des paquets ICMP et donc d'être invisible pour les éléments de sécurité.

Fonctionnement

L'outil PTunnel fonctionne de la manière suivante :



Dans le cadre de ce TP nous avons voulu accéder à des sites bloqués par le pare-feu d'Ynov, pour cela nous avons travaillé en collaboration avec le BSI (bureau des systèmes d'information).

Pour réaliser cette action nous avons utilisé l'outil Ptunnel qui permet d'encapsuler les requêtes http dans de l'ICMP. Il est basé sur un modèle client-serveur, dans notre cas le client est dans le réseau de l'école Ynov et le serveur est situé dans le cloud chez Digital Océan.

L'infrastructure est la suivante :



Le but de ce TP est donc d'accéder à un site qui est bloqué par le pare-feu, nous tenterons d'abord un accès sur le site <u>http://apache.org</u> qui a préalablement été bloqué par le BSI d'Ynov.

Nous avons d'abord installé et lancé PTunnel sur un serveur Debian 9 herbergé chez dans le cloud :

root@debian-s-1vcpu-2gb-lon1-01:~# ptunnel -c eth0 -x Ynov2018!
[inf]: Starting ptunnel v 0.72.
<pre>[inf]: (c) 2004-2011 Daniel Stoedle, <daniels@cs.uit.no></daniels@cs.uit.no></pre>
[inf]: Security features by Sebastien Raveau, <sebastien.raveau@epita.fr></sebastien.raveau@epita.fr>
[inf]: Forwarding incoming ping packets over TCP.
[inf]: Initializing pcap.
<pre>[inf]: Ping proxy is listening in privileged mode.</pre>
[inf]: Incoming tunnel request from 195.200.178.226.
[inf]: Starting new session to 52.41.96.17:80 with ID 44690
[err]: Dropping duplicate proxy session request.
[inf]: Received session close from remote peer.
[inf]: Received session close from remote peer.
[inf]: Received session close from remote peer.
[inf]: Received session close from remote peer.

L'outil se lance tout simplement avec la commande avec la commande « ptunnel », le -c correspond à l'interface à utiliser et le -x correspond au mot de passe à entrer pour s'authentifier côté client.

Il reste maintenant à lancer le client PTunnel côté client, dans notre cas sur une machine Kali Linux :



La même installation de PTunnel est nécessaire sur le client, il est cependant installé automatiquement sur Kali Linux. Les différentes options utilisées sont -p pour indiquer l'adresse IP de notre serveur distant, -lp pour indiquer le port d'écoute de notre client. C'est sur ce port où nous allons nous connecter via le navigateur Web. L'option -da indique l'URL à accéder, -dp pour le port distant et enfin -x pour le mot de passe nécessaire à l'authentification.

Une fois la commande lancée, le tunnel ICMP est créé et nous pouvons accéder à notre site Web bloqué sur localhost:4444 dans un navigateur.



Le site apache.org est donc accessible via le tunnel ICMP, même si ce dernier est bloqué par le parefeu, les paquets http pour apache.org sont transparents pour le pare-feu car ils sont à l'intérieur des paquets ICMP.

Par contre l'utilitaire PTunnel devient vieillissant et ne fonctionne pas avec l'ensemble des sites, dans notre cas nous n'avons pas réussi à le faire fonctionner avec des sites en https et/ou qui utilisent flash player.

Mais les flux traversent bien le firewall même pour des sites bloqués comme Netflix, qui nous renvoie un message d'erreur :

				Netflix - Error report
Netflix - Error report	×	+		
$\leftarrow \rightarrow $ C $rac{1}{2}$		① 127.0.0.1:5555		
HTTP Status 404 -				
type Status report				
message				
description The requested resour	rce is no	ot available.		
Netflix				

La même commande a été utilisée pour utiliser le site Netflix, avec seulement le port qui a changé, le 443 à la place du port 80.

c. Attaque ICMP Covert Channels Dissimulation

Contexte

Ce TP a été réalisé sur le réseau de l'école Ynov d'Aix-en-Provence. 2 machines sous Kali Linux ont été nécessaires à sa réalisation.

Problématique

Comment administrer un outil à distance, ou communiquer de façon cachée avec ICMP ?

Problématique nous allons utiliser l'outil Hping qui permet de manipuler les trames ICMP.

Fonctionnement

Le principe de l'attaque est le suivant, nous avons plusieurs machines sur un réseau et leurs communications sont bloquées par un pare-feu ou autres. La communication peut avoir lieu pour plusieurs répons, communication entre 2 personnes, administration d'un malware, etc.

Voici l'architecture dans laquelle nous avons effectué cette attaque :



Deux machines sont présentes dans le réseau de l'école doivent pouvoir communiquer en étant transparente sur le réseau de l'école, dans notre cas deux machines Kali Linux.

La communication sera faite via de simples requêtes ICMP sur le réseau car l'ICMP n'est que très rarement bloqué sur un réseau. Pour ce faire nous avons utilisé encore une fois l'outil hping3 qui permet de stocker des données dans des trames ICMP (commande, message, etc.).

Son utilisation s'est faite de la manière suivante :

- Depuis le PC 1 nous avons envoyé la requête suivante :
 - Hping3 -1 -c 1 10.13.4.250 -e « TPSECU »

Cette commande va donc envoyer une seule requête grâce à l'argument –c 1 qui correspond au « count », en ICMP avec le -1 qui est le choix du protocole, et le plus important elle va placer notre chaîne de caractères « TPSECU » dans l'emplacement data de la requête ICMP. L'IP rentrée est l'IP de la machine qui recevra cette requête, voici le résultat avec une capture de trame sur cette même machine :

5110 2011T414TT10	TO'TO'O'TTO	TO'TO'L'500	001	202 24272 - 24272 FOU-202
2774 98.715574477	10.13.4.250	10.13.7.161	ICMP	60 Echo (ping) request id=0x1c17, seq=2560/10, ttl=64 (reply in 2775)
2775 98.715602851	10.13.7.161	10.13.4.250	ICMP	48 Echo (ping) reply id=0x1c17, seq=2560/10, ttl=64 (request in 2774)
2776 98.817492937	98:22:ef:99:38:d1	Broadcast	ARP	60 Who has 10.13.1.129? Tell 10.13.6.46
2777 98.817518699	98:22:ef:99:38:d1	Broadcast	ARP	60 Who has 10.13.5.203? Tell 10.13.6.46
2778 98.817538750	10.13.0.133	10.13.7.255	UDP	305 54915 → 54915 Len=263
2779 98.817561941	98:22:ef:99:38:d1	Broadcast	ARP	60 Who has 10.13.1.94? Tell 10.13.6.46
2780 98.920688642	98:22:ef:99:38:d1	Broadcast	ARP	60 Who has 10.13.1.145? Tell 10.13.6.46
0000 b0 10 41 16 aa	8f f8 16 54 16 50 df	08 00 45 00	A T.PE.	
0010 00 22 64 55 00	00 40 01 f5 d1 0a 0d	07 a1 0a 0d	."dU@	
0020 04 fa 00 00 ec	fd 1c 17 0c 00 54 50	53 45 43 55		
wireshark_wlo1_20	180417103928_HmcmWW			Paquets: 2863

Nous voyons très bien ici l'arrivée du paquet ICMP qui contient bien dans son paquet notre chaîne de caractères. Cette méthode peut être utilisée également pour l'administration d'outil, de l'injection de commande, etc.

3. Conclusion

Pour conclure sur ce TP, nous allons d'abord faire une analyse rapide du protocole ICMP et pour finir les protections que nous pouvons mettre en œuvre.

Ce protocole est indispensable au bon fonctionnement du réseau car il gère à lui seul une grande partie des remontées d'erreur (problème de réseau, configuration, etc.). Il est donc impossible de bloquer totalement ce protocole sous peine d'avoir des gros ralentissements et problème sur le réseau.

Comme vu tout au long de ce TP, ICMP présente de multiples failles, elles permettent de réaliser des communications cachées et des dénis de service par exemple. Pour sécuriser l'utilisation de ce protocole ils existent plusieurs moyens et outils comme :

- Le blocage des requêtes ICMP sur l'IP de broadcast
- Utiliser un HIDS (Wazuh par exemple) qui permet de déceler un nombre de requête anormale et de les bloquer automatiquement
- Le même HIDS peut également intégrer une interface pour suivre en temps réel les alertes liées à ce protocole
- La mise en place d'un NIDS (Snort, Suricata) permet aussi de protéger une partie de son réseau des attaques ICMP