

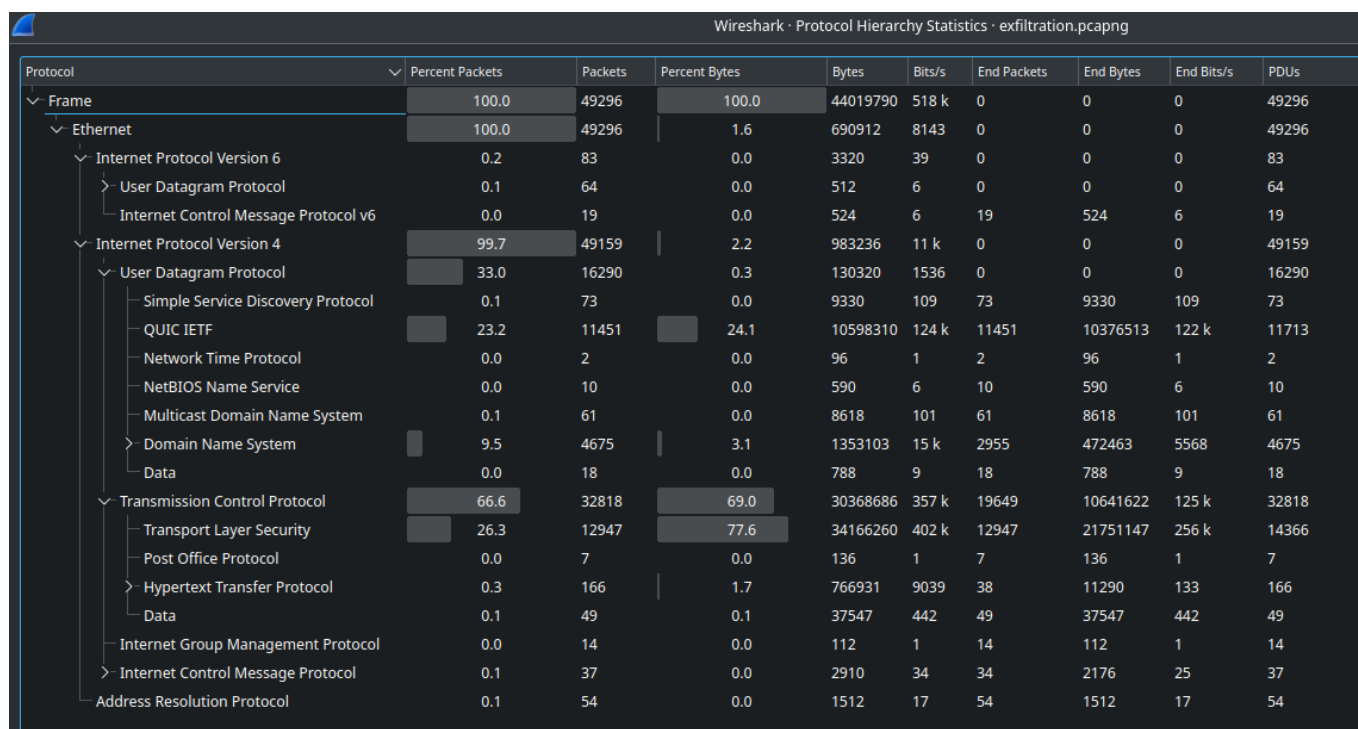
Exfiltration

Auteur: bWlrYQ

1. Recon

L'infrastructure de votre CTF préféré a été attaquée... Les indices de compromission (IOC) indiquent que de la donnée a été exfiltrée, on se retrouve avec une capture réseau assez conséquente à analyser.

Etant donné que le nombre de paquets est important, on va commencer par faire le tri à l'aide des statistiques offertes par Wireshark.



Wireshark · Protocol Hierarchy Statistics · exfiltration.pcapng

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDU's
Frame	100.0	49296	100.0	44019790	518 k	0	0	0	49296
Ethernet	100.0	49296	1.6	690912	8143	0	0	0	49296
Internet Protocol Version 6	0.2	83	0.0	3320	39	0	0	0	83
User Datagram Protocol	0.1	64	0.0	512	6	0	0	0	64
Internet Control Message Protocol v6	0.0	19	0.0	524	6	19	524	6	19
Internet Protocol Version 4	99.7	49159	2.2	983236	11 k	0	0	0	49159
User Datagram Protocol	33.0	16290	0.3	130320	1536	0	0	0	16290
Simple Service Discovery Protocol	0.1	73	0.0	9330	109	73	9330	109	73
QUIC IETF	23.2	11451	24.1	10598310	124 k	11451	10376513	122 k	11713
Network Time Protocol	0.0	2	0.0	96	1	2	96	1	2
NetBIOS Name Service	0.0	10	0.0	590	6	10	590	6	10
Multicast Domain Name System	0.1	61	0.0	8618	101	61	8618	101	61
Domain Name System	9.5	4675	3.1	1353103	15 k	2955	472463	5568	4675
Data	0.0	18	0.0	788	9	18	788	9	18
Transmission Control Protocol	66.6	32818	69.0	30368686	357 k	19649	10641622	125 k	32818
Transport Layer Security	26.3	12947	77.6	34166260	402 k	12947	21751147	256 k	14366
Post Office Protocol	0.0	7	0.0	136	1	7	136	1	7
Hypertext Transfer Protocol	0.3	166	1.7	766931	9039	38	11290	133	166
Data	0.1	49	0.1	37547	442	49	37547	442	49
Internet Group Management Protocol	0.0	14	0.0	112	1	14	112	1	14
Internet Control Message Protocol	0.1	37	0.0	2910	34	34	2176	25	37
Address Resolution Protocol	0.1	54	0.0	1512	17	54	1512	17	54

La majorité des échanges dans la capture sont chiffrés (TLS, QUIC), on peut donc explorer les différentes techniques utiles au déchiffrement de ces protocoles mais ça ne donne rien, aucun résultat.

Il faut donc aller vers d'autres protocoles présents en quantité importante (car on rappelle que c'est de l'exfiltration de données). On peut voir que le protocole DNS représente 10% des échanges de la capture ce qui est un montant assez conséquent. Essayons de filtrer les paquets DNS.

On peut voir que la grande majorité des paquets sont des requêtes vers un même domaine dont les noms de sous-domaines sont assez étranges. On voit aussi des requêtes vers d'autres services comme Twitter, Spotify... Mais on peut se concentrer sur le domaine oastify.com pour le moment.

Il existe une technique d'exfiltration courante appelée exfiltration DNS, elle est généralement utilisée par les attaquants pour extraire de la donnée post-compromission. On est désormais presque sûr que c'est ce qui a été utilisé dans notre cas, il reste à déterminer le type des données exfiltrées afin de pouvoir les traiter au mieux.

Après une rapide analyse, la donnée exfiltrée semble être le nom de la machine sur le sous-domaine `uzzx72ps81pk1cdc3qzvfishqsgomea3` du domaine `oastify.com`. Cette donnée est au format hexadécimal, un

oeil aguerri pourrait repérer le type de donnée exfiltrée au premier coup d'oeil sur le premier échange DNS vers le domaine oastify. Dans notre cas nous allons simplement faire une recherche google [ici](#) avec les premiers octets. On y apprend que c'est la signature des fichiers PNG.

2. Extraction de la donnée

Il nous faut maintenant extraire le contenu de requêtes DNS faites à ce domaine, pour se faire on peut utiliser le one-liner suivant : `tshark -r exfiltration.pcapng -Y '(ip.src==192.168.5.77)' -T fields -e dns.qry.name > requests.txt`

Cela nous permet d'obtenir toutes les requêtes dans un .txt que l'on va pouvoir traiter par la suite avec un script python. D'après les informations récoltées sur les premiers échanges en les décodant il ne semble pas qu'il y ait de padding particulier ou de règle à respecter pour reconstituer notre image, c'est une exfiltration "nature" qui a été faite.

On peut établir le script suivant:

```
from os import system
with open('requests.txt','r') as f:
    content = f.read().splitlines()
    reqs = []
    for i in range(len(content)):
        if 'oastify' in content[i]:
            if 'polling' in content[i]:
                pass
            else:
                reqs.append(content[i])
    data = []
    for i in range(len(reqs)):
        data.append(reqs[i].replace(".uzx72ps81pk1cdc3qzvfisqxgomea3.oastify.com",""))

    olddata = ""
    final = ""
    for i in range(len(data)):
        try:
            olddata=data[i-1]
        except:
            pass
        if olddata == data[i]:
            pass
        else:
            final+=data[i]
    with open ('flag','w') as flag:
        flag.write(final)
    system('xxd -p -r flag flag.png')
    flag.close()
f.close()
```

Lors de l'analyse de la capture, j'ai remarqué qu'une requête vers la machine "polling" a été réalisé, celle-ci n'est pas au format hexadécimal et n'est pas de la même taille que les autres, j'ai donc fait en sorte qu'elle soit exclue lors du traitement, sinon cela fausse notre fichier.

Par ailleurs, j'effectue une condition pour vérifier que chaque nouvelle donnée que je récupère est différente de la précédente. Dans un échange DNS si aucune réponse n'est reçue, alors le client renvoie la même demande. Si on ne traite pas cette possible condition alors nos données sont elles aussi faussées.

J'utilise ensuite `xxd` pour reconstituer la donnée d'un format hexadécimal vers un format de fichier binaire en `.png`

3. Flag !

C'est à ce moment là qu'on peut ouvrir notre précieux flag et dire "FLAG". `FMCTF{DNS_Exfil_4TW}`

