Отчет анализа дампа Серов Глеб Ильич

1. Участвующие хосты

В дампе присутствуют следующие основные узлы:

IP-адрес	**Роль**
192.168.118.129	Клиент
192.168.118.128	Сервер
192.168.118.1	Предполагаемый шлюз (ответов не наблюдается в дампе)

Сеть относится к диапазону 192.168.118.0/24.

2. Общая картина взаимодействия.

Взаимодействие между клиентом и сервером происходит в несколько этапов:

1. ARP-резолвинг (клиент пытается определить MAC-адреса других узлов).
2. HTTP-соединение (порт 1337) – передача данных в открытом виде.
3. TLS-соединение (порт 1338) – защищенный канал.

3. Анализ переданных данных.

ARP-трафик

В дампе наблюдаются ARP-запросы вида “Who has 192.168.118.1?/Who has 192.168.118.128?”. Ответы получены только от сервера 192.168.118.128 (рис. 1).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	VMware_23:75:b6	Broadcast	ARP	42	Who has 192.168.118.1? Tell 192.168.118.129
2	3.005975	VMware_23:75:b6	Broadcast	ARP	42	Who has 192.168.118.1? Tell 192.168.118.129
3	4.004544	VMware_23:75:b6	Broadcast	ARP	42	Who has 192.168.118.1? Tell 192.168.118.129
4	5.004453	VMware_23:75:b6	Broadcast	ARP	42	Who has 192.168.118.1? Tell 192.168.118.129
5	8.009009	VMware_23:75:b6	Broadcast	ARP	42	Who has 192.168.118.1? Tell 192.168.118.129
6	9.008399	VMware_23:75:b6	Broadcast	ARP	42	Who has 192.168.118.1? Tell 192.168.118.129
7	9.509763	VMware_23:75:b6	Broadcast	ARP	42	Who has 192.168.118.128? Tell 192.168.118.129
8	9.511210	VMware_ff:a1:64	VMware_23:75:b6	ARP	60	192.168.118.128 is at 00:0c:29:ff:a1:64

Рисунок 1 – Ответ от сервера на ARP запрос.

В дампе не наблюдается защиты ARP (по типу ARP ACL, DAI), то есть, в сети отсутствуют механизмы предотвращения ARP-spoofing.

HTTP-трафик (порт 1337).

Клиент выполняет HTTP-запрос “GET /k3y HTTP/1.1”. Ответ сервера содержит закрытый криптографический ключ в формате PEM (рис. 2).

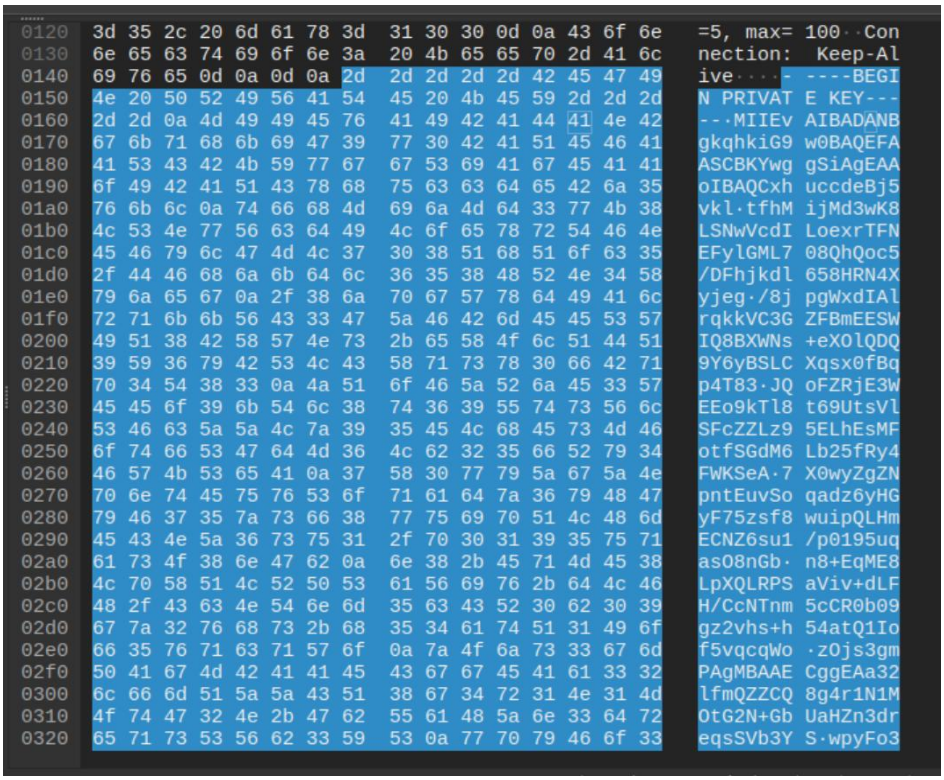

Рисунок 2 - Ответ сервера

Данный ключ передается без шифрования, что является критической утечкой секретной информации.

TLS-трафик (порт 1338).

После получения ключа клиент устанавливает TLS-соединение с сервером. Можно проследить следующий используемый параметр (рис. 3): Cipher Suite (TLS_RSA_WITH_AES_256_CBC_SHA) – используется RSA key exchange, отсутствует Perfect Forward Secrecy (что достигается с помощью

DHE/ECDHE). То есть, RSA применяется для шифрования premaster secret, который генерируется клиентом, шифруется публичным ключом сервера, расшифровывается сервером приватным ключом, и далее из premaster secret вычисляются master secret, session keys. То есть, безопасность сессии зависит от приватного ключа сервера.

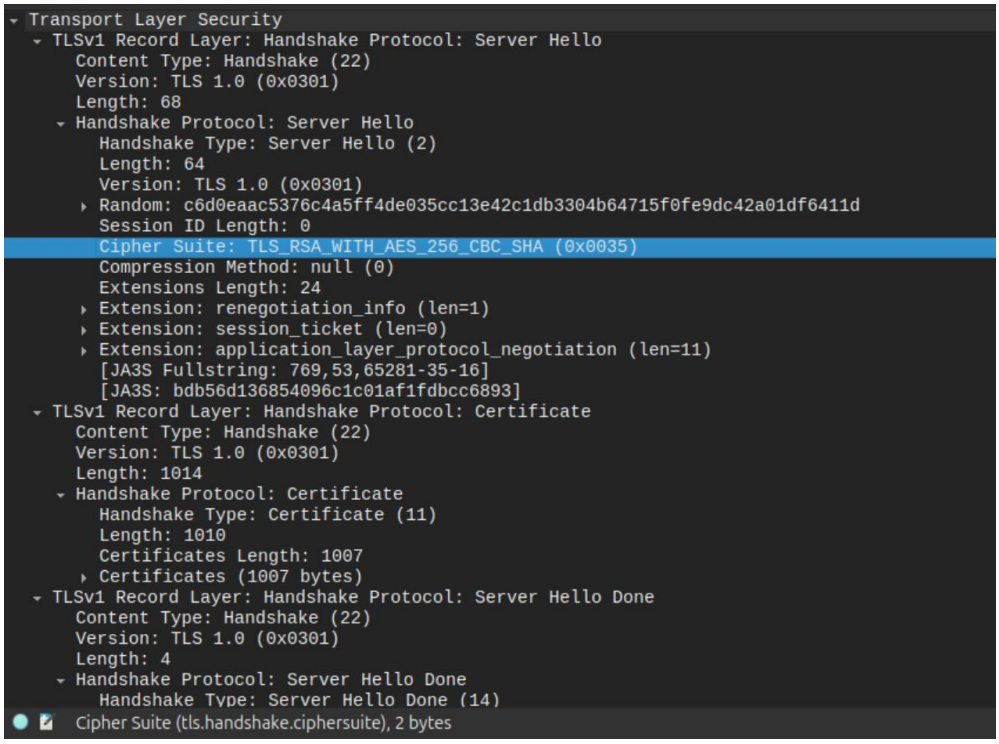

Рисунок 3 – Использованный Cipher Suite

Далее возникла идея расшифровать TLS. Был скопирован полученный раннее ключ, проверен на валидность (рис. 4).

Рисунок 4 – Проверка ключа на валидность

Далее в Wireshark был расшифрован TLS (рис. 5)

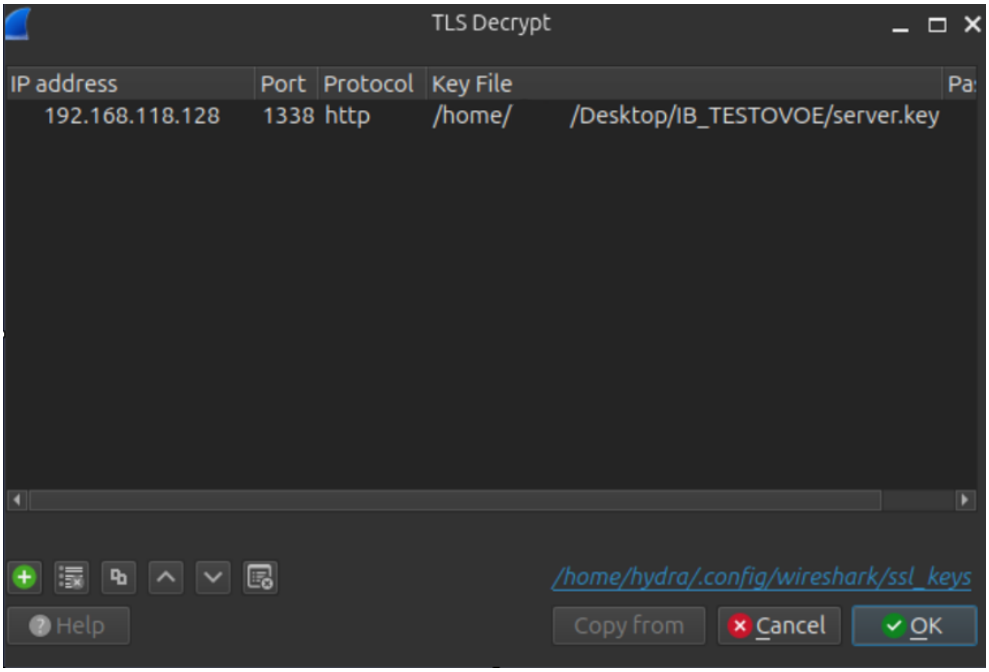

Рисунок 5 – Расшифровка TLS

В результате расшифрования обнаружен еще один GET запрос (рис. 6).

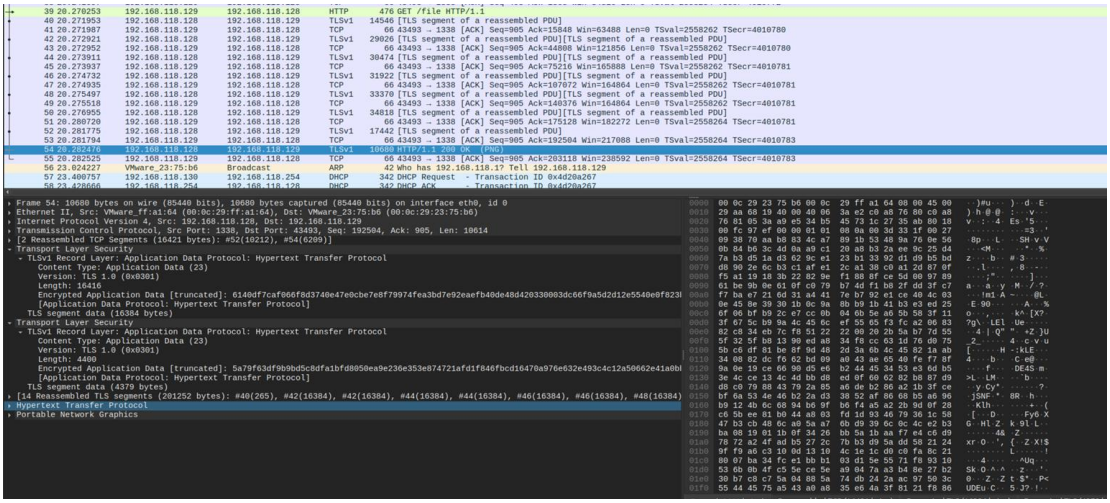

Рисунок 6 – GET-request

В ответе – PNG файл (рис. 7).

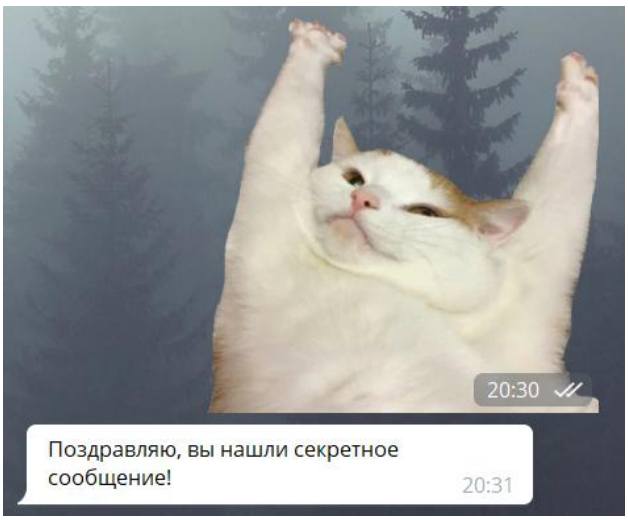

Рисунок 7 – Полученный PNG файл

5. Итоговые выводы

В дампе зафиксировано клиент-серверное взаимодействие. Передавались ARP-requests, HTTP-requests, закрытый RSA-ключ, TLS-трафик, бинарный PNG-файл. В результате произошла утечка закрытого ключа, TLS-трафик был расшифрован.

Из уязвимостей можно отметить отсутствие защиты от MITM в локальной сети (ARP без механизмов защиты). Закрытый ключ передается по HTTP – наиболее критичная уязвимость. Закрытый RSA-ключ передается в октрытом виде, компрометируется вся модель доверия TLS. Также используется устаревший Cipher Suite.

Также HTTP-headers раскрывают тип сервера и версию ПО, что потенциально облегачает подбор известных уязвимостей.