..		
README.md	bsidessf	3 months ago
e_corp_pki.pcapng	bsidessf	3 months ago
factorize.py	bsidessf	3 months ago
priv.key	bsidessf	3 months ago

README.md

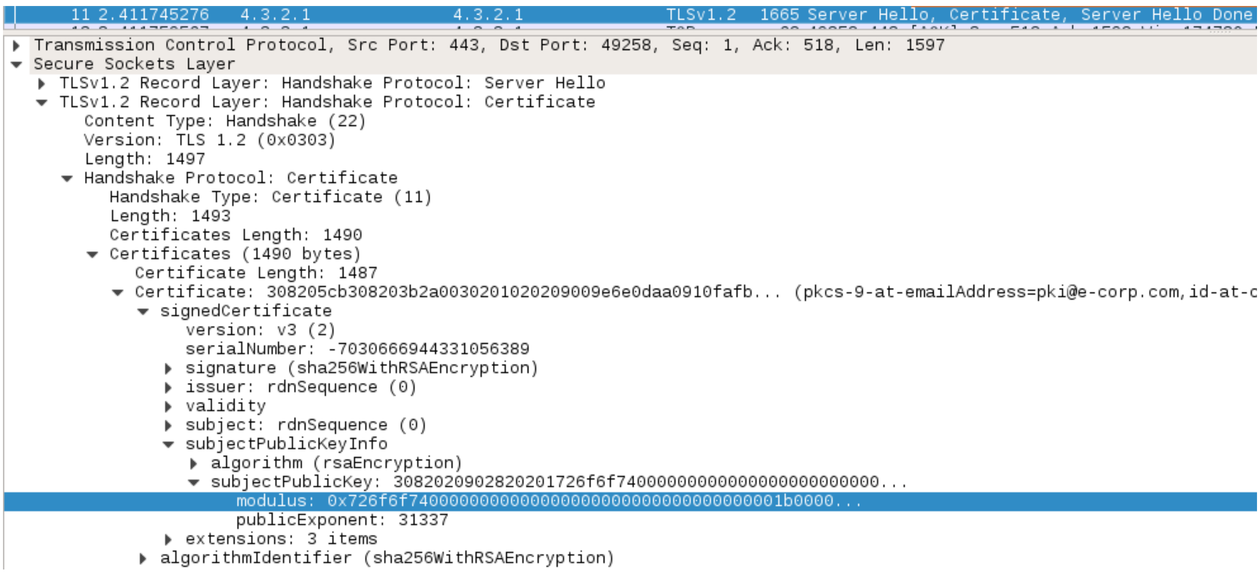
BsidesSF 2017 - []root (crypto, 250 pts)

> Our guy inside e-corp was able to get that packet capture of their backend PKI you asked for. Unfortunately it seems they're using TLS to protect the modulus fetch. Now, I have been told that the best crackers in the world can do this in 60 minutes. Unfortunately I need someone who can do it in 60 seconds.

> Note: Flag does not follow the "Flag:" format but is recognizable

> e_corp_pki.pcapng

We are provided with a packet capture file containing a TLS exchange. Loading it up in Wireshark, we can quickly identify a TLS handshake, with a "server hello" message containing the server certificate, itself containing the RSA public key (modulus + public exponent):



We need to recover the private key by factoring the modulus. We used Fermat's factorisation method to recover the two prime factors (script here):

This quickly yielded values for `p` and `q`. Plugging those values into rsatool, we were able to reconstruct the server's private key:

```
python ./rsatool.py -p 34570934193606833873067800377840532358210931707502119860545125908126852629765481893
```

```
[...]
```

```
Saving PEM as priv.key
```

The next step was to load this private key back into Wireshark to see decrypted TLS traffic. We could see a `GET /modulus` HTTP request with some form of ASCII art:

Nice ASCII art key we thought... But then we looked closely to the first non-zero bytes towards the end: `66 6c 61 67` . This looks like ASCII for "flag"! And indeed:

```
>>> '66:6c:61:67:3a:77:68:65:6e:5f:73:6f:6c:76:69:6e:67:5f:70:72:6f:62:6c:65:6d:73:5f:64:69:67:5f:61:74:5f:
'flag:when_solving_problems_dig_at_the_roots_instead_of_just_hacking_at_the_leaves'
```