AES Trojan (Instructor only)

Team Yellow: Casey Avila, Ben Kunkle, Noah Masten

Trigger

We implemented this paper for the trojan. To summarize, essentially, every time a block is encrypted, the plaintext is checked to see if certain bits are set. If they are, a counter is incremented. In our implementation, the 10th and 92nd bits must be a 1, and the 16th and 107th bit must be a 0. If these conditions are true for 63 blocks in a row, the 64th block will output a different ciphertext, from which the key can be derived. *Figure 3* in the paper shows this in a diagram of AES.

Payload

The payload works by setting the input to AES's 10th internal round to all 1's. This is the last round in AES, each of which uses a subkey which is calculated from the original key. By having an input of all 1's, we can work backwards to find the 10th subkey, and then work backwards from there to calculate each of the previous subkeys, eventually leading to the original key. *Algorithm 1* in the paper describes how to calculate the key from the ciphertext.

Key Reversal Algorithm

Algorithm 1: Reverse Key Schedule

Input: SubKey (K_{10}) of round R_{10}
Output: Original Key (K)

for $i = 10$ **to** 1 **do**
  $[W_i^0, W_i^1, W_i^2, W_i^3] \leftarrow assign(K_i)$;
  $W_{i-1}^3 \leftarrow W_i^3 \oplus W_i^2$;
  $W_{i-1}^2 \leftarrow W_i^2 \oplus W_i^1$;
  $W_{i-1}^1 \leftarrow W_i^1 \oplus W_i^0$;
  $W_{i-1}^0 \leftarrow W_i^0 \oplus f(W_{i-1}^3)$;
end
$K \leftarrow \{W_0^0 \parallel W_0^1 \parallel W_0^2 \parallel W_0^3\}$;
Report K ;
Function $assign\,(K_i)$**:**
  $[k_i^1, k_i^2 \ldots k_i^{16}] \leftarrow K_i$;
  $W_i^0 \leftarrow \{k_i^0 \parallel k_i^1 \parallel k_i^2 \parallel k_i^3\}$;
  $W_i^1 \leftarrow \{k_i^4 \parallel k_i^5 \parallel k_i^6 \parallel k_i^7\}$;
  $W_i^2 \leftarrow \{k_i^8 \parallel k_i^9 \parallel k_i^{10} \parallel k_i^{11}\}$;
  $W_i^3 \leftarrow \{k_i^{12} \parallel k_i^{13} \parallel k_i^{14} \parallel k_i^{15}\}$;
  Return $[W_i^0, W_i^1, W_i^2, W_i^3]$;
Function $f\,(W)$**:**
  $G^0 \leftarrow S(k^{13}) \oplus RC_i$;
  $G^1 \leftarrow S(k^{14})$;
  $G^2 \leftarrow S(k^{15})$;
  $G^3 \leftarrow S(k^{12})$;
  Return $[G^0, G^1, G^2, G^3]$;

Demo

Included is a demo payload for the board, which activates the trojan, as well as our implementation of the key reversal in Python. Don't give this to the other teams!

```
$ python3 ../binary.py -b DEADBEEFDEADBEEFDEADBEEFDEADBEEF payload
334a41cf0608c174faa2ee3a4d14bcf6
334a41cf0608c174faa2ee3a4d14bcf6
⋮
d700d7127b43e90a1590acde3dd829cc  // Trojan activated here! line 64
⋮

$ python3 derive.py d700d7127b43e90a1590acde3dd829cc
deadbeefdeadbeefdeadbeefdeadbeef
```