Simulation of MITM in PEAP with hostap

Siarhei Siniak

December 27, 2016

The introduction.

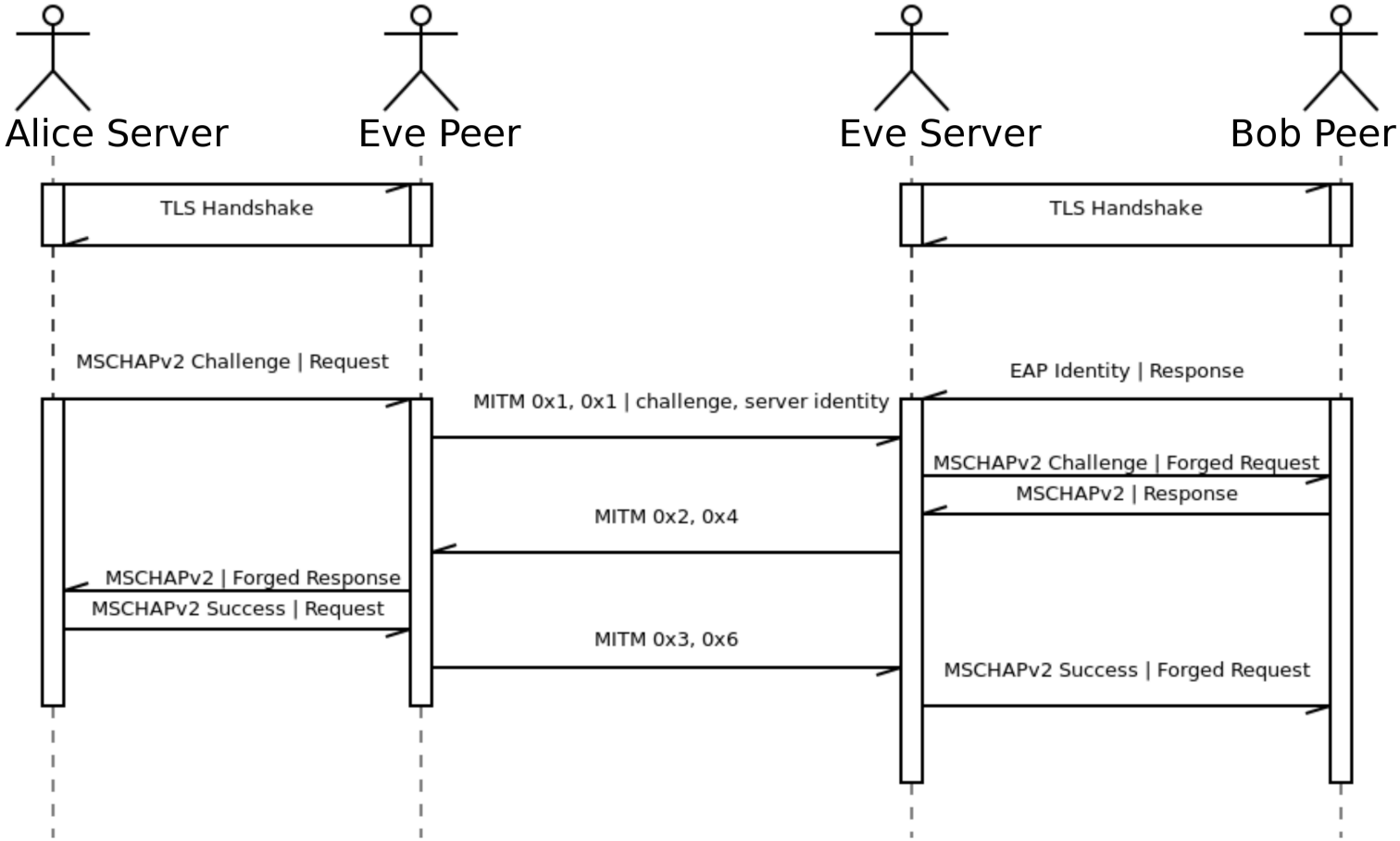
Cryptobindings are important.

PEAP is Protected Extensibl Authentication Protocol.
Before that bare EAP has been used, and it was fine for trusted network tunnel.

We need to simulate the attack.

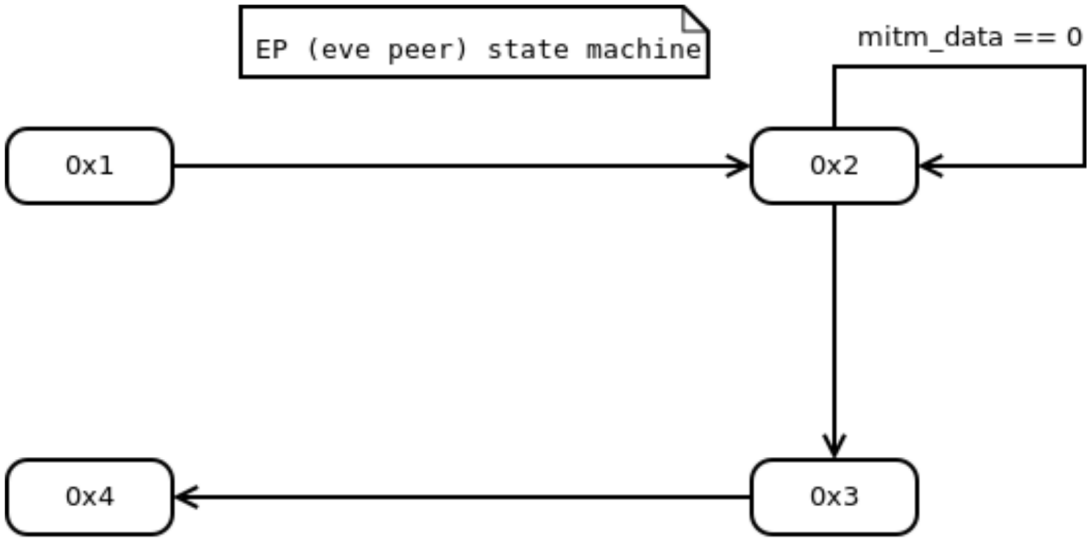
We want to guarantee its correctness.

1. MitM waits for a legitimate device to enter an untunneled legtacy remote authentication protocol and captures the initial messages sent by the legitimate client.
2. MitM initiates a tunneled authentication protocol with an authentication agent.
3. After the tunnel is set up between MitM and the authentication agent, the MitM starts forwarding legitimate client's authentication messages through the tunnel.
4. MitM unwraps the legacy authentication protocol messages received through the tunnel from the authentication agent and fowards them to the legitimate client.
5. After the remote authentication ended successfully, MitM derives the session keys from the same keys it is using for the tunnel.


Alice Server
Eve Peer
Eve Server
Bob Peer
TLS Handshake
TLS Handshake
MSCHAPv2 Challenge | Request
EAP Identity | Response
MITM 0x1, 0x1 | challenge, server identity
MSCHAPv2 Challenge | Forged Request
MSCHAPv2 | Response
MITM 0x2, 0x4
MSCHAPv2 | Forged Response
MSCHAPv2 Success | Request
MITM 0x3, 0x6
MSCHAPv2 Success | Forged Request

It's not that easy to implement in one click the simulation. Especially when the codebase is huge, written in C, and involves few RFC papers. That define the logic and concepts behind.

Both EAP state machines, that are described in RFC 4317, are not easy to modify. It was challenging to find the way to suspend and resume their behaviour on demand.
It happens when one of Eve's machines waits for a missing data from the other one. By default it is not supported. But hostap has pending functionality.
It saves decrypted message and feeds it in again on the next iteration.


EP (eve peer) state machine
mitm_data == 0
0x1
0x2
0x3
0x4

0x1 → 0x2

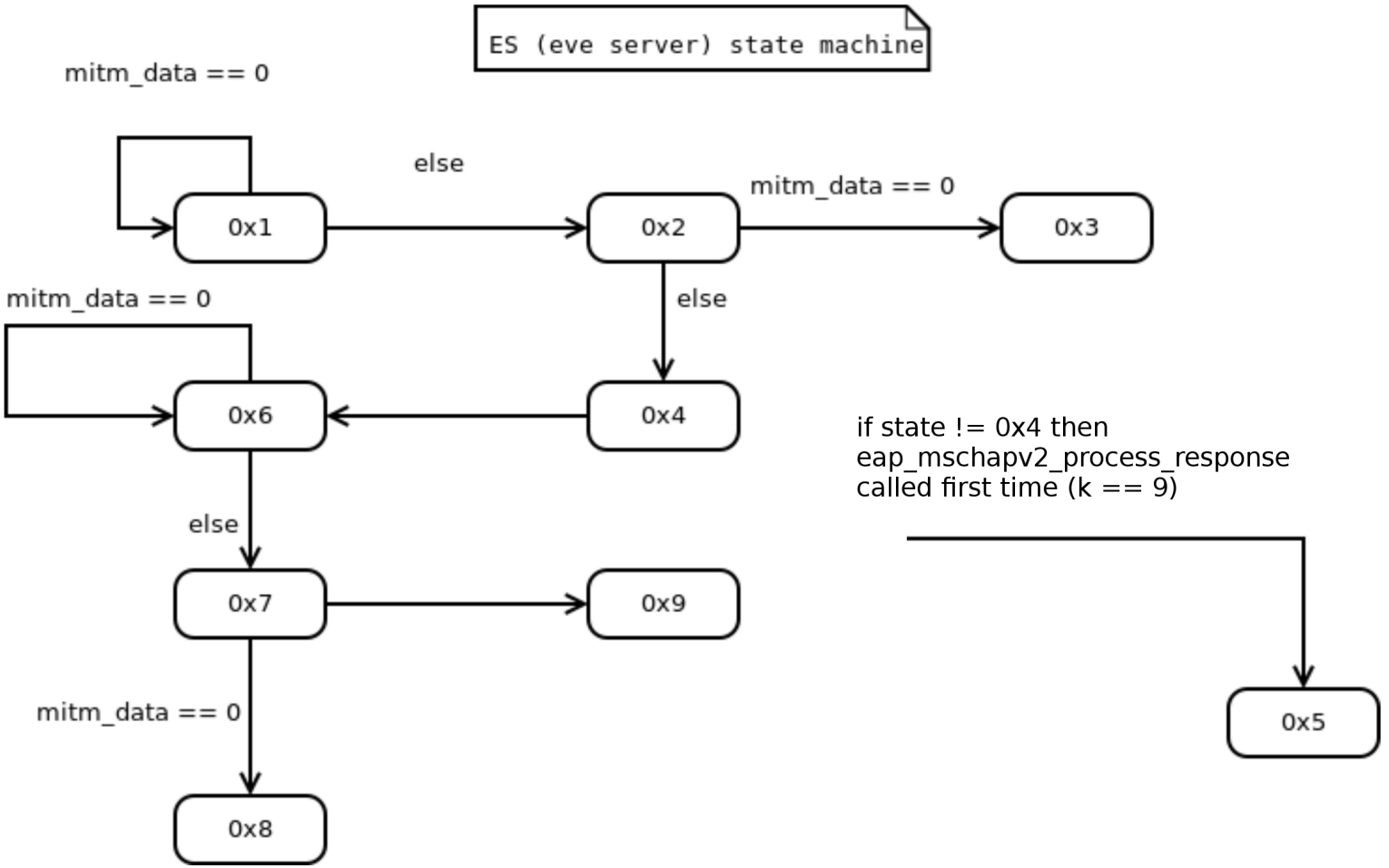
1. Transmit MITM protocol message with MSCHAPv2 Challenge Request from AS(alice server)

0x2 → 0x3

1. Receive MITM protocol message with MSCHAPv2 Challenge Response from BP (bob peer)
2. Build Forged MSCHAPv2 Challenge Response using obtained challenge response

0x3 → 0x4

1. Transmit MITM protocol message with MSCHAPv2 Challenge Response form BP(bob peer)
2. Build MSCHAPv2 Success Response without verification of authenticator response in success request


ES (eve server) state machine
mitm_data == 0
else
mitm_data == 0
0x1
0x2
0x3
mitm_data == 0
else
0x6
0x4
if state != 0x4 then
eap_mschapv2_process_response
called first time (k == 9)
else
0x7
0x9
mitm_data == 0
0x5
0x8

0x1 → 0x2

1. Recieve MITM protocol message: MSCHAPv2 Challenge Request from AS (alice server)

0x2 → 0x3, 0x* → 0x5, 0x7 → 0x8

1. Failure

0x2 → 0x4

1. Build Forged MSCHAPv2 Challenge Request using obtained auth_challenge and server_id

0x4 → 0x6

1. Transmit MITM protocol message with MSCHAPv2 Response from BP(bob peer)

0x6 → 0x7

1. Receive MITM protocol message MSCHAPv2 Success Request from AS (alice server)
2. Skip Challenge Response verification, state = SUCCESS_REQ, master_key_valid=1

0x7 → 0x9

1. Build Forged MSCHAPv2 Success Request using obtained success request

Thanks for attention.