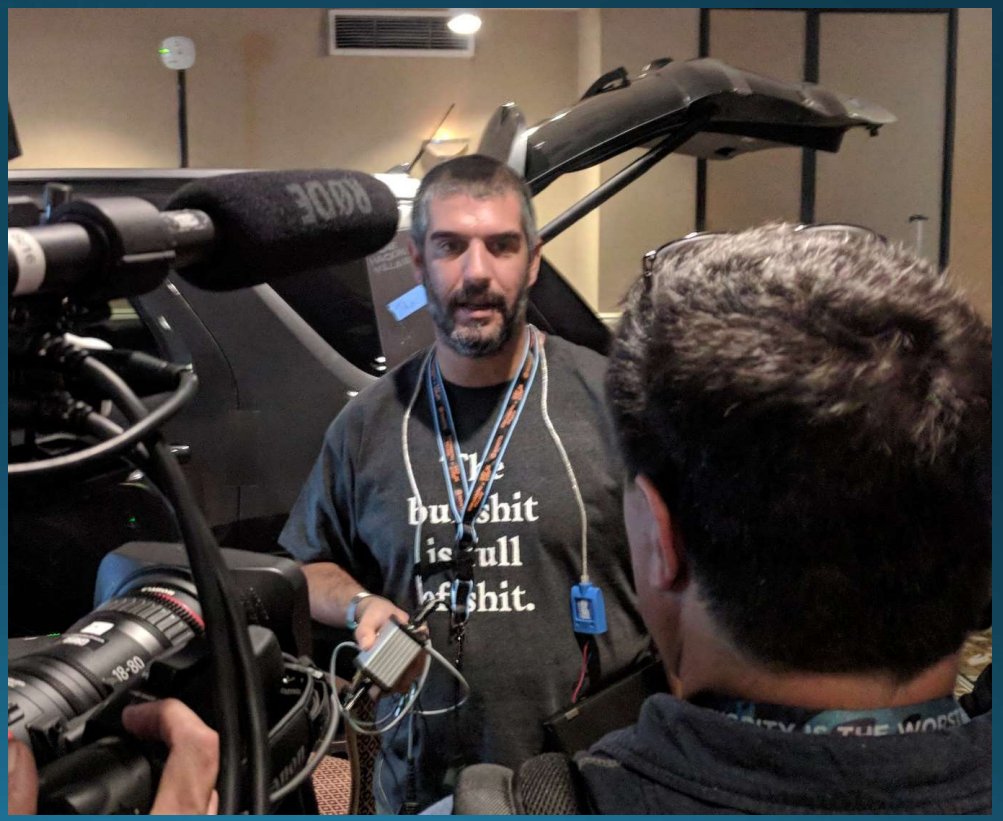
tomotive Diagnostics

Who are we?

- Javier is a Hardware Security Specialist
- He is from Cadiz (Spain)
- Enjoys reversing products that are interesting, or could potentially be more fun
- Likes cake (when it's not a lie) and bbqs



Who are we?

- Ethan is an automotive security engineer
- From the USA
- Works on electrical and mechanical systems

Why are we here?

- Car hacking has become mainstream
- Injection/replay was cool in the 90's
- ECU Security mechanisms are mostly primitive
- Diagnostics are way more simple than they seem

Automotive Diagnostics Protocols

All the Diagnostics Protocols used in Automotive have something in common:

- Based on request/response scheme.
- Tester is a client, DUT is a server.
- Certain functionality requires authentication.

Security Access (0x27)

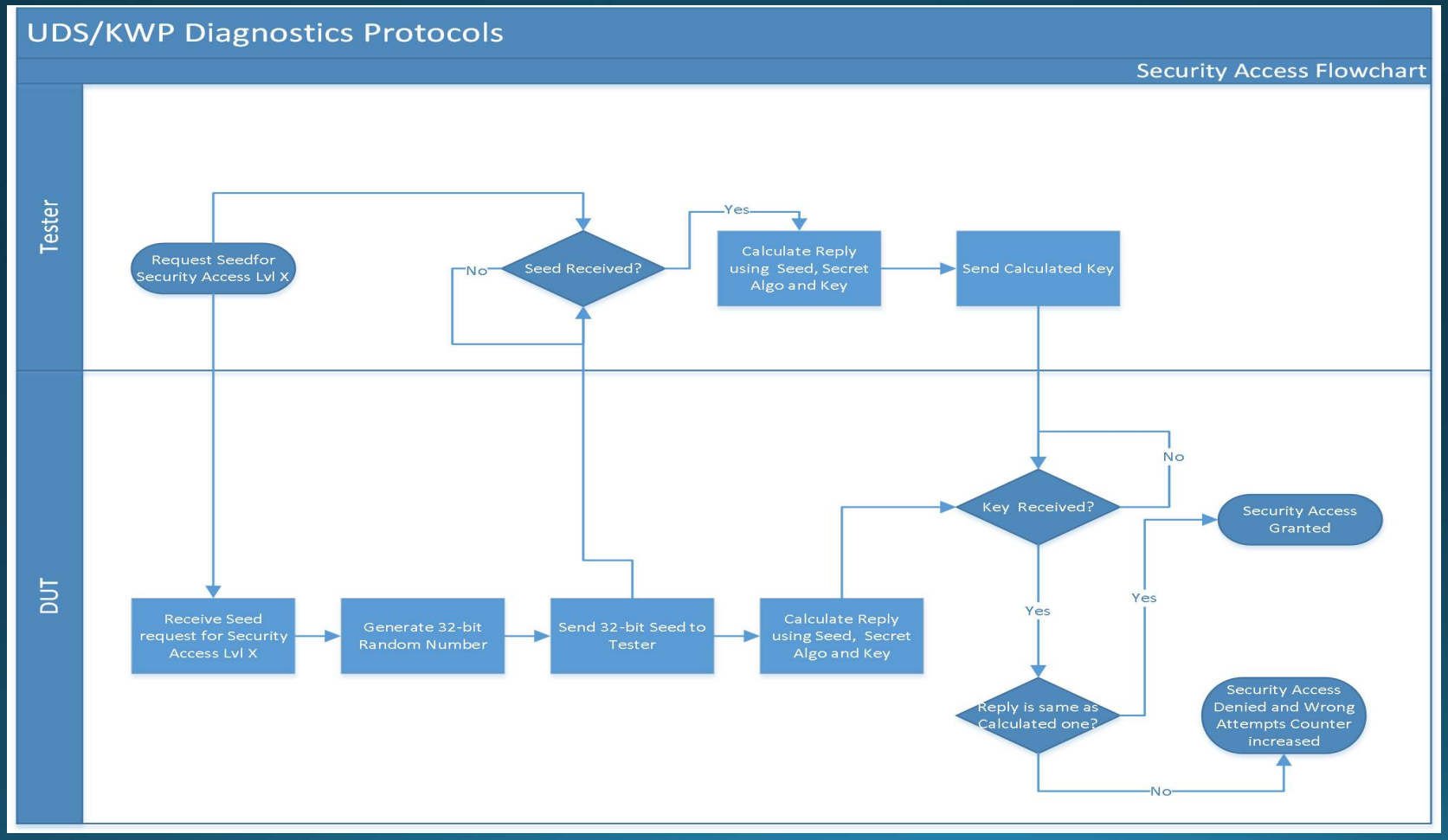
This SID (Service ID) is commonly used to restrict access to functionality that requires protection, such as:

- Read/Write Flash/Calibration data.
- Write parameters (such as VIN)
- Perform operations that are restricted to the manufacturer.

It is divided in two separate stages. One to request the seed, and one to provide the reply.

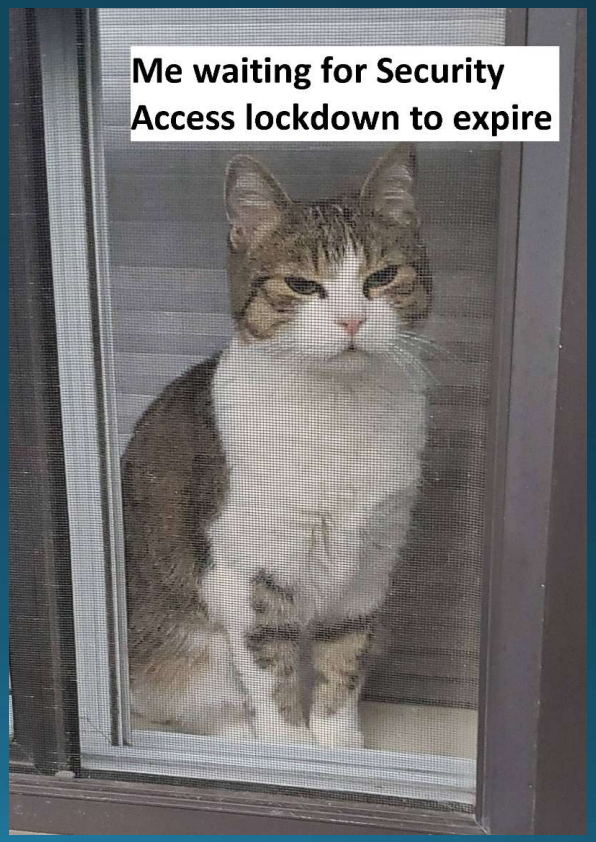
There is even a market for selling/buying Security Access algorithms/keys from manufacturers.

Security Access – How does it work?



Security Access – Additional Information

- Different Security levels usually require different algorithm/key.
- Some Security Access levels require being in a specific Diagnostics Session (0x10) type.
- After X wrong attempts (typically 3), Security Access will be blocked for a time defined by the manufacturer (typically 1-30 minutes). This discourages brute-force attacks.



What is the logic behind Security Access?

```
uint8_t data[8];
uint8_t seed[4];
bool securityAccessRequest(uint8_t data) //data contains the data section of the ISO-TP frame
{
        uint8_t level = data[1]; //the requested level is in the second byte of the data array
        if(isblocked(level) == true)//verifies that the requested security access level is not blocked
        {
                return false; //if the level is currently blocked due to too many wrong attempts, return false
        }
        uint8_t seed[4];//array that contains the seed
        for(uint8_t i=0;i<4;i++)
                seed[i]=rand()%255; //Generate number between 0 to 255
        sendSeed(seed); //sends the seed to the tester
        return true; //all good, request was addressed
}
```

What is the logic behind Security Access?

```
uint8_t seed[4]; //at this point, it contains the seed that was sent previously
bool securityAccessVerify(uint8_t *data) //data contains the data section of the ISO-TP frame
{
        processKey(seed); //calculates the correct reply and stores it back in the seed array (or a dedicated one)
        if(memcmp(data + 2, seed,4)  == 0) //if the seed provided by the tester is the same one that was calculated
        {
                return true; //all good to go
        }
        incrementWrongAttempts(data[1] - 1); //increment the wrong attempts counter for that level
        return false; //and report back the failure
}
```

The Hacker kitty noticed something… Did you notice it too?

Using MITM attacks on Security Access

Man-in-the-middle attacks are not new. Just like CAN traffic injection, they were pretty cool in the 90's.

Considering that both KWP2K UDS were developed in the same decade one would think that this kind of attack was accounted for.

The answer is nope.

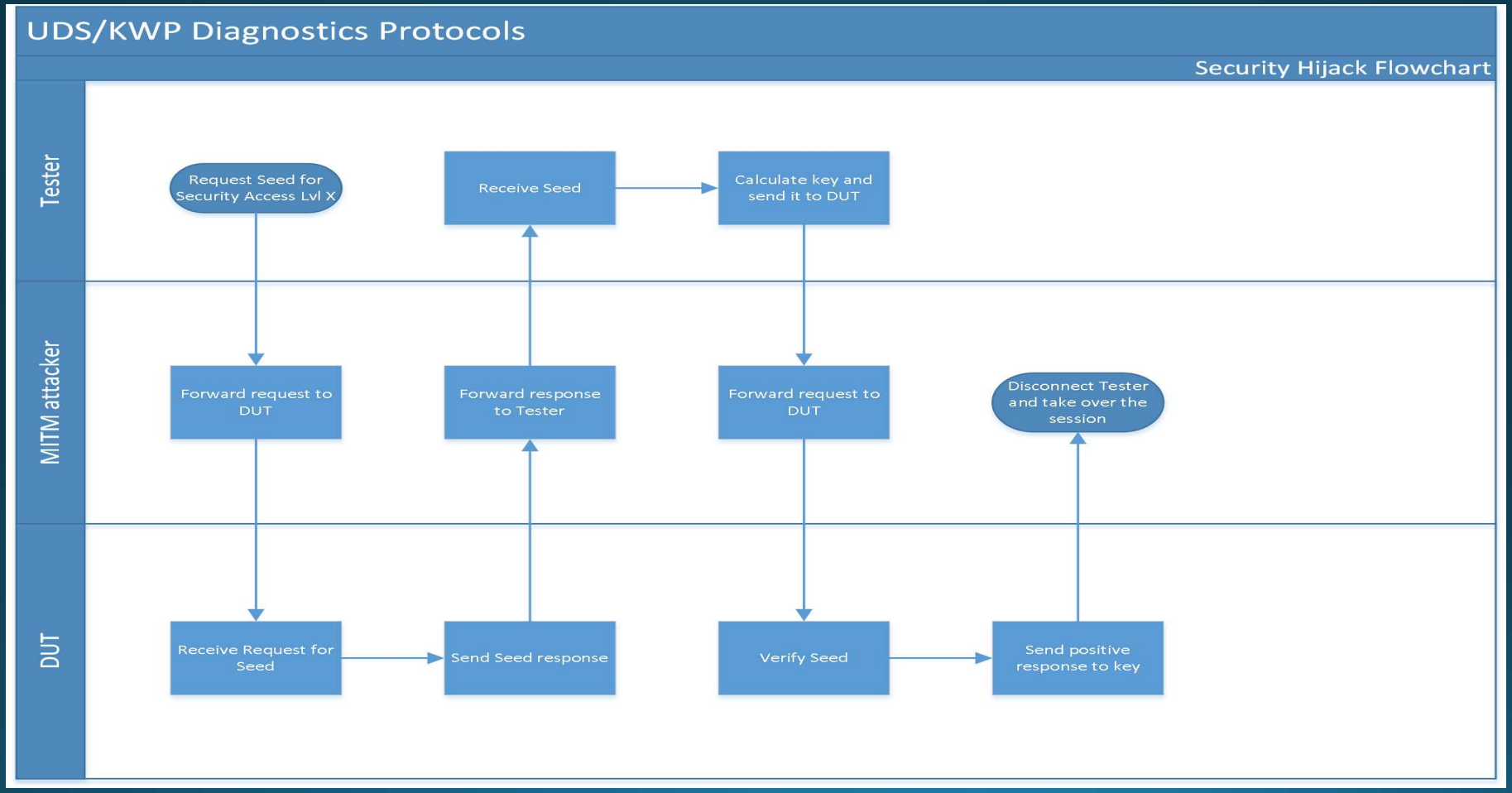
Security Hijack

Initially released in 2016 at BlackHat and Defcon 24, many ECUs that are being manufactured today are still vulnerable to it.

The attack has three stages:

- Forward all traffic until a successful security access happens.
- Disconnect the tester when security access auth is successful.
- Take over the now authenticated diagnostics session.

Security Hijack



Security Hijack – Pros and Cons

Pros:

- An attacker does not need to know the security access algorithm or key to obtain the authorization.
- The operation is easily repeatable with consistent time to perform it.

Cons:

- The attacker needs constant access to a tester that performs the specific authorization that he needs.

What is SecHammer?

Security Hammer exploits poor implementation practices in the state machine logic used for diagnostics.

By not sending a reply to the seed, there is no “wrong attempt” flagged.

By using a combination of sequences, DUTs can be forced in most cases to randomize the seed, even if they don’t upon re-requesting it.

This means that we can now gather information about the Seed Generation randomness.

SecHammer Example

Line	Time (abs/rel)	Tx	Er	Description	ArbId/Header	Len	DataBytes	Network	Node	ChangeCnt	Timestamp
					x18da40f1,x1...						
1				HS CAN $18DA40F1	x18DA40F1	3	02 10 02	HS CAN			2022/05/11 13:32:59:023922
2	44.243 ms			HS CAN $18DAF140	x18DAF140	4	03 7F 10 78	HS CAN			2022/05/11 13:32:59:068164
3	95.188 ms			HS CAN $18DAF140	x18DAF140	8	03 7F 10 78 00 00 00 00	HS CAN			2022/05/11 13:32:59:163353
4	4.366 ms			HS CAN $18DAF140	x18DAF140	8	06 50 02 00 32 01 F4 00	HS CAN			2022/05/11 13:32:59:167719
5	28.772 ms			HS CAN $18DA40F1	x18DA40F1	3	02 27 01	HS CAN			2022/05/11 13:32:59:196491
6	1.033 ms			HS CAN $18DAF140	x18DAF140	8	06 67 01 6F E6 33 23 00	HS CAN			2022/05/11 13:32:59:197524
7	28.999 ms			HS CAN $18DA40F1	x18DA40F1	3	02 27 01	HS CAN			2022/05/11 13:32:59:226523
8	975 µs			HS CAN $18DAF140	x18DAF140	8	06 67 01 A2 CC E7 5F 00	HS CAN			2022/05/11 13:32:59:227499
9	29.057 ms			HS CAN $18DA40F1	x18DA40F1	3	02 27 01	HS CAN			2022/05/11 13:32:59:256555
10	915 µs			HS CAN $18DAF140	x18DAF140	8	06 67 01 69 97 EC 22 00	HS CAN			2022/05/11 13:32:59:257471
11	29.116 ms			HS CAN $18DA40F1	x18DA40F1	3	02 27 01	HS CAN			2022/05/11 13:32:59:286587
12	855 µs			HS CAN $18DAF140	x18DAF140	8	06 67 01 66 44 E5 24 00	HS CAN			2022/05/11 13:32:59:287442
13	29.176 ms			HS CAN $18DA40F1	x18DA40F1	3	02 27 01	HS CAN			2022/05/11 13:32:59:316619
14	801 µs			HS CAN $18DAF140	x18DAF140	8	06 67 01 27 15 E0 BE 00	HS CAN			2022/05/11 13:32:59:317420
15	29.229 ms			HS CAN $18DA40F1	x18DA40F1	3	02 27 01	HS CAN			2022/05/11 13:32:59:346649
16	742 µs			HS CAN $18DAF140	x18DAF140	8	06 67 01 A2 5A 11 D9 00	HS CAN			2022/05/11 13:32:59:347390
17	29.290 ms			HS CAN $18DA40F1	x18DA40F1	3	02 27 01	HS CAN			2022/05/11 13:32:59:376680
18	683 µs			HS CAN $18DAF140	x18DAF140	8	06 67 01 42 44 56 C4 00	HS CAN			2022/05/11 13:32:59:377364
19	29.348 ms			HS CAN $18DA40F1	x18DA40F1	3	02 27 01	HS CAN			2022/05/11 13:32:59:406712
20	626 µs			HS CAN $18DAF140	x18DAF140	8	06 67 01 BA 74 05 7B 00	HS CAN			2022/05/11 13:32:59:407337
21	29.406 ms			HS CAN $18DA40F1	x18DA40F1	3	02 27 01	HS CAN			2022/05/11 13:32:59:436743
22	566 µs			HS CAN $18DAF140	x18DAF140	8	06 67 01 D9 8B 9D 57 00	HS CAN			2022/05/11 13:32:59:437308
23	29.464 ms			HS CAN $18DA40F1	x18DA40F1	3	02 27 01	HS CAN			2022/05/11 13:32:59:466773
24	516 µs			HS CAN $18DAF140	x18DAF140	8	06 67 01 3F 0F 6C 85 00	HS CAN			2022/05/11 13:32:59:467288

Does it always work?

From more than 40 ECUs that have been tested, 38 were vulnerable to this attack. This includes models manufactured in 2019.

One of them was not considered vulnerable because it was always providing the same seed…

Some ECUs will provide the same seed if requested consecutively, so some tricks have to be done, such as:

- Switching to a different Diagnostics Session type before requesting it again.
- Requesting a seed for a different level and then re-requesting the targeted one.
- Terminating and restarting the Diagnostics Session before requesting a new seed.

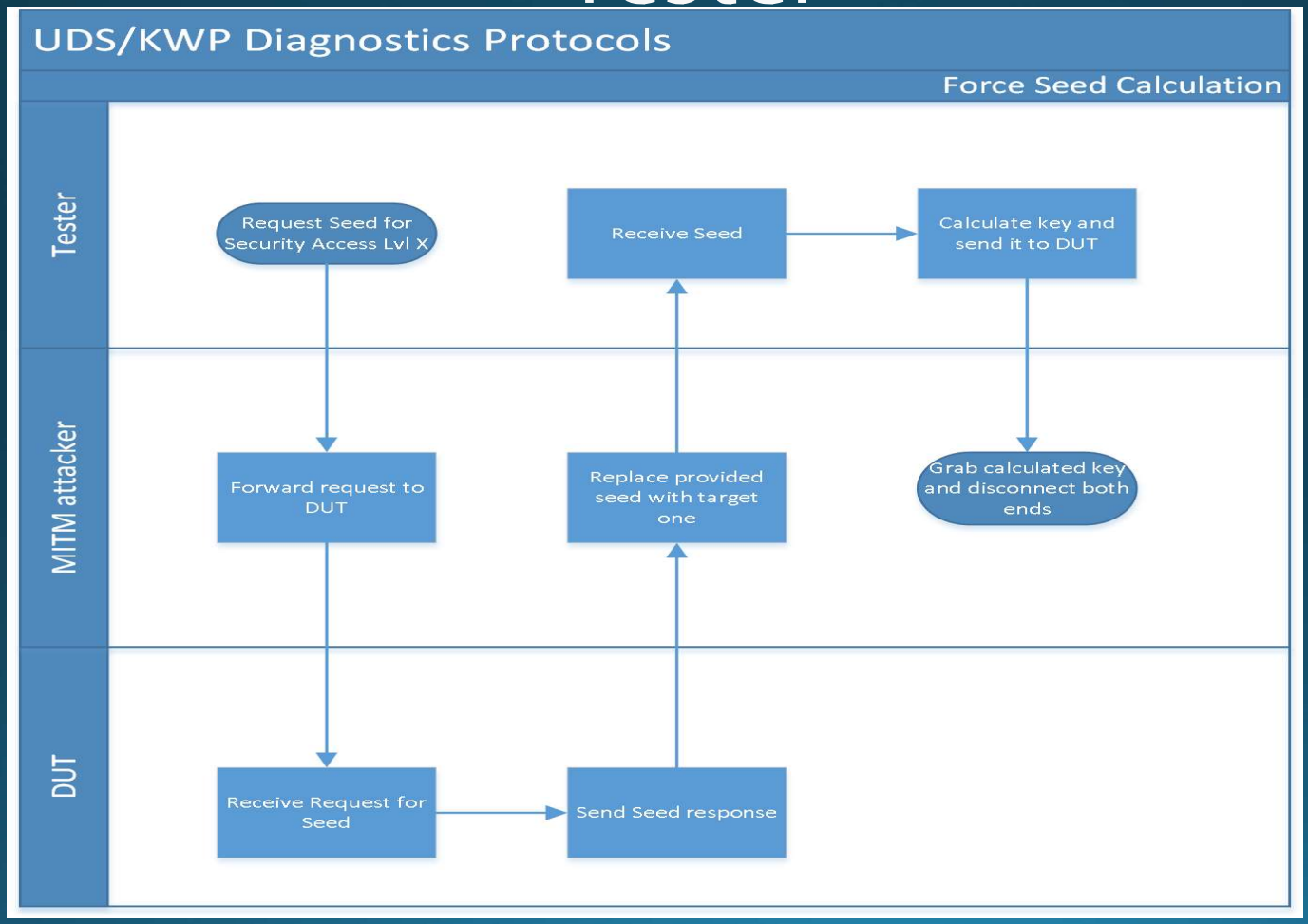
What is SecPuppet?

Security Puppet uses a combination of both MITM and SecHammer to force a specific Seed with a precalculated response.

This attack is divided in three parts:

- Randomness analysis of seeds using SecHammer
- MITM attack to force a specific seed calculation on the tester
- SecHammer attack to force a specific seed on the DUT

SecPuppet – Forcing a seed on the Tester



SecPuppet – Pros and Cons

Pros:

- An attacker does not need to know the security access algorithm or key to obtain the authorization.
- Requires a one-time access to test equipment

Cons:

- The target needs to be vulnerable to SecHammer
- The target needs to have poor or predictable randomness in the seeds provided

Use cases for these attacks

- Testing ECU randomness source for Seed generation
- Reading/Writing memory/flash offsets that are “off limits” for non-OEM tools.
- Performing operations that are not supported by the tester.

Thanks for attending!

Should you have any questions, please reach out!

@fjvva

javier.vazquez.vidal@gmail.com

ethan@elgansodorado.com