# Danger of Client Side Controls - E2E Encryption (E2EE)

Anton Bolshakov

London 2018, May

# About Me

Anton Bolshakov (aka blshkv)

- In security business since 1999
- Penetration Testing (PT) of: Network, Wireless, Radio, Mobile, Application, VoIP and forensics
- Industries: Financial, Medical, Transportation, Oil&gas in Apac region
- Developer of Pentoo Linux
- Current: ITDefence.asia

# Topic

- Types of clients side controls
- Penetration Testing of End-to-end encryption (E2EE) enabled applications
- Methods of implementations and ways to bypass them

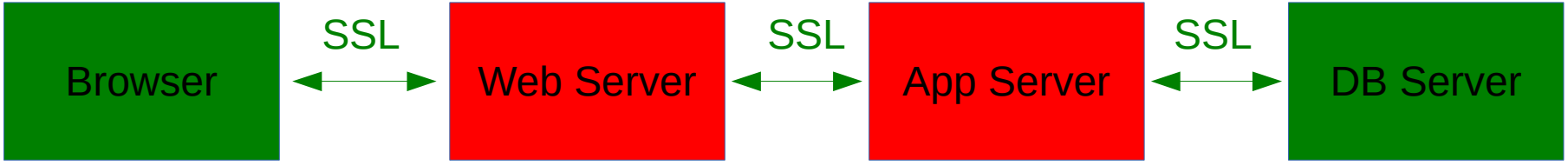
# Types of Client Side Controls

- Mobile jailbreak detection (i.e. “administrator” access on a desktop)
- Mobile SSL Certificate Pinning (mobile only)
- HPKP HTTP Public Key pinning (getting abandoned, 1 May 2018)
- MDM solutions (have been bypassed regardless of vendor)
- Endpoint security solutions (client agent break)
- Proprietary (binary) protocols (reverse engineering)
- E2E encryption

# What is E2E Encryption

Wikipedia:

- “End-to-end encryption (E2EE) is a system of communication where only the communicating users can read the messages”
- “The systems are designed to defeat any attempts at surveillance or tampering because no third parties can decipher the data being communicated or stored”



# Regulations

## Monetary Authority of Singapore (MAS)

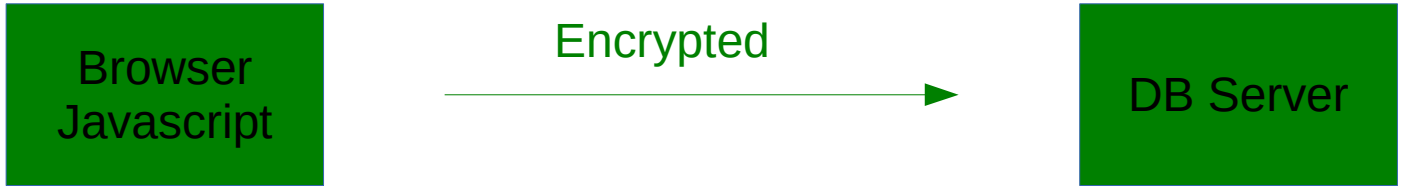
| | | | | | | |
|---|---|---|---|---|---|---|
| 20. | 4.1.5 | Encryption of customer PINs and other sensitive data is maintained end-to-end at the application layer. The encryption process is kept intact from the point of data entry to the final system destination where decryption and/or authentication takes place. | ☐ | ☐ | ☐ | ☐ |
| 78. | 5.2.1 (x) | Deploy strong cryptography and end-to-end application layer encryption to protect customer PINs, user passwords and other sensitive data in networks and in storage. | ☐ | ☐ | ☐ | ☐ |

- 200x – Java, non-standard crypto libraries
- 201x – Javascript, non-standard crypto libraries

**0-day stops here**

# Use case 1: Idea

Encrypt with Javascript on a client side:




**0-day stops here**

# Use case 1: Implementation

- Self-written crypto Javascript library with hardcoded passwords:

```
function Encrypt(plainText)
{
        var iv = "F27D5C9927726BCEFE7510B1BDD3D137";
        var salt = "3FF2EC019C627B945225DEBAD71A01B6985FE84C95A70EB132882F88C0A59A55";
        var keySize = 128;
        var iterationCount = 10000;
        var passPhrase = "ResetPwdOtpModule";
        var aesUtil = new AesUtil(keySize, iterationCount);
        var encrypt = aesUtil.encrypt(salt, iv, passPhrase, plainText);

        return encrypt;
}
```

**0-day stops here**

# Use case 1: Hack

- Use existing AES burp plugin or decrypt manually
- Initially: Login form to encrypt username/password
- Later: All forms with sensitive data
- Change Password Form encrypts **username**/password too!

# Use case 1: Hack

- Ability to change any user's password:

**Request**

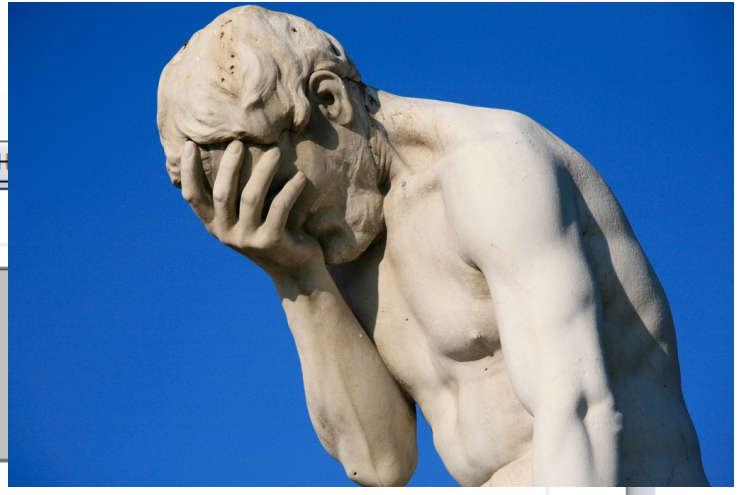
Raw | Params | Headers | Hex

```
POST /ResetPassword/unlock/callRSAService HTTP/1.1
Host: 10.22.8.213:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101
Firefox/52.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.22.8.213:8080/ResetPassword/unlock/forgetPage
Cookie: JSESSIONID=48FEA1B2C87B6F191BB88F0B75F5022B
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 163

userame=3pWptlMqAHKkSgqdR2b%252BLQ%253D%253D&newPassw
ord=QhcPWgUs6Zwr3e2c9AAjPyioVDPtW6weTHENiFm1EHw%253D&pa
ssword=QhcPWgUs6Zwr3e2c9AAjPyioVDPtW6weTHENiFm1EHw%253D
```

**Response**

Raw | Headers | H



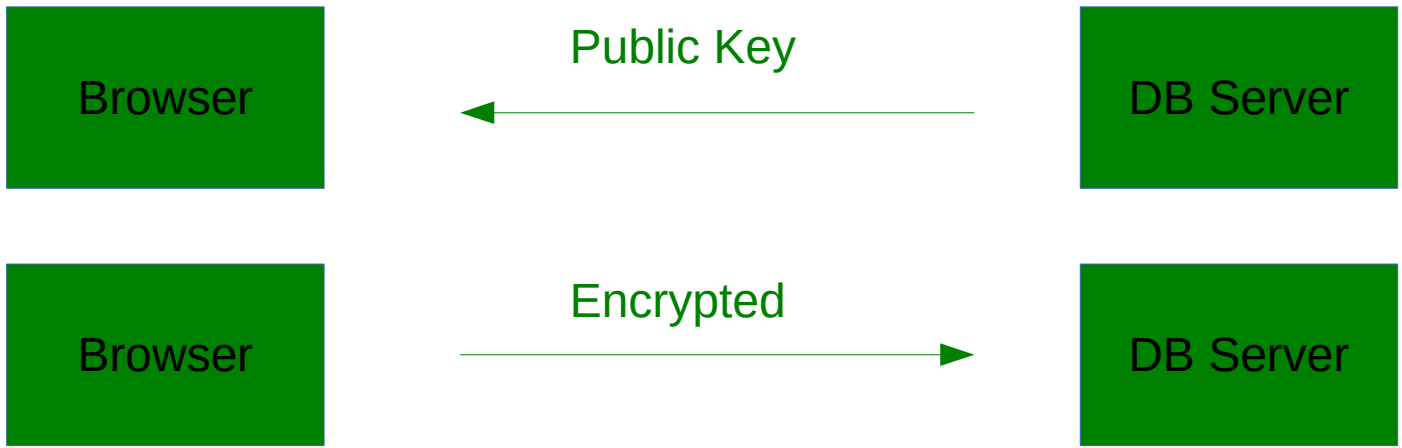
**Password Assistance**

**Your password will be changed and synchronized in less than 60 seconds**

**0-day stops here**

# Use case 2: Idea

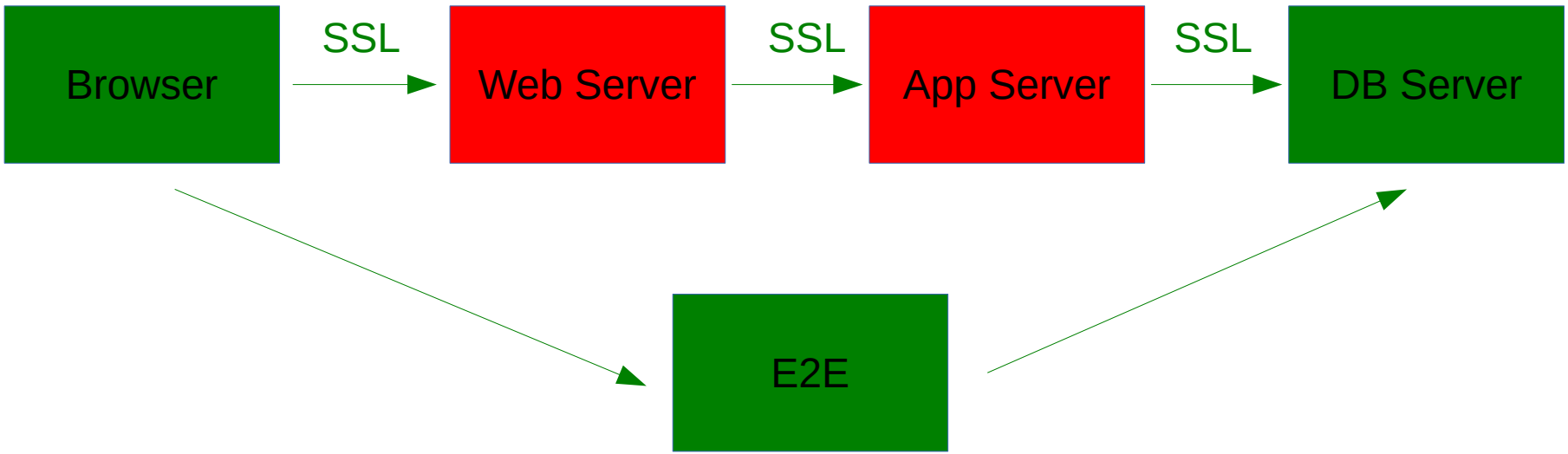
Push PKI public key:




Data encrypted with an unsupported RSA library

# Use case 2: Expected Flow

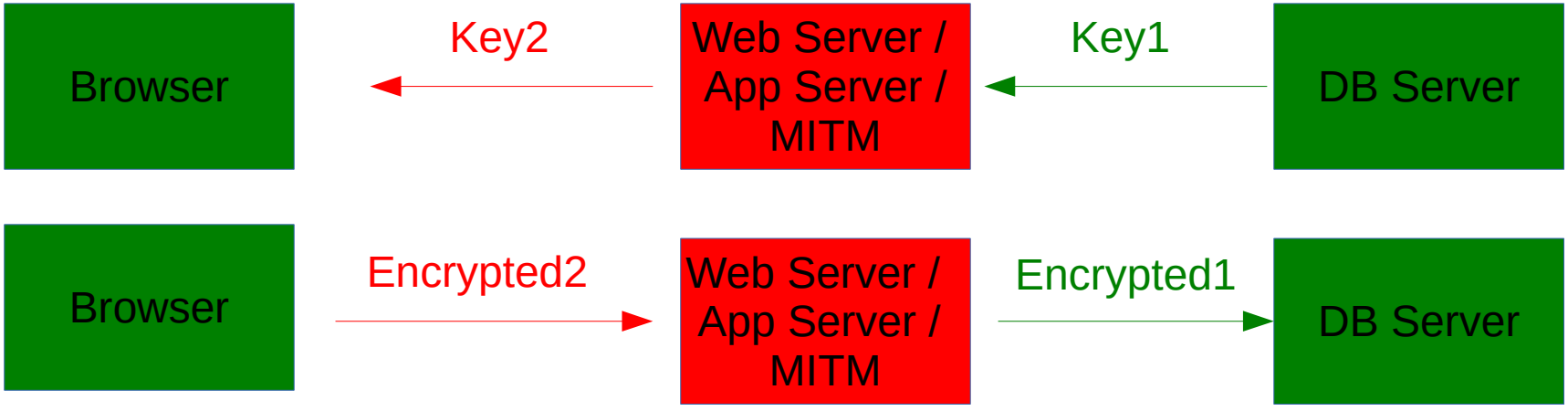
- Encrypted communication between untrusted termination points




**0-day stops here**

# Use case 2: Attack

- Classic MITM Attack: Change the key




**0-day stops here**

# MITMA on E2E Encryption

Wikipedia:

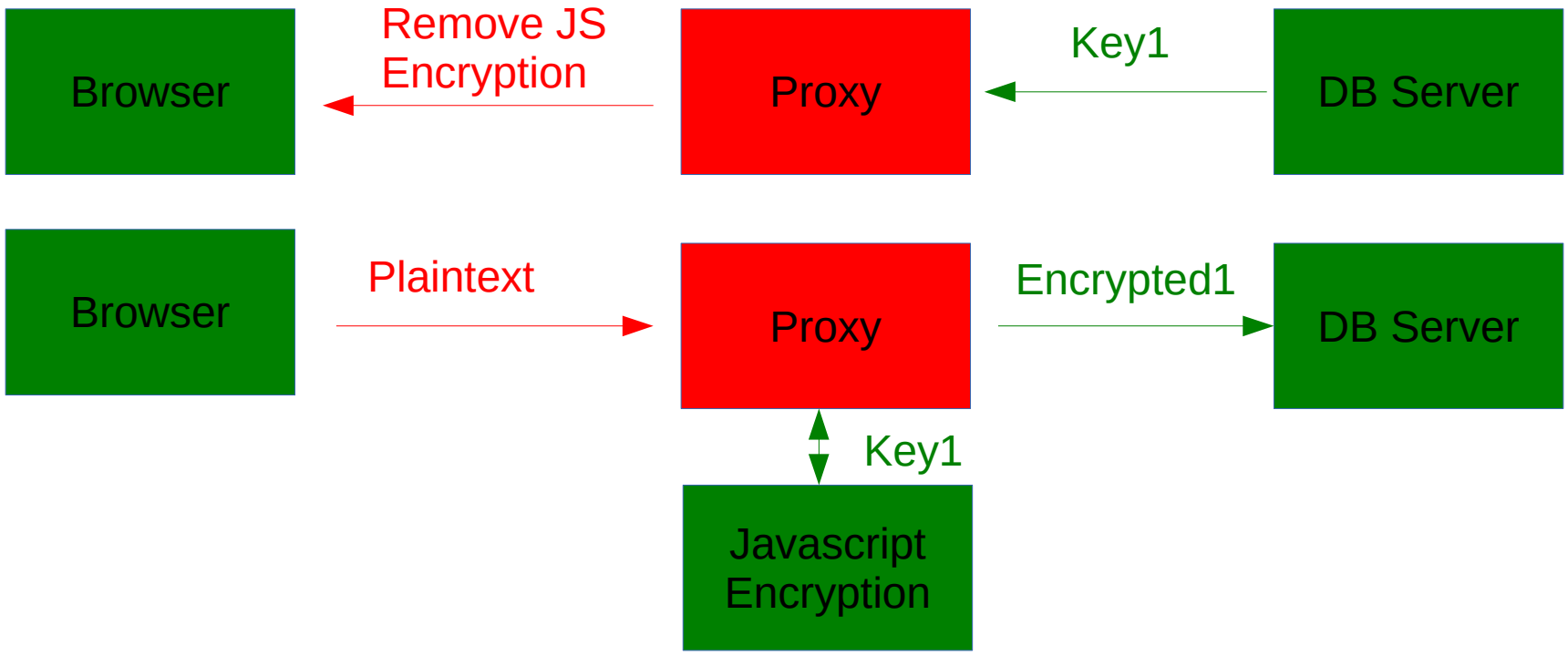
“A man-in-the-middle attacker may impersonate a message recipient so that messages are encrypted with a key known to the attacker”

**0-day stops here**

# Plugin Implementations

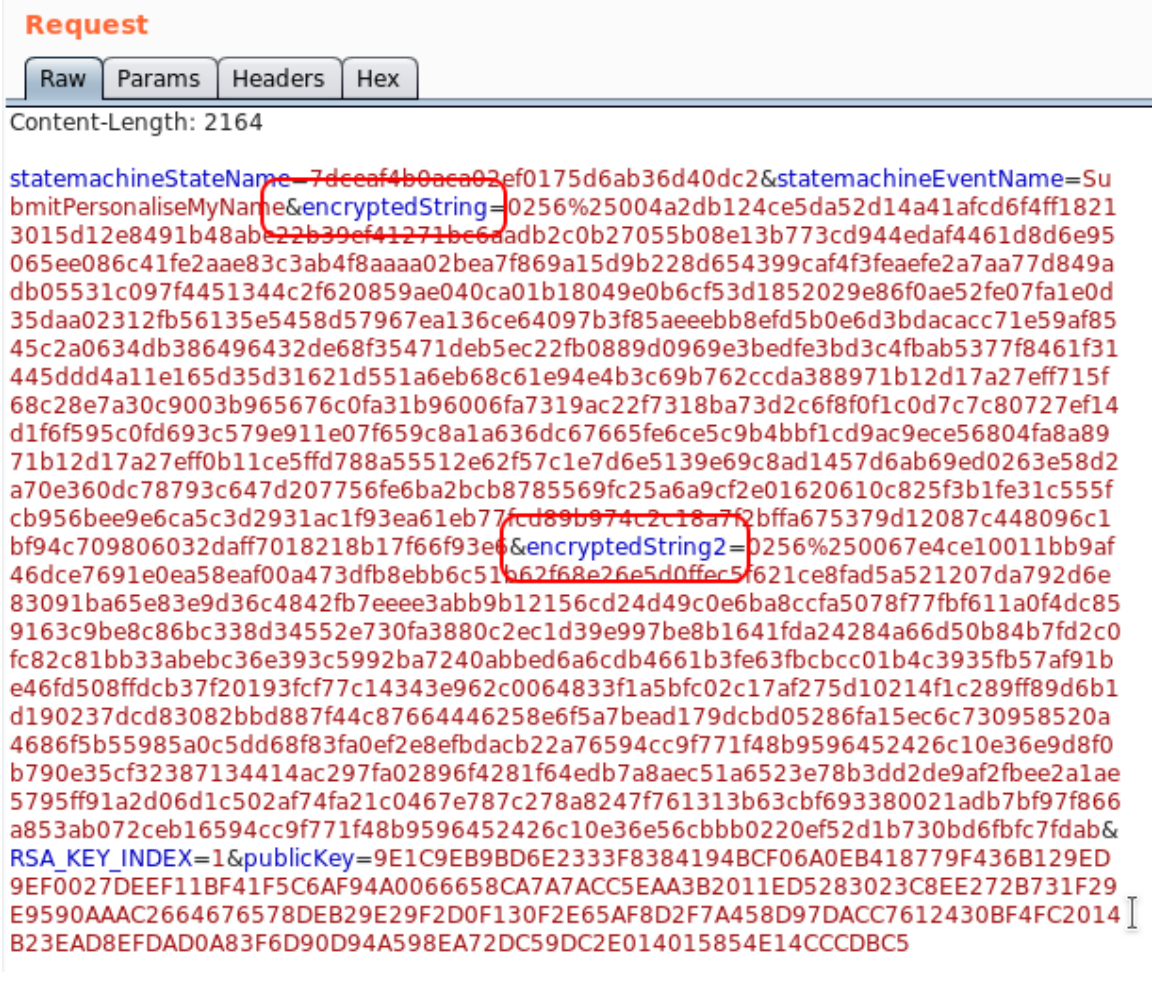
1. Classic MITM Attack: Public Key Replacement (Brida by Federico Dotta):
   - Cons: Decrypt with Key2, Encrypt with Key1
   - Cons: Do not support customised parameters order and encoding
2. Remove Encryption on a client side (Posponer by ITDefence):
   - Pros: Encrypt only once;
   - Pros: Call encryption library without any modifications
   - Cons: Harder to modify a mobile binary client

# Burp: Plugin Implementation (Java)



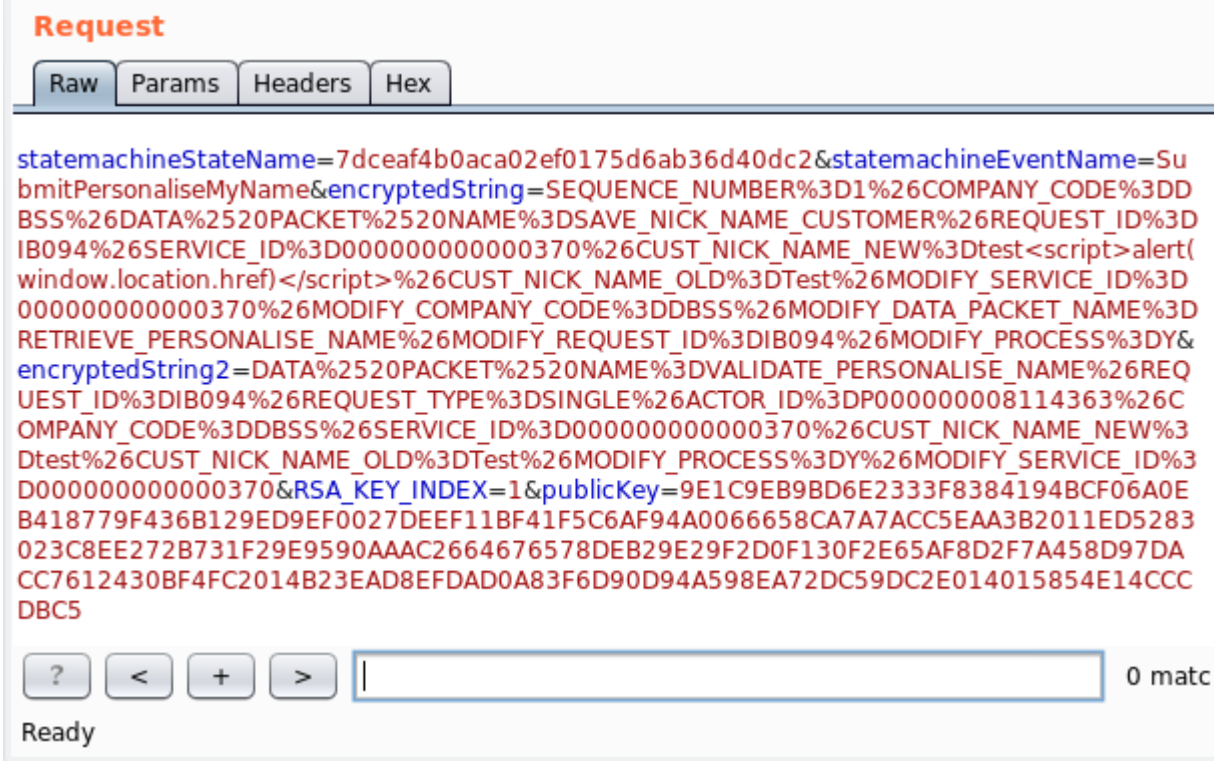

Java has embedded Javascript engine!

# Plugin: Before




**0-day stops here**

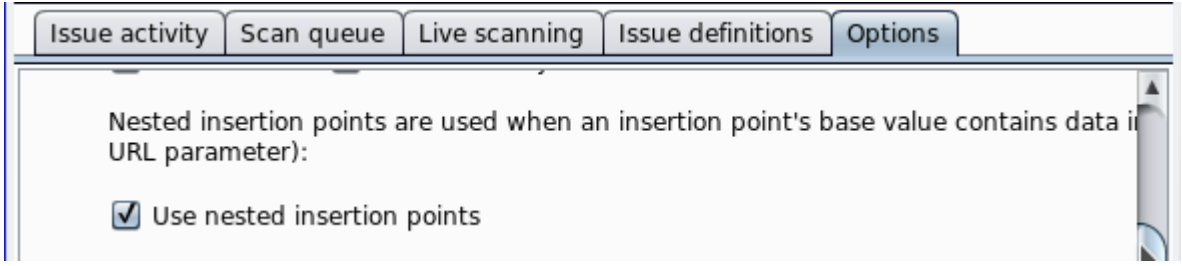
# Plugin: After




**0-day stops here**

# Nested parameters

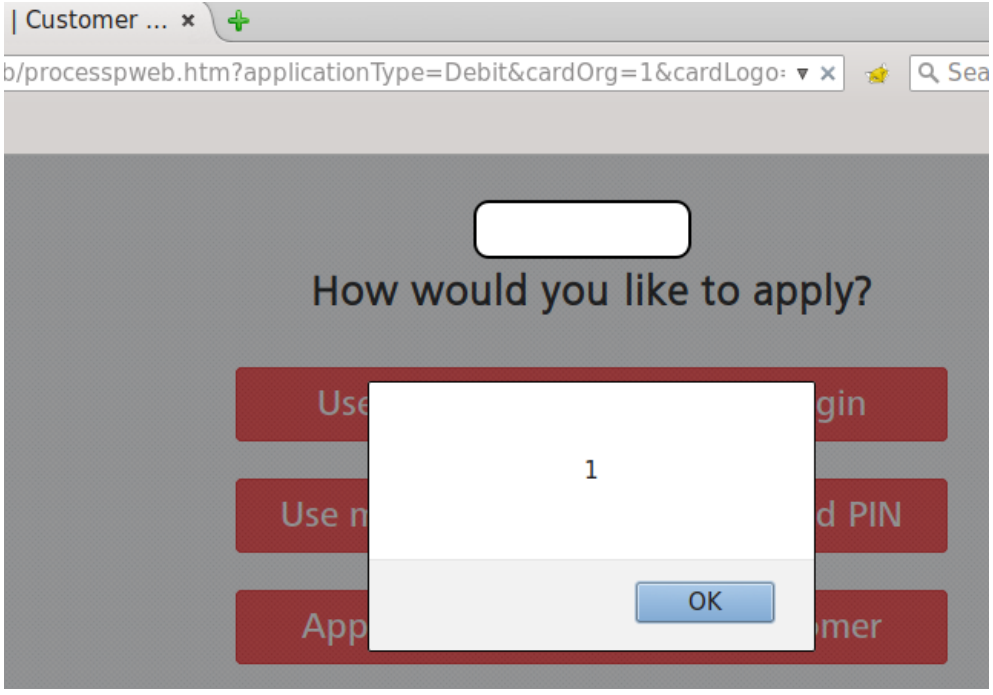
- EncryptedString: encoded(VAR1=VALUE1&VAR2=VALUE2)




- JSON: encryptedString: pollerRequest= [{"Key":420918415693, "cId":"10157"}])
- Other formats might not be supported

# Plugin: Hack

- Multiple Stored XSS
- Input validation was done against the “encryptedString2” parameter only




**0-day stops here**

# Other E2EE Implementations

- Change Public Key with each request
- 7 Public Keys (for each day) to encrypted an AES encrypted block with a random password generated on a client side
- Custom Made Encryption
- Public Key hardcoded in a mobile client (use Brida)

# Summary

- Clients Side Controls can be bypassed (once again)
- E2E Encryption can lead to a false sense of security
- E2EE restricts penetration testers, not hackers

Q & A Time

anton.bolshakov@itdefence.asia
https://github.com/ITDefence