

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

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

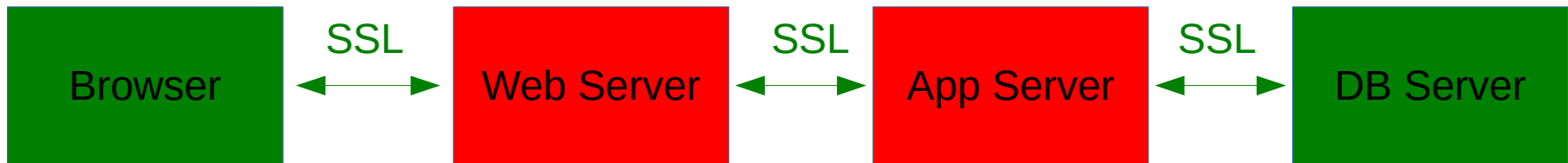
Types of Client Side Controls

- 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.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
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.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

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

Use case 1: Idea

Encrypt with Javascript on a client side:



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;
}
```


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

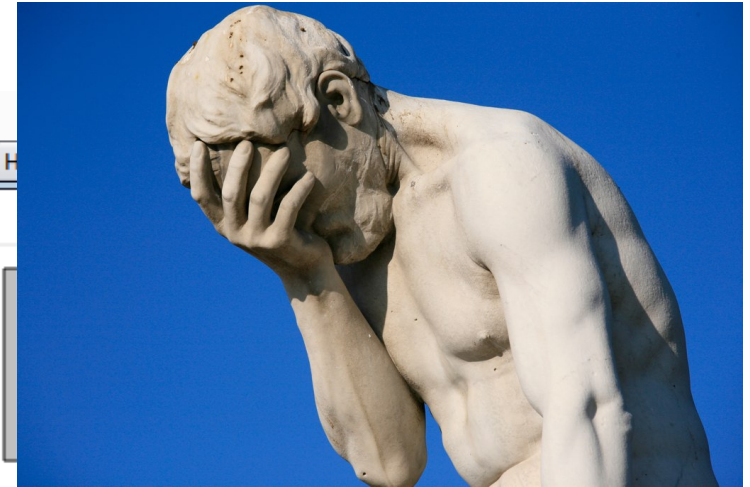
Raw Params Headers Hex

```
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
```

```
username=3pWptlMqAHKkSgqdR2b%252BLQ%253D%253D&newPassw
ord=QhcPWgUs6Zwr3e2c9AAjPyioVDptW6weTHENiFm1EHw%253D&pa
ssword=QhcPWgUs6Zwr3e2c9AAjPyioVDptW6weTHENiFm1EHw%253D
```

Response

Raw Headers H



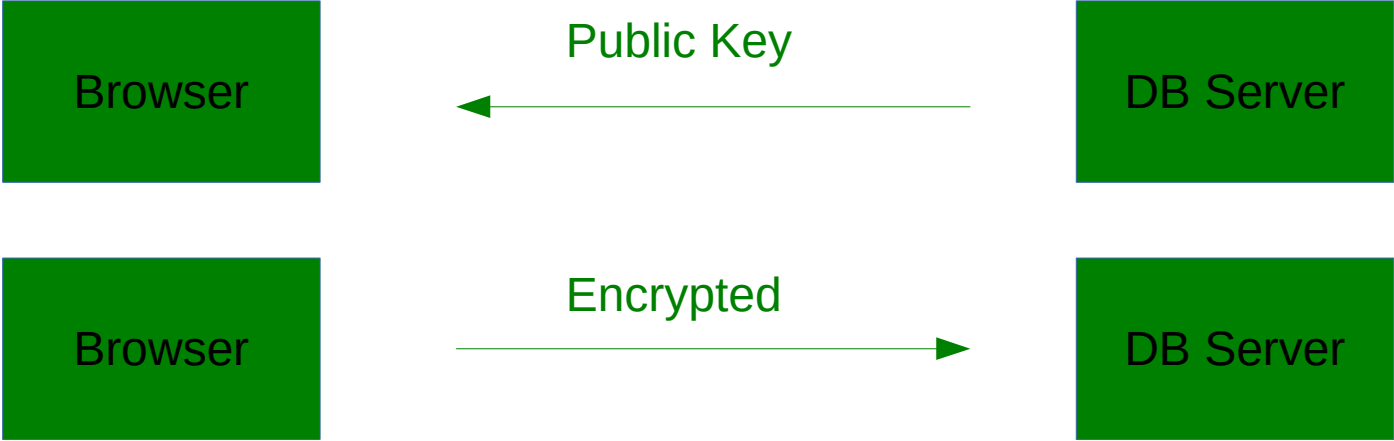
Password Assistance

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

0-day stops here

Use case 2: Idea

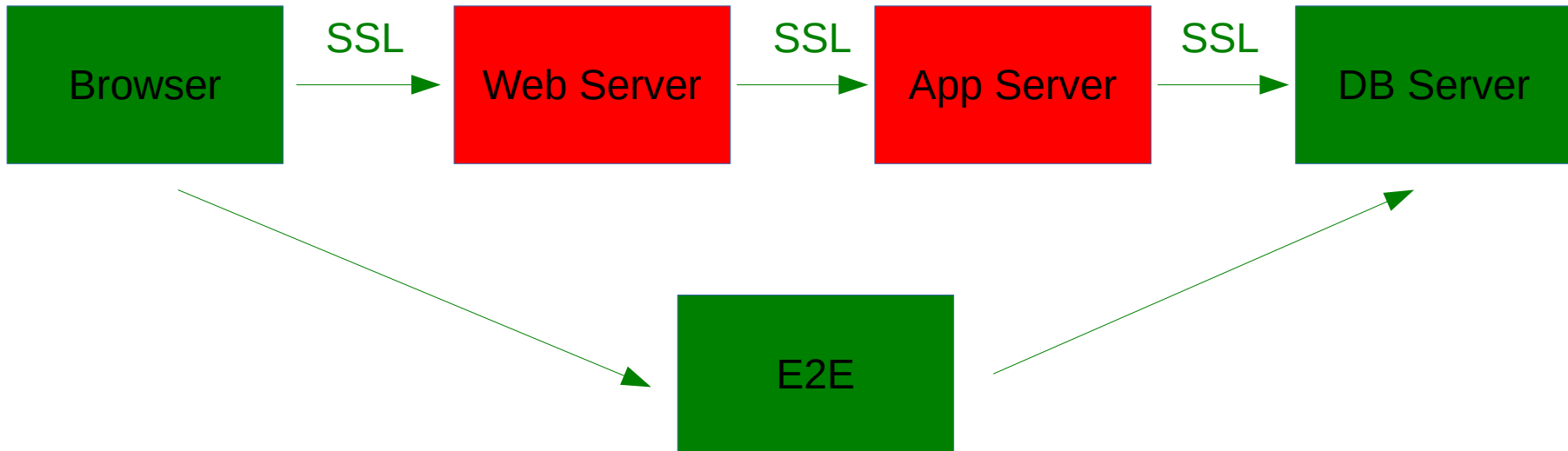
Push PKI public key:



Data encrypted with an unsupported RSA library

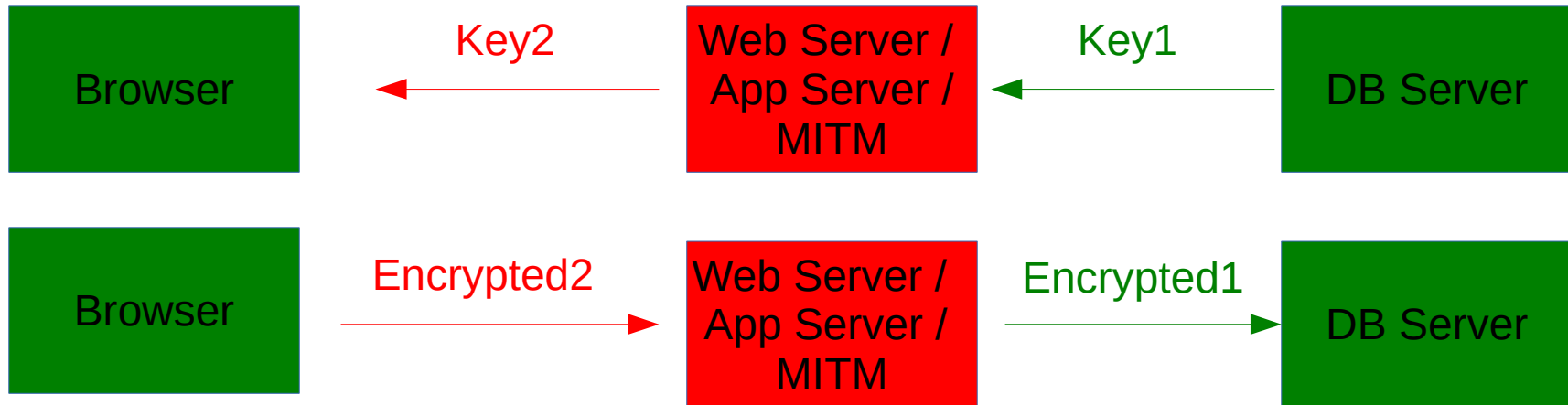
Use case 2: Expected Flow

- Encrypted communication between untrusted termination points



Use case 2: Attack

- Classic MITM Attack: Change the key



MITMA on E2E Encryption

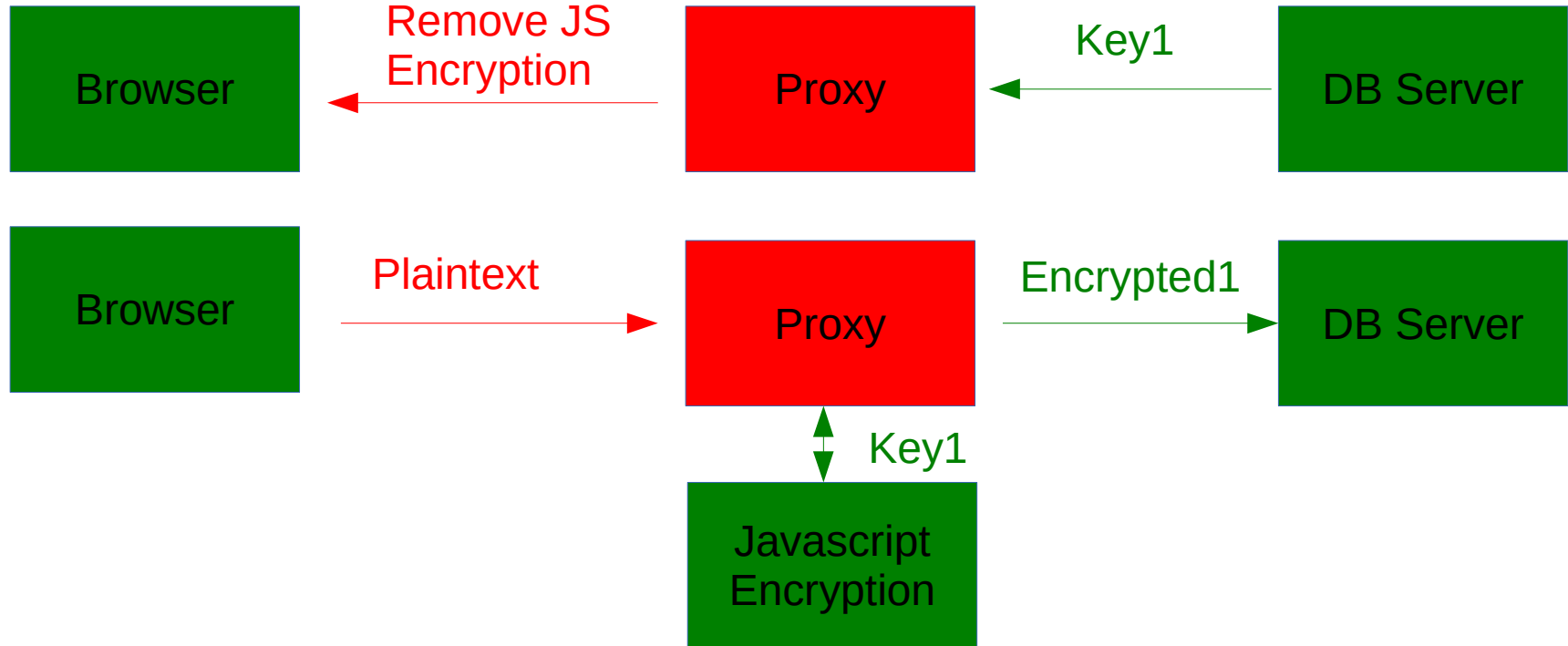
Wikipedia:

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

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

Request

Raw Params Headers Hex

Content-Length: 2164

```
statemachineStateName=7dccaaf4b0aca03ef0175d6ab36d40dc2&statemachineEventName=Su  
bmitPersonaliseMyName&encryptedString=0256%25004a2db124ce5da52d14a41afcd6f4ff1821  
3015d12e8491b48abe22b39ef41271bec6daadb2c0b27055b08e13b773cd944edaf4461d8d6e95  
065ee086c41fe2aae83c3ab4f8aaaa02bea7f869a15d9b228d654399caf4f3feae2a7aa77d849a  
db05531c097f4451344c2f620859ae040ca01b18049e0b6cf53d1852029e86f0ae52fe07fa1e0d  
35daa02312fb56135e5458d57967ea136ce64097b3f85aeeebb8efd5b0e6d3bdacacc71e59af85  
45c2a0634db386496432de68f35471deb5ec22fb0889d0969e3bedfe3bd3c4fbab5377f8461f31  
445ddd4a11e165d35d31621d551a6eb68c61e94e4b3c69b762ccda388971b12d17a27eff715f  
68c28e7a30c9003b965676c0fa31b96006fa7319ac22f7318ba73d2c6f8f0f1c0d7c7c80727ef14  
d1f6f595c0fd693c579e911e07f659c8a1a636dc67665fe6ce5c9b4bbf1cd9ac9ece56804fa8a89  
71b12d17a27eff0b11ce5ffd788a55512e62f57c1e7d6e5139e69c8ad1457d6ab69ed0263e58d2  
a70e360dc78793c647d207756fe6ba2bcb8785569fc25a6a9cf2e01620610c825f3b1fe31c555f  
cb956bee9e6ca5c3d2931ac1f93ea61eb77cd89b974c2c18a7f2bffa675379d12087c448096c1  
bf94c709806032daff7018218b17f66f93e6e62f68e26e5d0ffc5621ce8fad5a521207da792d6e  
46dce7691e0ea58eaf00a473dfb8ebb6c51b62f68e26e5d0ffc57621ce8fad5a521207da792d6e  
83091ba65e83e9d36c4842fb7e3ee3abb9b12156cd24d49c0e6ba8ccfa5078f77fbf611a0f4dc85  
9163c9be8c86bc338d34552e730fa3880c2ec1d39e997be8b1641fda24284a66d50b84b7fd2c0  
fc82c81bb33abebc36e393c5992ba7240abbed6a6c8db4661b3fe63fbc0c01b4c3935fb57af91b  
e46fd508ffdc37f20193fcf77c14343e962c0064833f1a5bfc02c17af275d10214f1c289ff89d6b1  
d190237dcd83082bbd887f44c87664446258e6f5a7bead179dcbd05286fa15ec6c730958520a  
4686f5b55985a0c5dd68f83fa0ef2e8efbdacb22a76594cc9f771f48b9596452426c10e36e9d8f0  
b790e35cf32387134414ac297fa02896f4281f64edb7a8aec51a6523e78b3dd2de9af2fbee2a1ae  
5795ff91a2d06d1c502af74fa21c0467e787c278a8247f761313b63cbf693380021adb7bf97f866  
a853ab072ceb16594cc9f771f48b9596452426c10e36e56cbbb0220ef52d1b730bd6fbfc7fdab&  
RSA_KEY_INDEX=1&publicKey=9E1C9EB9BD6E2333F8384194BCF06A0EB418779F436B129ED  
9EF0027DEEF11BF41F5C6AF94A0066658CA7A7ACC5EAA3B2011ED5283023C8EE272B731F29  
E9590AAAC2664676578DEB29E29F2D0F130F2E65AF8D2F7A458D97DACC7612430BF4FC2014  
B23EAD8EFDAD0A83F6D90D94A598EA72DC59DC2E014015854E14CCDCB5
```

0-day stops here

Plugin: After

Request

Raw Params Headers Hex

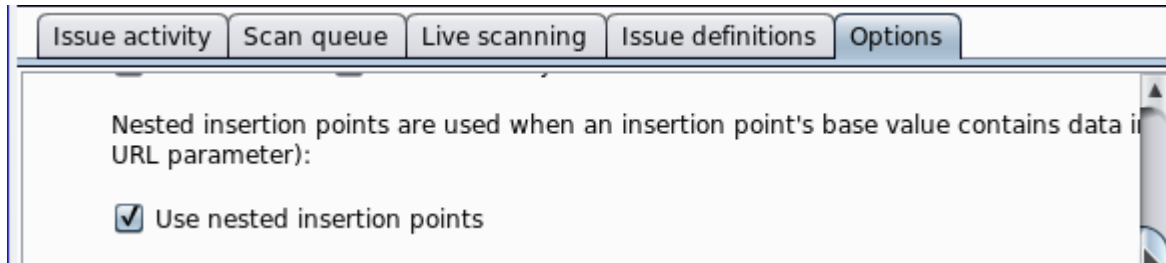
```
statemachineStateName=7dceaf4b0aca02ef0175d6ab36d40dc2&statemachineEventName=SubmitPersonaliseMyName&encryptedString=SEQUENCE_NUMBER%3D1%26COMPANY_CODE%3DDBSS%26DATA%2520PACKET%2520NAME%3DSAVE_NICK_NAME_CUSTOMER%26REQUEST_ID%3DIB094%26SERVICE_ID%3D0000000000000370%26CUST_NICK_NAME_NEW%3Dtest<script>alert(window.location.href)</script>%26CUST_NICK_NAME_OLD%3DTest%26MODIFY_SERVICE_ID%3D0000000000000370%26MODIFY_COMPANY_CODE%3DDBSS%26MODIFY_DATA_PACKET_NAME%3DRETRIEVE_PERSONALISE_NAME%26MODIFY_REQUEST_ID%3DIB094%26MODIFY_PROCESS%3DY&encryptedString2=DATA%2520PACKET%2520NAME%3DVALIDATE_PERSONALISE_NAME%26REQUEST_ID%3DIB094%26REQUEST_TYPE%3DSINGLE%26ACTOR_ID%3DP0000000008114363%26COMPANY_CODE%3DDBSS%26SERVICE_ID%3D0000000000000370%26CUST_NICK_NAME_NEW%3Dtest%26CUST_NICK_NAME_OLD%3DTest%26MODIFY_PROCESS%3DY%26MODIFY_SERVICE_ID%3D0000000000000370&RSA_KEY_INDEX=1&publicKey=9E1C9EB9BD6E2333F8384194BCF06A0EB418779F436B129ED9EF0027DEEF11BF41F5C6AF94A0066658CA7A7ACC5EAA3B2011ED5283023C8EE272B731F29E9590AAAC2664676578DEB29E29F2D0F130F2E65AF8D2F7A458D97DACC7612430BF4FC2014B23EAD8EFDAD0A83F6D90D94A598EA72DC59DC2E014015854E14CCCDBC5
```

? < + > 0 matches

Ready

Nested parameters

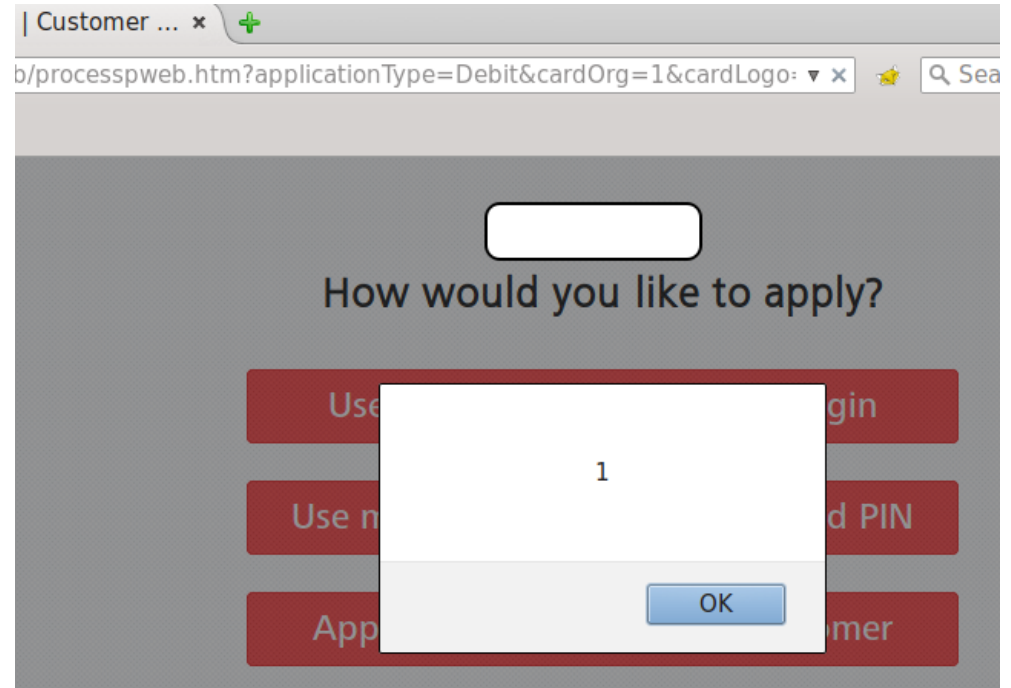
- EncryptedString: encoded(VAR1=VALUE1&VAR2=VALUE2)



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

Plugin: Hack

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



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>