Burp-ing through your cryptography shield

08.11.2018

Atos

1

Who am I?

# ~# whoami&&id

- root\n uid=0(root) gid=0(root) groups=0(root)
- **Work and education:**

Pentester	@Atos Romania
BEng	@University “Politehnica” of Bucharest
MEng	@University “Politehnica” of Bucharest
MSc	@Bucharest Academy of Economic Studies
Member	@Romanian Security Team community
Speaker	@DefCamp 2017 && @OWASP RO 2018
CEH	

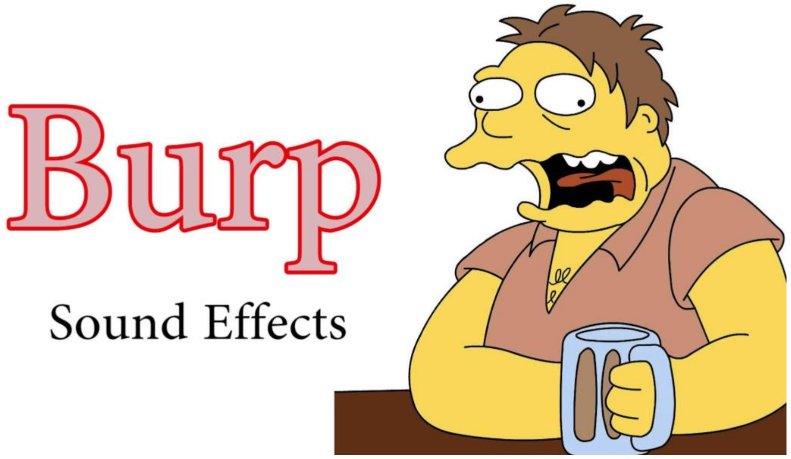
- Interests:
  - Web App Security
  - Infra Security
  - IoT Device Security
- Contact: @matasareanu13

2

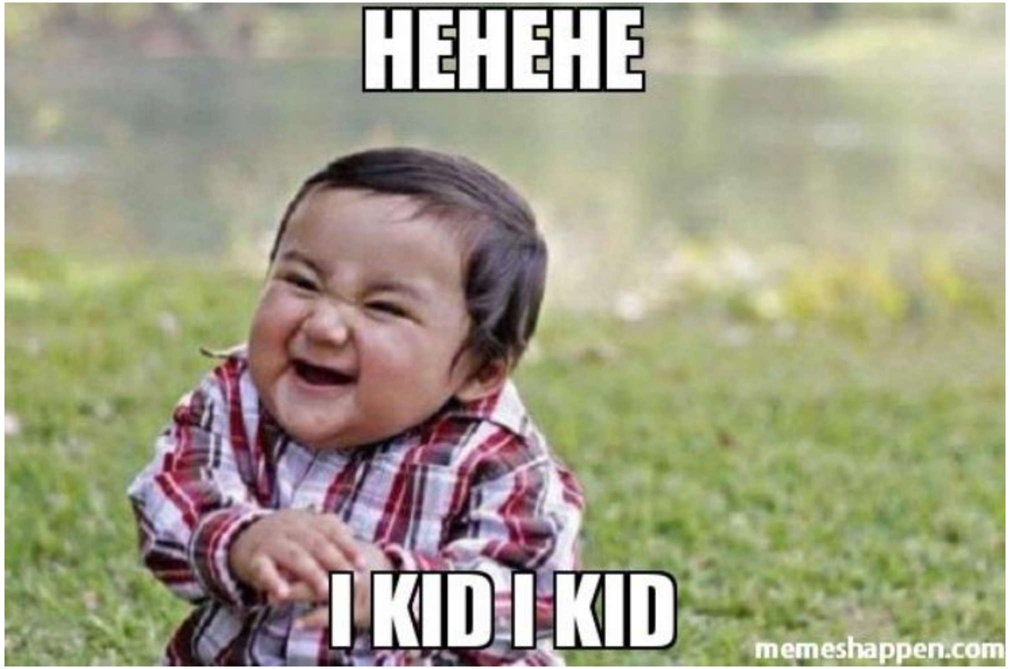
Glossary ☺

What is Burp-ing?

- Burping (also known as belching, ructus, eruptus or eructation) is the release of gas from the digestive tract (mainly esophagus and stomach) through the mouth.



What is Burp-ing?

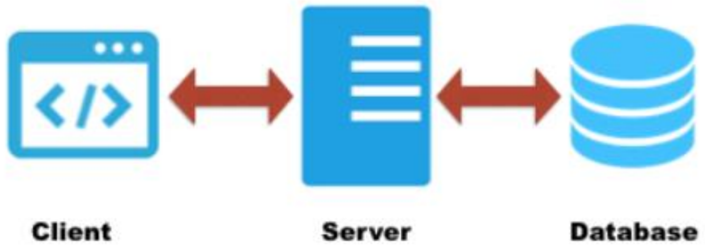


What is Burp?

Burp Suite is the leading software for web security testing_

What is a Thick Client?

- A thick client, also known as Fat Client is a client in client–server architecture or network and typically provides rich functionality, independent of the server. In these types of applications, the major processing is done at the client side and involves only aperiodic connection to the server.



3 What is the problem?

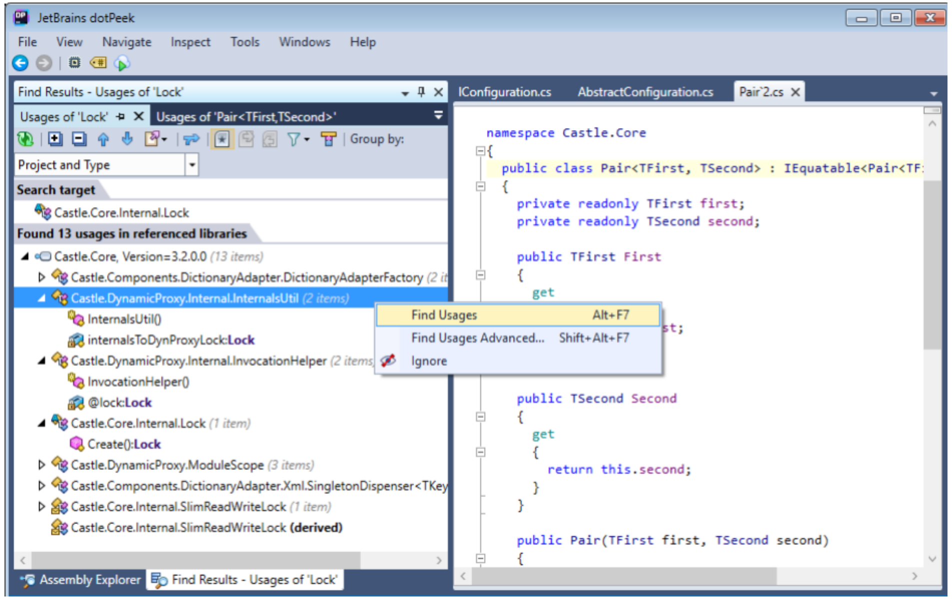
What is the problem at hand?

- Be on a pentest assignment with a thick client in front of you
- Set up Burp as a system proxy
- Manage to finally intercept requests
- Cool – SOAP requests
- Stumble upon gibberish in the requests and response of the application

STUCK

What to do?

Decompile the .NET application



JetBrains dotPeek

File View Navigate Inspect Tools Windows Help

Find Results - Usages of 'Lock'

Usages of 'Lock' | Usages of 'Pair<TFirst,TSecond>'

Group by: Project and Type

Search target

Castle.Core.Internal.Lock

Found 13 usages in referenced libraries

- Castle.Core, Version=3.2.0.0 *(13 items)*
  - Castle.Components.DictionaryAdapter.DictionaryAdapterFactory *(2 it*
  - Castle.DynamicProxy.Internal.InternalsUtil *(2 items)*
    - InternalsUtil()
    - internalsToDynProxyLock:**Lock**
  - Castle.DynamicProxy.Internal.InvocationHelper *(2 items)*
    - InvocationHelper()
    - @lock:**Lock**
  - Castle.Core.Internal.Lock *(1 item)*
    - Create():**Lock**
  - Castle.DynamicProxy.ModuleScope *(3 items)*
  - Castle.Components.DictionaryAdapter.Xml.SingletonDispenser<TKey
  - Castle.Core.Internal.SlimReadWriteLock *(1 item)*
  - Castle.Core.Internal.SlimReadWriteLock **(derived)**

Find Usages	Alt+F7
Find Usages Advanced...	Shift+Alt+F7
Ignore	

IConfiguration.cs AbstractConfiguration.cs Pair`2.cs

```
namespace Castle.Core
{
  public class Pair<TFirst, TSecond> : IEquatable<Pair<TF
  {
    private readonly TFirst first;
    private readonly TSecond second;

    public TFirst First
    {
      get
                             st;

    public TSecond Second
    {
      get
      {
        return this.second;
      }
    }

    public Pair(TFirst first, TSecond second)
    {
```

Assembly Explorer | Find Results - Usages of 'Lock'

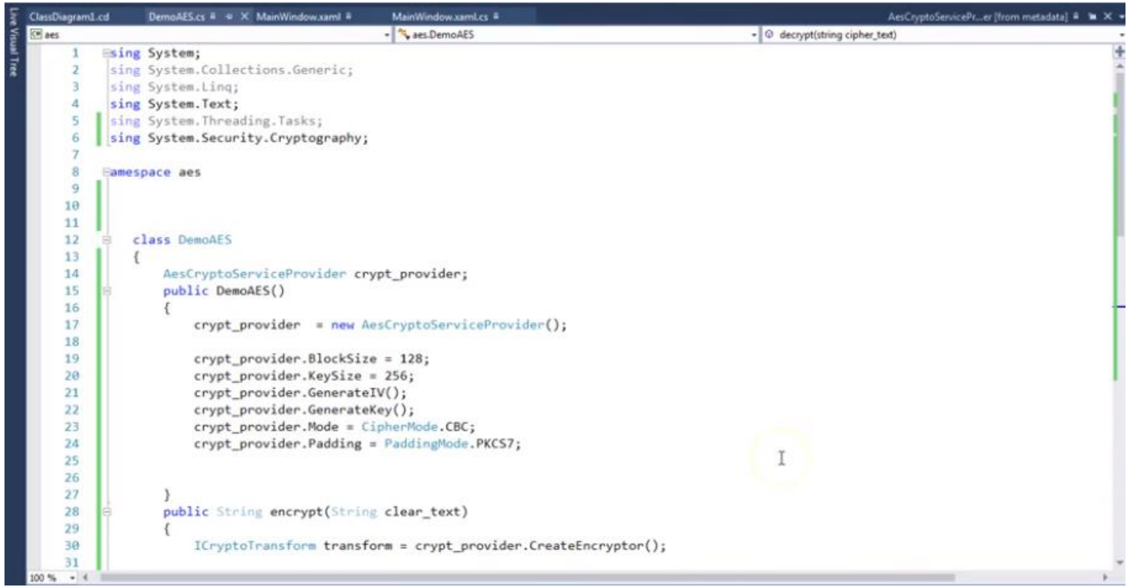
dotPeek part of ReSharper Ultimate

Free .NET Decompiler and Assembly Browser

Atos

What to do?

- Find the encryption method used



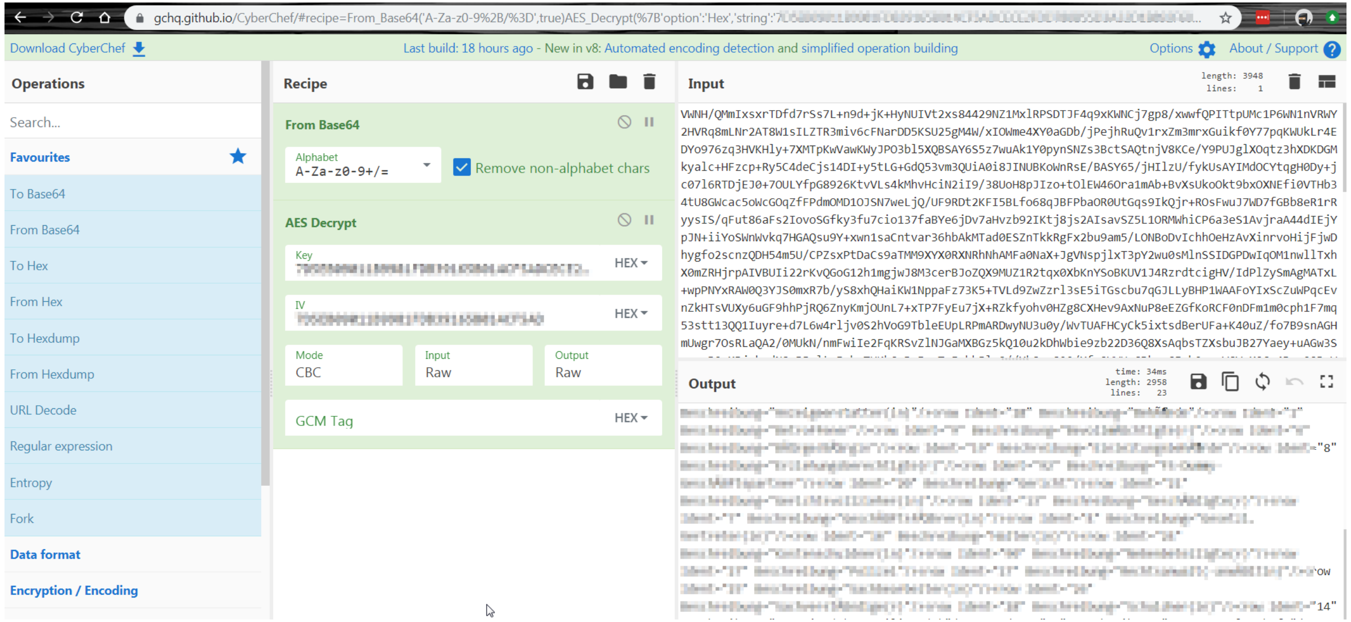
```
sing System;
sing System.Collections.Generic;
sing System.Linq;
sing System.Text;
sing System.Threading.Tasks;
sing System.Security.Cryptography;

amespace aes



    class DemoAES
    {
        AesCryptoServiceProvider crypt_provider;
        public DemoAES()
        {
            crypt_provider  = new AesCryptoServiceProvider();

            crypt_provider.BlockSize = 128;
            crypt_provider.KeySize = 256;
            crypt_provider.GenerateIV();
            crypt_provider.GenerateKey();
            crypt_provider.Mode = CipherMode.CBC;
            crypt_provider.Padding = PaddingMode.PKCS7;


        }
        public String encrypt(String clear_text)
        {
            ICryptoTransform transform = crypt_provider.CreateEncryptor();
```

What to do?

What to do?

Find the IV and the Key ☺

Cyberchef



gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true)AES_Decrypt(%7B'option':'Hex','string':'[redacted]

Download CyberChef

Last build: 18 hours ago - New in v8: Automated encoding detection and simplified operation building

Options | About / Support

Operations

Search...

Favourites

- To Base64
- From Base64
- To Hex
- From Hex
- To Hexdump
- From Hexdump
- URL Decode
- Regular expression
- Entropy
- Fork

Data format

Encryption / Encoding

Recipe

From Base64

Alphabet: A-Za-z0-9+/=

✓ Remove non-alphabet chars

AES Decrypt

Key: [redacted] HEX

IV: [redacted] HEX

Mode: CBC | Input: Raw | Output: Raw

GCM Tag HEX

Input

length: 3948
lines: 1

```
VWNH/QMmIxsxrTDfd7rSs7L+n9d+jK+HyNUIVt2xs84429NZ1MxlRPSDTJF4q9xKWNCj7gp8/xwwfQPITtpUMc1P6WN1nVRWY
2HVRq8mLNr2AT8W1sILZTR3miv6cFNarDD5KSU25gM4W/xIOWme4XY0aGDb/jPejhRuQv1rxZm3mrxGuikf0Y77pqKWUkLr4E
DYo976zq3HVKHly+7XMTpKwVawKWyJPO3bl5XQBSAY6S5z7wuAk1Y0pynSNZs3BctSAQtnjV8KCe/Y9PUJglXOqtz3hXDKDGM
kyalc+HFzcp+Ry5C4deCjs14DI+y5tLG+GdQ53vm3QUiA0i8JINUBKoWnRsE/BASY65/jHIlzU/fykUsAYIMdOCYtqgH0Dy+j
c07l6RTDjEJ0+7OULYfpG8926KtvVLs4kMhvHciN2iI9/38UoH8pJIzo+tOlEW46Ora1mAb+BvXsUkoOkt9bxOXNEfi0VTHb3
4tU8GWcac5oWcGOqZfFPdmOMD1OJSN7weLjQ/UF9RDt2KFI5BLfo68qJBFPbaOR0UtGqs9IkQjr+ROsFwuJ7WD7fGBb8eR1rR
yysIS/qFut86aFs2IovoSGfky3fu7cio137faBYe6jDv7aHvzb92IKtj8js2AIsavSZ5L1ORMWhiCP6a3eS1AvjraA44dIEjY
pJN+iiYoSWnWvkq7HGAQsu9Y+xwn1saCntvar36hbAkMTad0ESZnTkkRgFx2bu9am5/LONBoDvIchhOeHzAvXinrvoHijFjwD
hygfo2scnzQDH54m5U/CPZsxPtDaCs9aTMM9XYX0RXNRhNhAMFa0NaX+JgVNspjlxT3pY2wu0sMlnSSIDGPDwIqOM1nwllTxh
X0mZRHjrpAIVBUIi22rKvQGoG12h1mgjwJ8M3cerBJoZQX9MUZ1R2tqx0XbKnYSoBKUV1J4RzrdtcigHV/IdPlZySmAgMATxL
+wpPNYxRAW0Q3YJS0mxR7b/yS8xhQHaiKW1NppaFz73K5+TVLd9ZwZzrl3sE5iTGscbu7qGJLLyBHP1WAAFoYIxScZuWPqcEv
nZkHTsVUXy6uGF9hhPjRQ6ZnyKmjOUnL7+xTP7FyEu7jX+RZkfyohv0HZg8CXHev9AxNuP8eEZGfKoRCF0nDFm1m0cph1F7mq
53stt13QQ1Iuyre+d7L6w4rljv0S2hVoG9TbleEUpLRPmARDwyNU3u0y/WvTUAFHCyCk5ixtsdBerUFa+K40uZ/fo7B9snAGH
mUwgr7OsRLaQA2/0MUkN/nmFwiIe2FqKRSvZlNJGaMXBGz5kQ10u2kDhWbie9zb22D36Q8XsAqbsTZXsbuJB27Yaey+uAGw3S
```

Output

time: 34ms
length: 2958
lines: 23

[redacted]

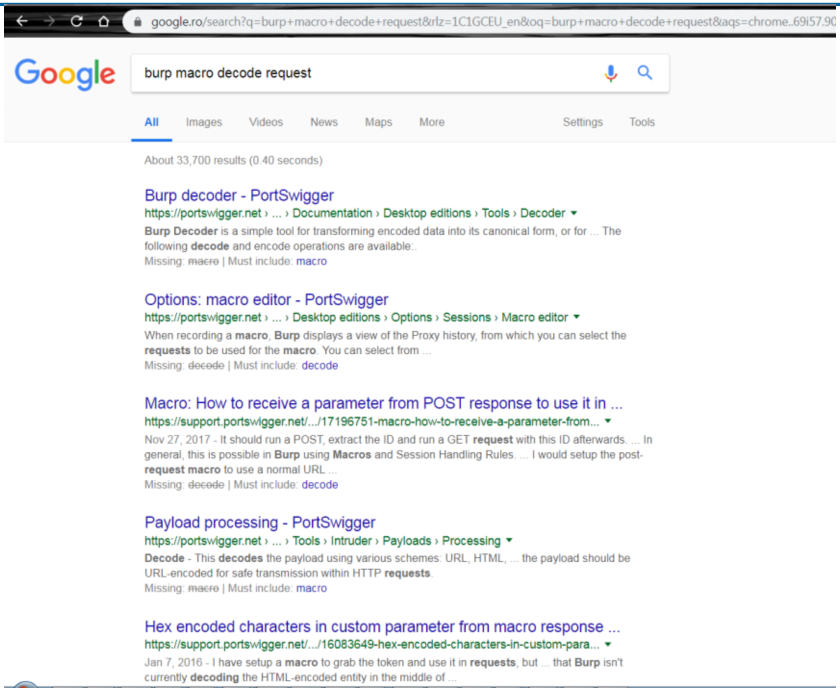
Results ☺

- We are starting to get there ☺

- But decrypting, decoding modifying the parameters, re-encoding, re-encrypting for each request and for each payload starts to get old very fast.

RABBIT'S HOWSE
Burp giving a hand

My Google Fu isn't providing



google.ro/search?q=burp+macro+decode+request&rlz=1C1GCEU_en&oq=burp+macro+decode+request&aqs=chrome..69i57.90

About 33,700 results (0.40 seconds)

Burp decoder - PortSwigger
https://portswigger.net › ... › Documentation › Desktop editions › Tools › Decoder
Burp Decoder is a simple tool for transforming encoded data into its canonical form, or for ... The following **decode** and encode operations are available:.
Missing: ~~macro~~ | Must include: macro

Options: macro editor - PortSwigger
https://portswigger.net › ... › Desktop editions › Options › Sessions › Macro editor
When recording a **macro**, **Burp** displays a view of the Proxy history, from which you can select the **requests** to be used for the **macro**. You can select from ...
Missing: ~~decode~~ | Must include: decode

Macro: How to receive a parameter from POST response to use it in ...
https://support.portswigger.net/.../17196751-macro-how-to-receive-a-parameter-from...
Nov 27, 2017 - It should run a POST, extract the ID and run a GET **request** with this ID afterwards. ... In general, this is possible in **Burp** using **Macros** and Session Handling Rules. ... I would setup the post-**request macro** to use a normal URL ...
Missing: ~~decode~~ | Must include: decode

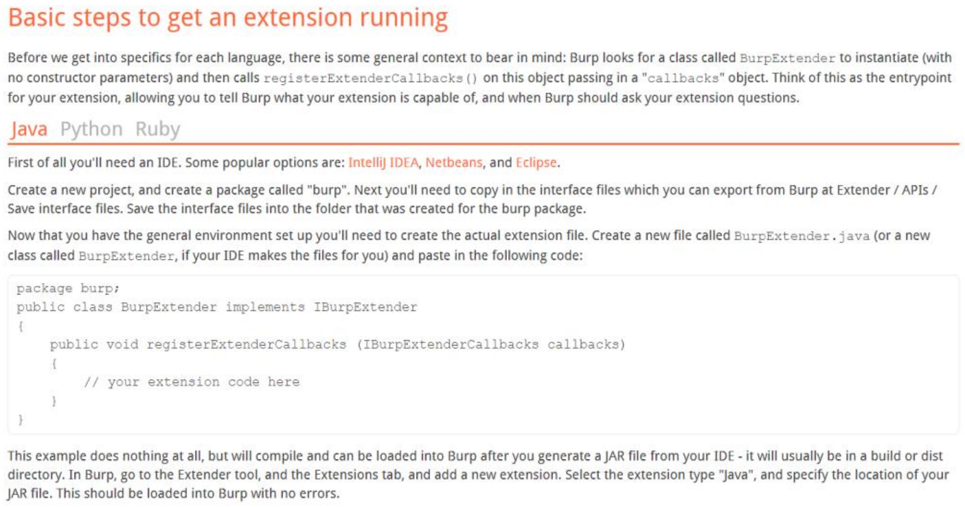
Payload processing - PortSwigger
https://portswigger.net › ... › Tools › Intruder › Payloads › Processing
Decode - This **decodes** the payload using various schemes: URL, HTML, ... the payload should be URL-encoded for safe transmission within HTTP **requests**.
Missing: ~~macro~~ | Must include: macro

Hex encoded characters in custom parameter from macro response ...
https://support.portswigger.net/.../16083649-hex-encoded-characters-in-custom-para...
Jan 7, 2016 - I have setup a **macro** to grab the token and use it in **requests**, but ... that **Burp** isn't currently **decoding** the HTML-encoded entity in the middle of ...

RABBIT'S HOWSE
TRY HARDER!

Writing a Burp extension ☺

- Getting started:
  - "Easy" way to extend Burp features
  - Basic steps:



Basic steps to get an extension running

Before we get into specifics for each language, there is some general context to bear in mind: Burp looks for a class called `BurpExtender` to instantiate (with no constructor parameters) and then calls `registerExtenderCallbacks()` on this object passing in a "`callbacks`" object. Think of this as the entrypoint for your extension, allowing you to tell Burp what your extension is capable of, and when Burp should ask your extension questions.

Java Python Ruby

First of all you'll need an IDE. Some popular options are: IntelliJ IDEA, Netbeans, and Eclipse.

Create a new project, and create a package called "burp". Next you'll need to copy in the interface files which you can export from Burp at Extender / APIs / Save interface files. Save the interface files into the folder that was created for the burp package.

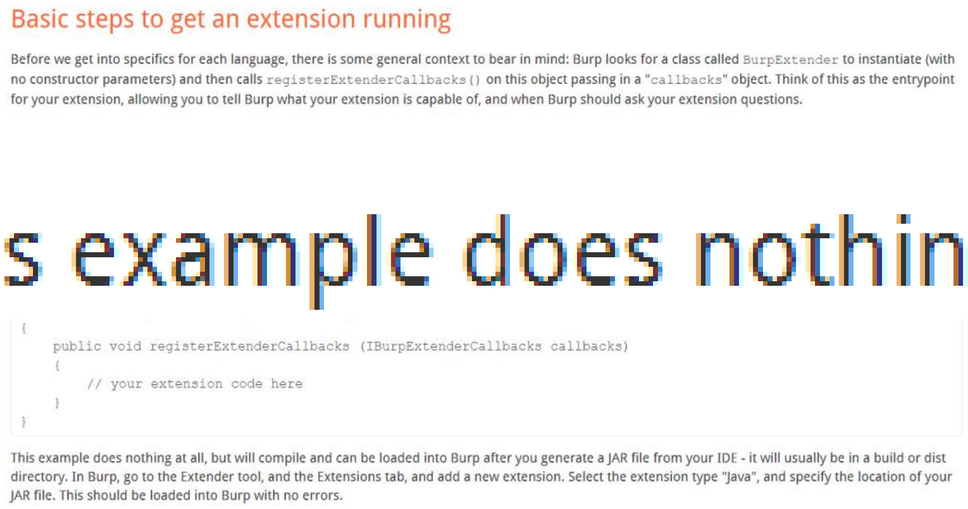
Now that you have the general environment set up you'll need to create the actual extension file. Create a new file called `BurpExtender.java` (or a new class called `BurpExtender`, if your IDE makes the files for you) and paste in the following code:

```
package burp;
public class BurpExtender implements IBurpExtender
{
    public void registerExtenderCallbacks (IBurpExtenderCallbacks callbacks)
    {
        // your extension code here
    }
}
```

This example does nothing at all, but will compile and can be loaded into Burp after you generate a JAR file from your IDE - it will usually be in a build or dist directory. In Burp, go to the Extender tool, and the Extensions tab, and add a new extension. Select the extension type "Java", and specify the location of your JAR file. This should be loaded into Burp with no errors.

Writing a Burp extension ☺

- Getting started:
  - "Easy" way to extend Burp features
  - Basic steps:



Basic steps to get an extension running

Before we get into specifics for each language, there is some general context to bear in mind: Burp looks for a class called `BurpExtender` to instantiate (with no constructor parameters) and then calls `registerExtenderCallbacks()` on this object passing in a "`callbacks`" object. Think of this as the entrypoint for your extension, allowing you to tell Burp what your extension is capable of, and when Burp should ask your extension questions.

This example does nothing at all

```
{
    public void registerExtenderCallbacks (IBurpExtenderCallbacks callbacks)
    {
        // your extension code here
    }
}
```

This example does nothing at all, but will compile and can be loaded into Burp after you generate a JAR file from your IDE - it will usually be in a build or dist directory. In Burp, go to the Extender tool, and the Extensions tab, and add a new extension. Select the extension type "Java", and specify the location of your JAR file. This should be loaded into Burp with no errors.

Writing a Burp extension ☺

Get more info about writing a Extension.

Learn by example ☺

https://portswigger.net/burp/extender#SampleExtensions

Writing a Burp extension ☺

Decide what language you will use??

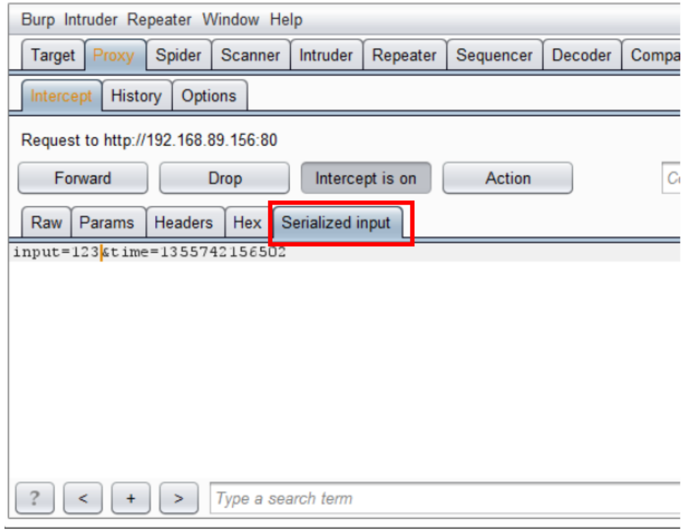
Writing a Burp extension ☺

Jython it is ☺

Writing a Burp extension ☺

https://portswigger.net/blog/sample-burp-suite-extension-custom-editor-tab

Start from something they built



Request to http://192.168.89.156:80

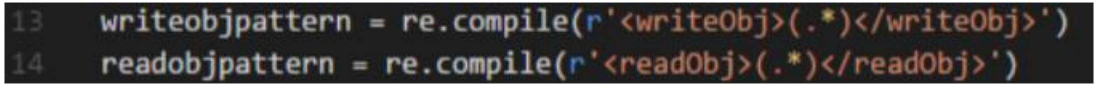
```
input=123&time=1355742156502
```

It’s always easier ☺

Writing a Burp extension ☺

- The encrypted values are found in SOAP requests under two XML tags:
  - <writeObj>
  - <readObj>

  - First step is to look for the pattern:



```
writeobjpattern = re.compile(r'<writeObj>(.*)</writeObj>')
readobjpattern = re.compile(r'<readObj>(.*)</readObj>')
```

Writing a Burp extension ☺

Decrypt the payload ☺

Don’t be that fast

PyCrypto doesn’t want to work in Jython ☹

Writing a Burp extension ☺

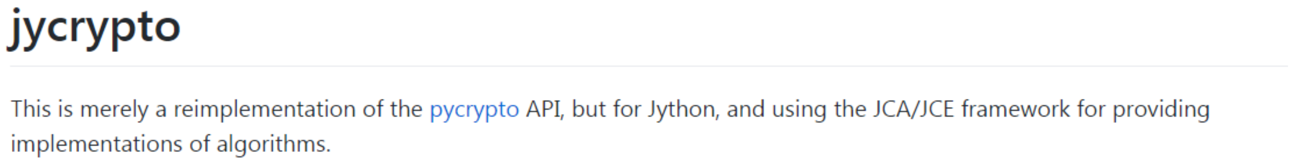
Decrypt the payload ☺

Don’t be that fast

PyCrypto doesn’t want to work in Jython ☹

Writing a Burp extension ☺

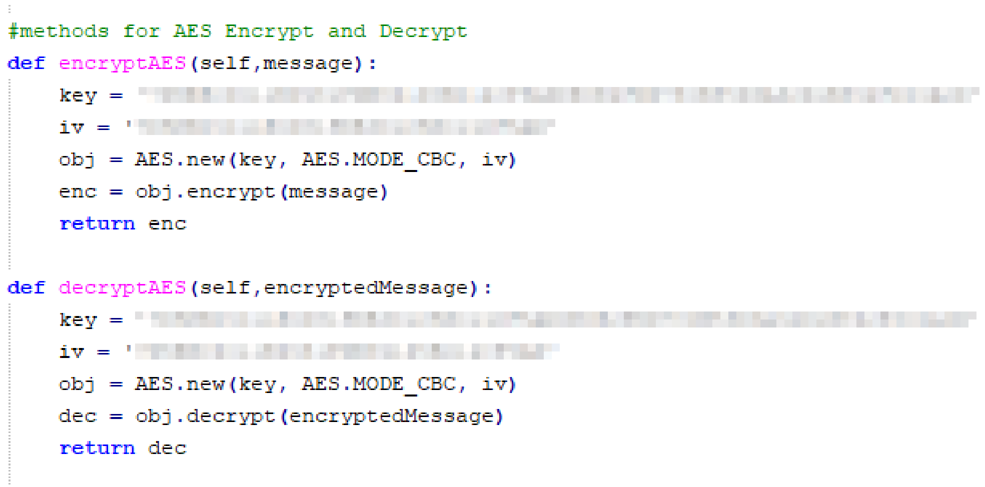
https://github.com/csm/jycrypto



jycrypto

This is merely a reimplementation of the pycrypto API, but for Jython, and using the JCA/JCE framework for providing implementations of algorithms.

Writing a Burp extension ☺



```
#methods for AES Encrypt and Decrypt
def encryptAES(self,message):
    key = '[redacted]
    iv = '[redacted]
    obj = AES.new(key, AES.MODE_CBC, iv)
    enc = obj.encrypt(message)
    return enc

def decryptAES(self,encryptedMessage):
    key = '[redacted]
    iv = '[redacted]
    obj = AES.new(key, AES.MODE_CBC, iv)
    dec = obj.decrypt(encryptedMessage)
    return dec
```

Writing a Burp extension ☺

- You now have .NET serialized objects…

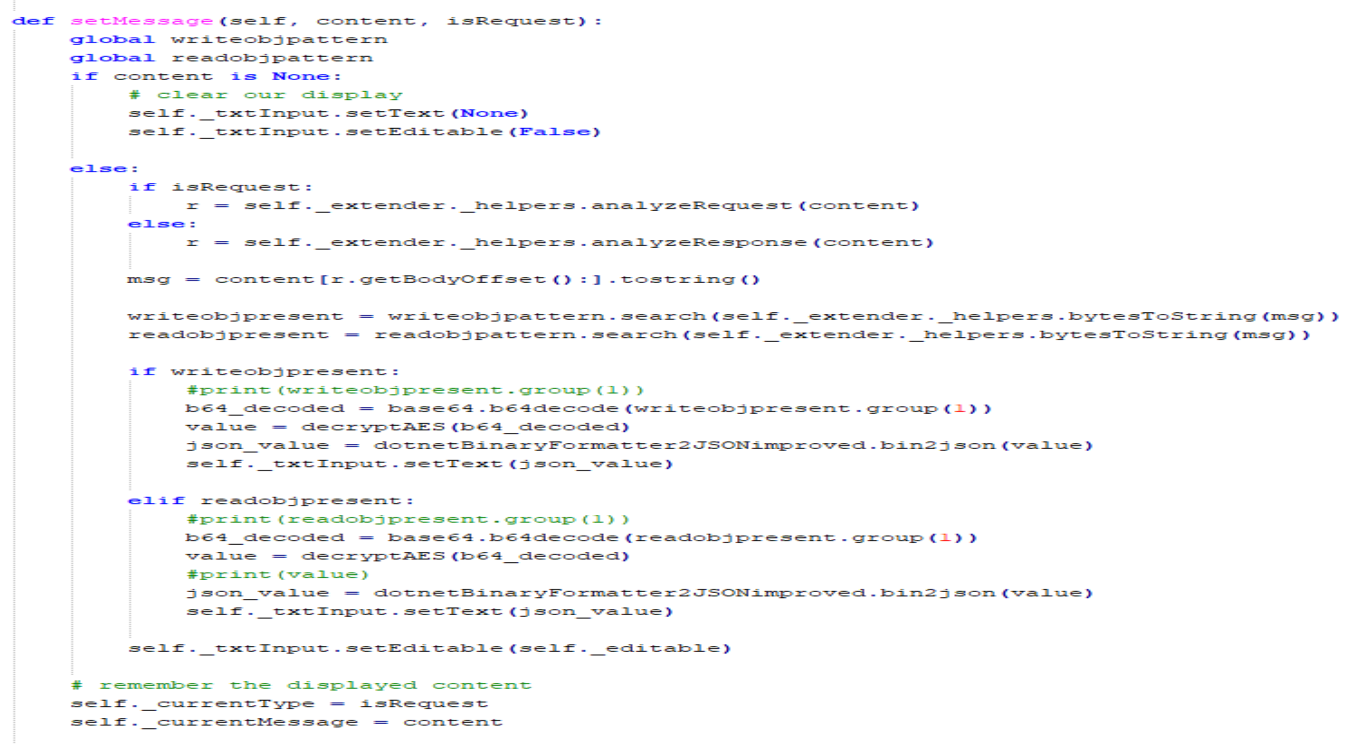
And now the struggle of Google-ing begins again..

Writing a Burp extension ☺

2 results (0.52 seconds)

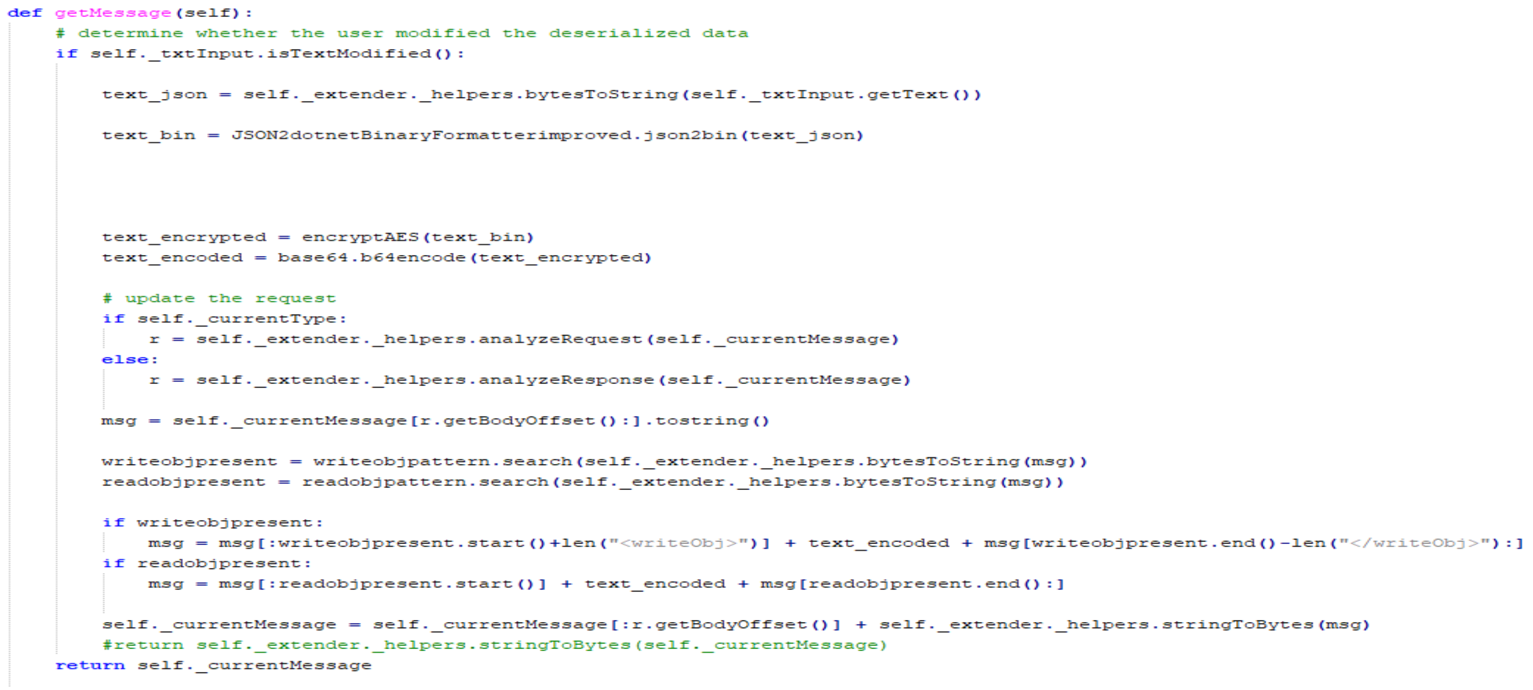
- https://github.com/agix/NetBinaryFormatterParser

Writing a Burp extension ☺



```
def setMessage(self, content, isRequest):
    global writeobjpattern
    global readobjpattern
    if content is None:
        # clear our display
        self._txtInput.setText(None)
        self._txtInput.setEditable(False)

    else:
        if isRequest:
            r = self._extender._helpers.analyzeRequest(content)
        else:
            r = self._extender._helpers.analyzeResponse(content)

        msg = content[r.getBodyOffset():].tostring()

        writeobjpresent = writeobjpattern.search(self._extender._helpers.bytesToString(msg))
        readobjpresent = readobjpattern.search(self._extender._helpers.bytesToString(msg))

        if writeobjpresent:
            #print(writeobjpresent.group(1))
            b64_decoded = base64.b64decode(writeobjpresent.group(1))
            value = decryptAES(b64_decoded)
            json_value = dotnetBinaryFormatter2JSONimproved.bin2json(value)
            self._txtInput.setText(json_value)

        elif readobjpresent:
            #print(readobjpresent.group(1))
            b64_decoded = base64.b64decode(readobjpresent.group(1))
            value = decryptAES(b64_decoded)
            #print(value)
            json_value = dotnetBinaryFormatter2JSONimproved.bin2json(value)
            self._txtInput.setText(json_value)

        self._txtInput.setEditable(self._editable)

    # remember the displayed content
    self._currentType = isRequest
    self._currentMessage = content
```

Writing a Burp extension ☺

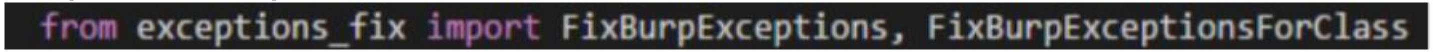


```
def getMessage(self):
    # determine whether the user modified the deserialized data
    if self._txtInput.isTextModified():

        text_json = self._extender._helpers.bytesToString(self._txtInput.getText())

        text_bin = JSON2dotnetBinaryFormatterimproved.json2bin(text_json)




        text_encrypted = encryptAES(text_bin)
        text_encoded = base64.b64encode(text_encrypted)

        # update the request
        if self._currentType:
            r = self._extender._helpers.analyzeRequest(self._currentMessage)
        else:
            r = self._extender._helpers.analyzeResponse(self._currentMessage)

        msg = self._currentMessage[r.getBodyOffset():].tostring()

        writeobjpresent = writeobjpattern.search(self._extender._helpers.bytesToString(msg))
        readobjpresent = readobjpattern.search(self._extender._helpers.bytesToString(msg))

        if writeobjpresent:
            msg = msg[:writeobjpresent.start()+len("<writeObj>")] + text_encoded + msg[writeobjpresent.end()-len("</writeObj>"):]
        if readobjpresent:
            msg = msg[:readobjpresent.start()] + text_encoded + msg[readobjpresent.end():]

        self._currentMessage = self._currentMessage[:r.getBodyOffset()] + self._extender._helpers.stringToBytes(msg)
        #return self._extender._helpers.stringToBytes(self._currentMessage)
    return self._currentMessage
```

Writing a Burp extension ☺

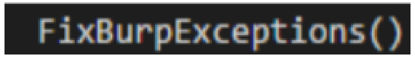
- Problem and solution:
  - Burp’s Java StackTrace doesn’t provide any help when using Jython!

  - Here comes the solution: https://github.com/securityMB/burp-exceptions
    - Put the file in your project folder
    - Import it in your code:



```
from exceptions_fix import FixBurpExceptions, FixBurpExceptionsForClass
```

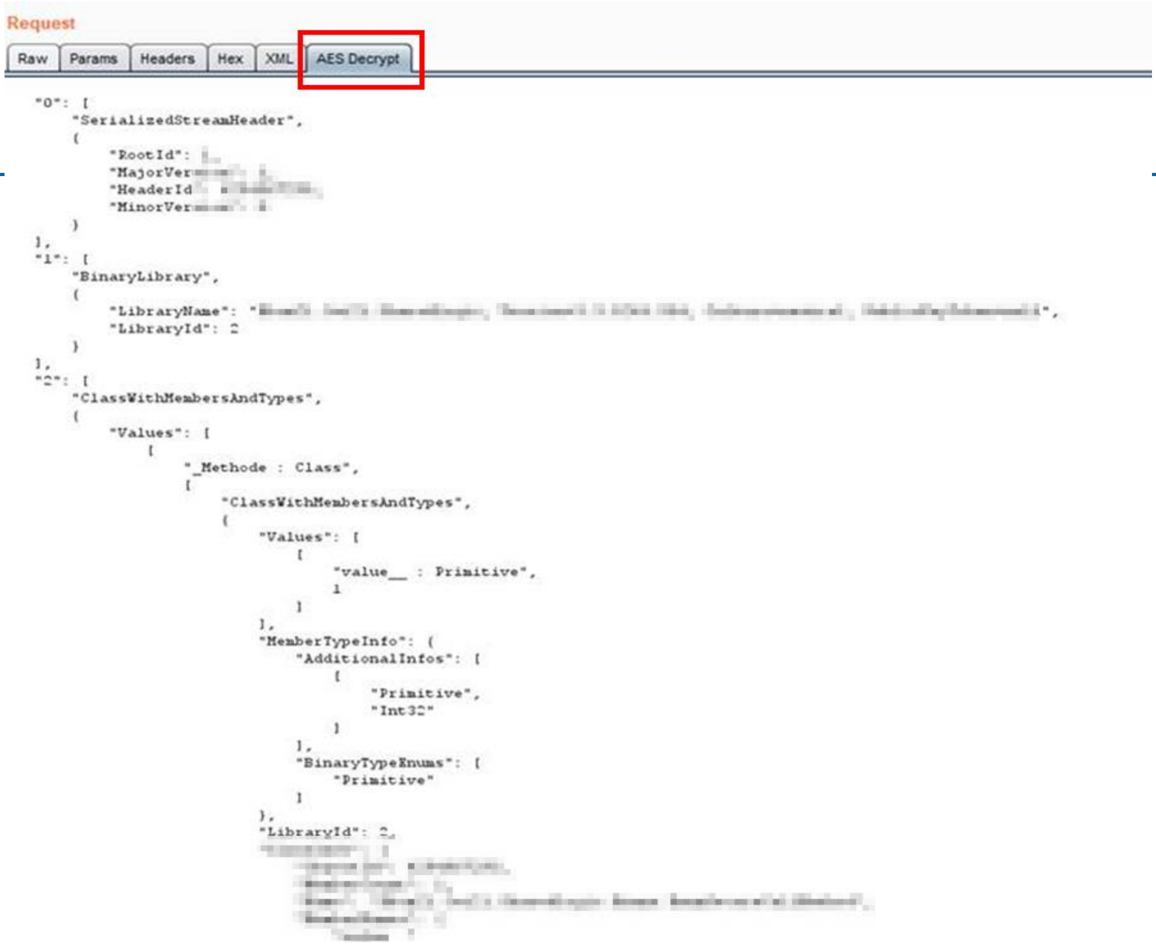
    - Call it in the end:



```
FixBurpExceptions()
```

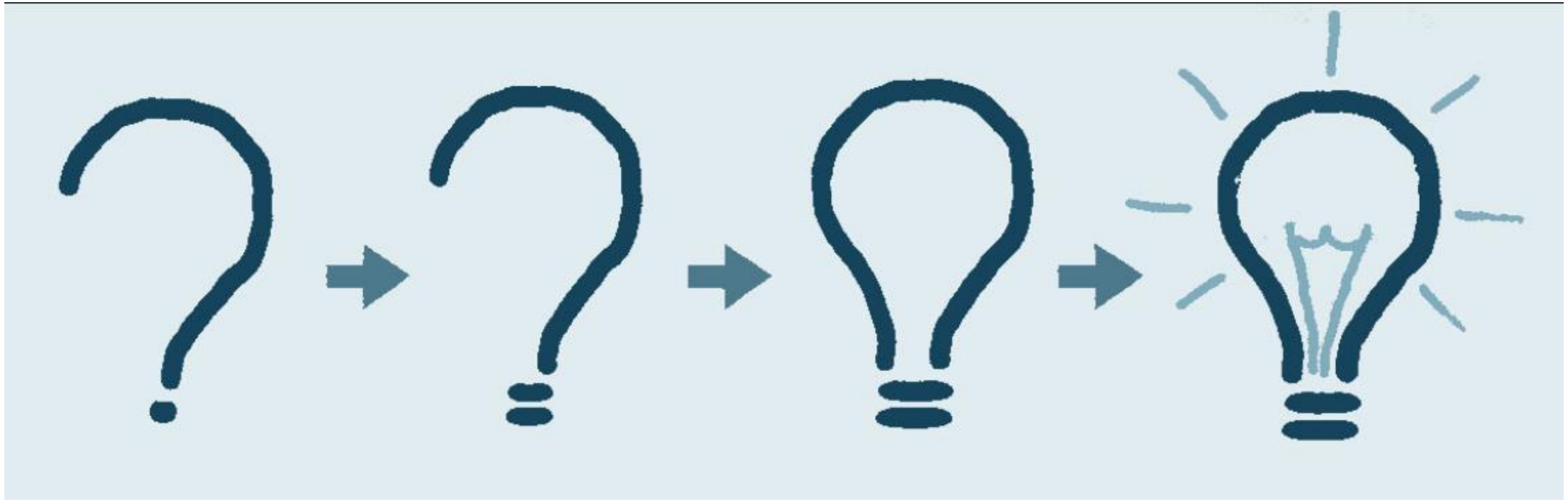
  - Now any Python exception is thrown in the Extender Output-Tab of Burp

DRUM
ROLL
SOUND



```
"0": [
    "SerializedStreamHeader",
    {
        "RootId": [redacted]
        "MajorVer[redacted]
        "HeaderId[redacted]
        "MinorVer[redacted]
    }
],
"1": [
    "BinaryLibrary",
    {
        "LibraryName": "[redacted]",
        "LibraryId": 2
    }
],
"2": [
    "ClassWithMembersAndTypes",
    {
        "Values": [
            [
                "_Methode : Class",
                [
                    "ClassWithMembersAndTypes",
                    {
                        "Values": [
                            [
                                "value__ : Primitive",
                                1
                            ]
                        ],
                        "MemberTypeInfo": {
                            "AdditionalInfos": [
                                [
                                    "Primitive",
                                    "Int32"
                                ]
                            ],
                            "BinaryTypeEnums": [
                                "Primitive"
                            ]
                        },
                        "LibraryId": 2,
                        [redacted]
```

Questions? And let’s hope some answers ☺



Thanks

For more information:
contact@cosminr.me
@Matasareanu13
github.com/Matasareanu

Or you know, in person ☺