Hacking Web App like a pimp

a different approach for web app pentest proxies

*@_hugsy_*

$ who

- By day *Christophe Alladoum*
  - Infosec consultant for Sense of Security (we're hiring !!!)
    - web app/external/internal pentests, code review, etc.

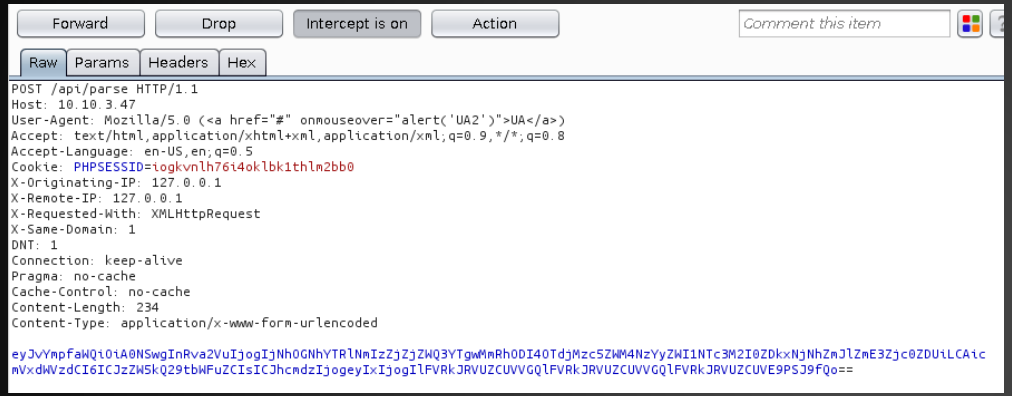
- By night *hugsy* (no questions 😂)
  - Do stuff
    - Low-level, RE, CTF addict, research, tool dev

$ why

- Burp is THE reference tool for web app pentest proxy
  - Useful modules (Proxy, Repeater, Intruder ($$$), etc.)
  - Well structured for comm. between modules
  - Nice GUI

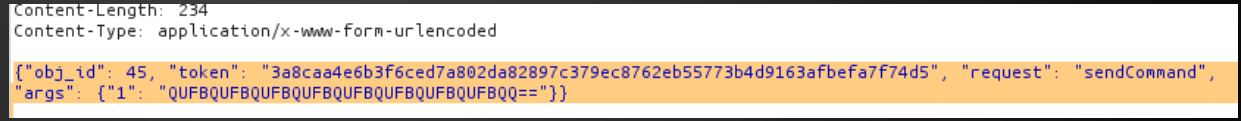
- But it is limited at MANY levels
  - Heaps and heaps of libs on top of Java VM
    - performance --
  - Works well for traditional web apps, almost useless for unconventional ones

$ why



Burp does not handle those cases...

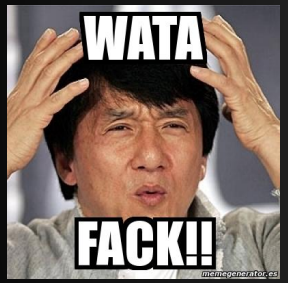
... like at all



Ok great, let's write a Burp plugin

$ why

Because



- Writing plugins for Burp is insanely complicated
  - To make it “simpler”, Burp implements unnatural bindings
    - Ruby over Java
    - Python over Java
- Awesome tools exist (like `burst` or `mitmproxy`), very extensible **BUT** specific to one language
- And (probably most importantly) I like having my own tools

So I wrote a tool...

$ info 

Introducing *proxenet*

From Ancient Greek, *πρόξενος* (próksenos, "public guest").

- A negotiator ; a factor ; a go-between.
- A mediator involved in immoral bargains (see pimp).

Goal

- Building a hacker friendly proxy **easily** pluggable.
- Micro-kernel approach: core does as little as possible, modules do the rest.

$ info Proxenet

- Plugin driven proxy for pentests
  - Bottom line : proxenet does **NOTHING** to HTTP layer

- 100% pure C code
  - Crazy fast
  - POSIX multi-threading
  - Super low memory use (w/ 5 VMs loaded ⇒ ~⅕ Burp mem use)

- Can view / edit / transform any request via plugins
  - Multiple linked list mechanism prevents overlap between plugins

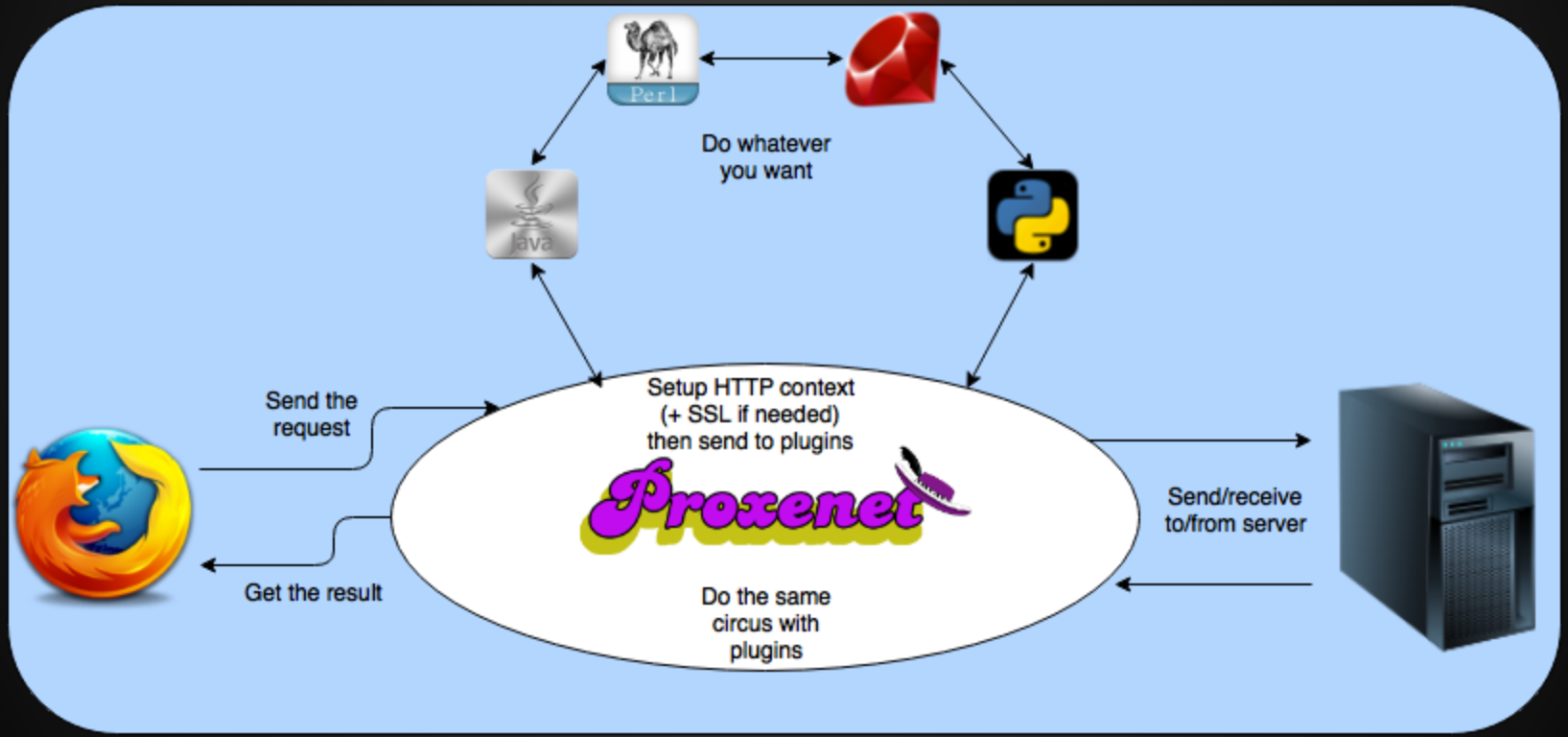
and more...

$ info Proxenet

- Some of the features
  - Full POSIX (Linux, OpenBSD, FreeBSD, OSX*)
  - Lightweight SSL/TLS
    - Full SSL/TLS interception (internal CA)
    - SSL/TLS client certificate authentication support
  - Future ready (i.e. IPv6 support)
  - HTTP Proxy forwarding
  - Regex filtering (white / black list)
  - Priority mechanism
  - Cool logo and nice ‘90 style colors
  - etc.

- Can be chained behind your favourite proxy (Burp, Zap, Proxystrike, etc.)

The Big Picture



DIY

- Meant to be ***extremely*** easily to add new plugins
  - If you think it's hard, it means I failed 😒
- To be valid a plugin **must** have:
  - One request function (default: `proxenet_request_hook`)
  - One response function (default: `proxenet_response_hook`)
- Use the rest of the code for whatever you like (other functions, tests, etc.)

DIY

The hook functions are called by the core with **3*** arguments:

1. A request/response identifier (Integer)
2. The request/response itself (Byte[ ] or String)
3. The URL (String)

The hook functions **must** return:

1. The modified request/response (Byte[ ] or String)
2. Null/None in case of error

DIY

- Literally as simple as it gets (yeah seriously)
  - Skeletons for each supported language in the RTFD

```
public class MyPlugin
{

    public static String AUTHOR = "";
    public static String PLUGIN_NAME = "";


    public static byte[] proxenet_request_hook(int request_id, byte[] request, String uri){
        return request;
    }

    public static byte[] proxenet_response_hook(int response_id, byte[] response, String uri){
        return response;
    }

    public static void main(String[] args){
        return;
    }
}
```

```
AUTHOR = ""
PLUGIN_NAME = ""

proc proxenet_request_hook {request_id request uri} {
    return $request
}

proc proxenet_response_hook {response_id response uri} {
    return $response
}

# add test cases here
```

```
AUTHOR = ""
PLUGIN_NAME = ""

function proxenet_request_hook (request_id, request, uri)
    return request
end

function proxenet_response_hook (response_id, response, uri)
    return response
end
```

```
module MyPlugin

    $AUTHOR = ""
    $PLUGIN_NAME = ""


    def proxenet_request_hook(request_id, request, uri)
        return request
    end

    def proxenet_response_hook(response_id, response, uri)
        return response
    end

end

if __FILE__ == $0
    # use for test cases
end
```

DIY

```java
package burp;

import IBurpExtender.*;
import ISessionHandlingAction.*;
import IParameter.*;
import java.util.Arrays;

public class BurpExtender implements IBurpExtender, ISessionHandlingAction
{

    public void registerExtenderCallbacks(IBurpExtenderCallbacks callbacks)
    {
        callbacks.getHelpers().setExtensionName("BurpAddHeader");
        callbacks.getHelpers().registerSessionHandlingAction();
        return;
    }

    public void performAction(IHttpRequestResponse currentRequest,
                              IHttpRequestResponse[] macroItems)
    {
        String newHeader;
        IExtensionHelpers helpers;
        IRequestInfo reqInf;
        List<String> headers;
        byte[] newHttpReq;
        byte[] body;

        newHeader = new String("X-Java-Injected: burp");
        helpers = this.callbacks.getHelpers();
        reqInf  = helpers.analyzeRequest(currentRequest);
        headers = reqInf.getHeaders();
        body = java.util.Arrays.copyOfRange(currentRequest.getRequest(),
                                            reqInf.getBodyOffset(),
                                            currentRequest.getRequest().length);

        headers.add( newHeader );  // this is actually the only line that matters

        newHttpReq = helpers.buildHttpMessage(headers,
                                              currentRequest.getRequest());

        currentRequest.setRequest(newHttpReq);
        return;
    }
}
```

U:@--- **BurpAddHeader.java** All of 1.5k (8,0) (Java/l pair Abbrev)

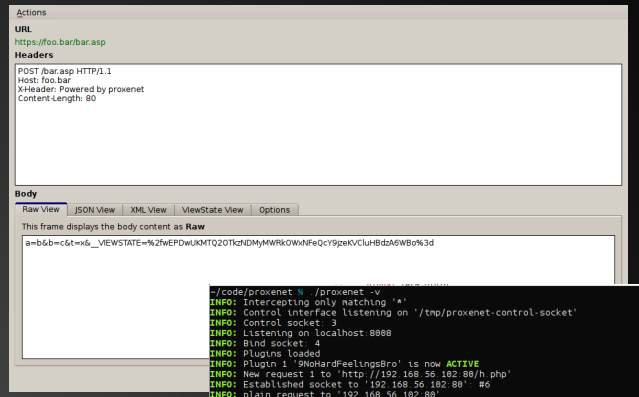
Written in ~1h (API search, test, debug, etc.)

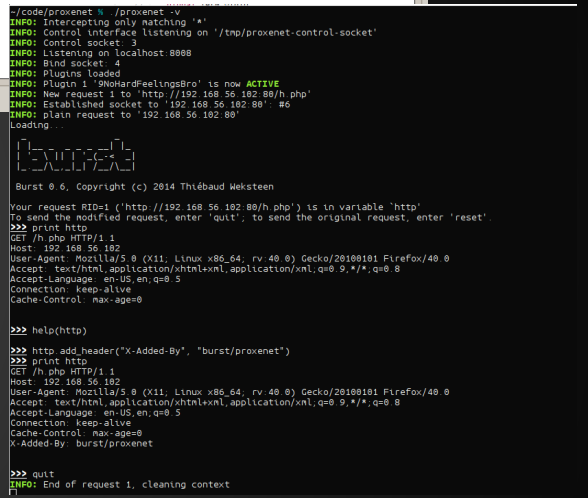
```java
public class AddHeader
{
    public static byte[] proxenet_request_hook(int request_id, byte[] request, String uri)
    {
        String myRequest = new String( request );
        myRequest = myRequest.replace("\r\n\r\n", "\r\nX-Java-Injected: proxenet\r\n\r\n");
        return myRequest.getBytes();
    }

    public static byte[] proxenet_response_hook(int response_id, byte[] response, String uri)
    {
        return response;
    }
}
```

U:@--- **ProxenetAddHeader.java** All of 441 (15,0) (Java/l pair Abbrev)

Written in ~5min (trivial API, relies on primitive types and objects)

DIY

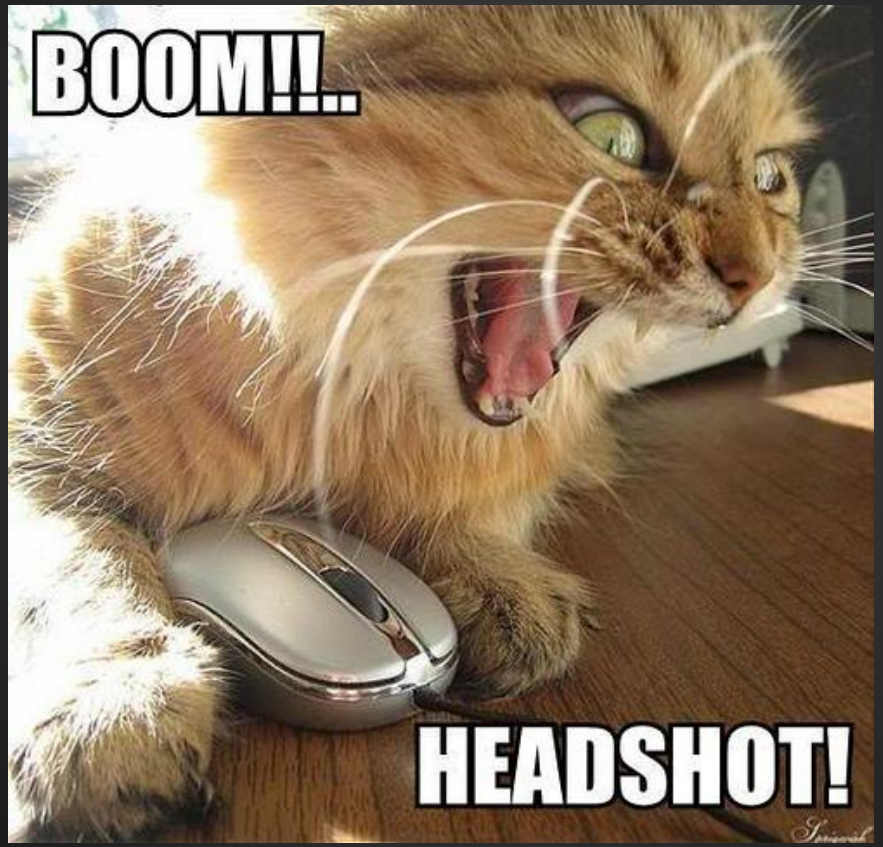




Some plugins already implemented :

- Interceptor GUI (based on PyQt4)
- Interceptor CLI (invoking *burst* from *proxenet*)
- Attacks plugins
  - automatic (stealth) ShellShock detection
  - CVE-2012-1823 (php-cgi code exec)
  - XXE
- Discovery plugins
  - automatic discovery on Directory Listing
  - retrieve all comments in HTML
  - add automatically parameters to all requests (&debug=1, &admin=1)

DIY



Demo

$ apt-get upgrade

- Keep improving code robustness & quality (core)
  - even though it's not too bad right now

CHECKMARX

Proxenet Scan Report

Project Name	Proxenet
Scan Start	Monday, July 13, 2015 3:07:40 PM
Preset	Default 2014
Scan Time	00h:02m:04s
Lines Of Code Scanned	14,121
Files Scanned	38
Report Creation Time	Monday, July 13,

	High	Medium	Low
To Verify	5	80	52
Not Exploitable	0	0	0
Confirmed	0	0	0
Urgent	0	0	0
Total	5	80	52

$ apt-get upgrade

- Add more features
  - JavaScript (v8) coming up
  - oCaml too (because why not?)
  - WebSockets maybe (because why not?)

- Add more plugins for/from community
  - ~10 publicly released plugins (intercept, XXE detection, dir list, etc.)
  - Feel free to contribute !!
    - Write & submit funky plugins
    - Report bugs (beerz 4 bugz policy)


BONUS
3 en 1
Pour un blanc PARFAIT comme au 1er jour !

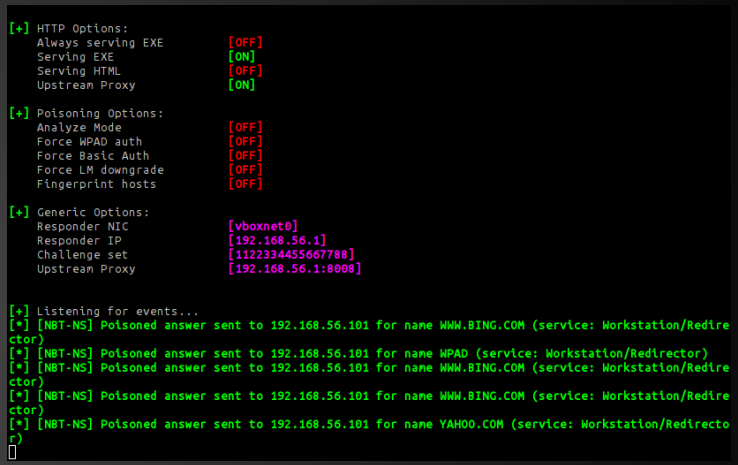
“proxenet-in-the-middle”© attack

Idea (from @lanjelot):

Why not use the plugins created for proxenet for easily view/modify HTTP streams in MITM attacks?

“proxenet-in-the-middle”© attack

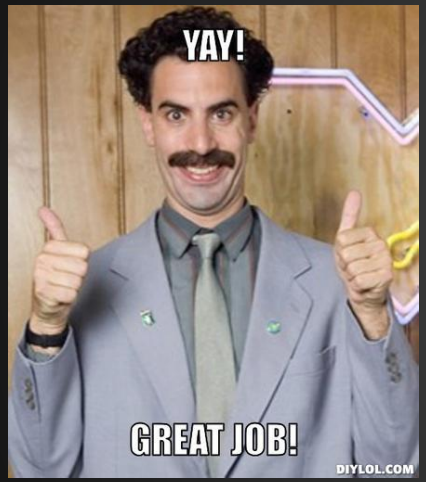
1. Use Responder
   a. Poison LLMNR on LAN
   b. Poison PAC (from WPAD requests) to proxenet



2. Use your proxenet plugin to modify HTTP on-the-fly
   a. Insert JavaScript (BeEF) in HTML body
   b. Replace on-the-fly documents (zip, doc, xls, etc.) with infected ones

Demo

“proxenet-in-the-middle”© attack



- Easy & practical attack for internal pentests
- Totally transparent for victims
- Free shells \o/

Thank you web browser auto-configuration mode

Thanks

proxenet repo: https://github.com/hugsy/proxenet.git
plugins public repo: https://github.com/hugsy/proxenet-plugins.git

@_hugsy_
hugsy
hugsy@blah.cat