The Honeynet PROJECT

Thug: a low-interaction honeyclient

Angelo Dell'Aera

Speaker

- Chief Executive Officer @ Honeynet Project
- Information Security Independent Researcher @ Antifork Research (10+ years)

Agenda

- Introduction
- Honeyclient technologies
- Thug
- Conclusions

Client-side attacks

- The number of client-side attacks has grown significantly in the past few years shifting focus on poorly protected vulnerable clients
- In the last years more and more attacks against client systems
- The browser is the most popular client system deployed on every user system
- A lot of vulnerabilities are identified daily and reported in the most used browsers and in third-party plugins

Honeyclients

- Just as the most known honeypot technologies enable research into server-side attacks, honeyclients allow the study of client-side attacks
- A complement to honeypots, a honeyclient is a tool designed to mimic the behavior of a user-driven network client application, such as a web browser, and be exploited by an attacker's content

Honeyclients

- What we need is something which seems like a real browser the same way as a classical honeypot system seems like a real vulnerable server
- A real system (high-interaction honeyclient) or an emulated one (low-interaction honeyclient)?

Document Object Model (DOM)

"*The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page.*"

- Thug DOM is (almost) compliant with W3C DOM Core, HTML, Events and Views specifications (Level 1, 2 and partially 3) and partially compliant with W3C DOM Style specifications
- Designed with the requirement that adding the missing features has to be as simple as possible

Document Object Model (DOM) Browser Personalities

- Drive-by download attacks target specific versions of the browser so a properly designed low-interaction honeyclient should be able to emulate different browser personalities
- Supporting different browser personalities is almost a matter of implementing different (and sometimes totally incompatible) DOM behaviors and interfaces

Supported Browser Personalities – 1/2

- Internet Explorer 6.0 (Windows XP)
- Internet Explorer 6.1 (Windows XP)
- Internet Explorer 7.0 (Windows XP)
- Internet Explorer 8.0 (Windows XP)
- Chrome 20.0.1132.47 (Windows XP)
- Firefox 12.0 (Windows XP)
- Safari 5.1.7 (Windows XP)
- Internet Explorer 6.0 (Windows 2000)
- Internet Explorer 8.0 (Windows 2000)
- Internet Explorer 8.0 (Windows 7)
- Internet Explorer 9.0 (Windows 7)
- Chrome 20.0.1132.47 (Windows 7)
- Firefox 3.6.13 (Windows 7)
- Safari 5.1.7 (Windows 7)

Supported Browser Personalities – 2/2

- Safari 5.1.1 (MacOS X 10.7.2)
- Chrome 19.0.1084.54 (MacOS X 10.7.4)
- Chrome 26.0.1410.19 (Linux)
- Chrome 30.0.1599.15 (Linux)
- Firefox 19.0 (Linux)
- Chrome 18.0.1025.166 (Samsung Galaxy S II, Android 4.0.3)
- Chrome 25.0.1364.123 (Samsung Galaxy S II, Android 4.0.3)
- Chrome 29.0.1547.59 (Samsung Galaxy S II, Android 4.1.2)
- Chrome 18.0.1025.133 (Google Nexus, Android 4.0.4)
- Safari 7.0 (iPad, iOS 7.0.4)

Document Object Model (DOM) Event Handling

- W3C DOM Events specification constitute the most difficult one to emulate because of the (sometimes huge) differences in how different browsers handle events
- Thug emulates the different behaviors of the supported browsers emulating *load* and *mousemove* events by default and allowing to emulate all the other ones if needed

Document Object Model (DOM) Hooks

- Thug defines some DOM hooks which are useful for analyzing well-known exploits
- The next example shows how Thug implements an hook for analyzing a Java exploit with security prompt/warning bypass (CVE-2013-2423)

Document Object Model (DOM) Hooks

```
def _handle_jnlp(self, data, headers):
    try:
        soup = BeautifulSoup.BeautifulSoup(data)
    except:
        return

    if soup.find("jnlp") is None:
        return

    log.ThugLogging.add_behavior_warn(description = '[JNLP Detected]', method = 'Dynamic Analysis')

    for param in soup.find_all('param'):
        log.ThugLogging.add_behavior_warn(description = '[JNLP] %s' % (param, ),
                                          method = 'Dynamic Analysis')
        self._check_jnlp_param(param)

    jar = soup.find("jar")
    if jar is None:
        return

    try:
        url = jar.attrs['href']
        headers['User-Agent'] = self.javaWebStartUserAgent
        response, content = self.window._navigator.fetch(url, headers = headers, redirect_type = "JNLP")
    except:
        pass
```

Javascript

- Google V8 Javascript engine wrapped through PyV8
  - *"V8 implements ECMAScript as specified in ECMA-262, 5th edition, and runs on Windows, Mac OS X , and Linux systems that use IA-32, x64, or ARM processors. The V8 API provides functions for compiling and executing scripts, accessing C++ methods and data structures, handling errors, and enabling security checks"*

  - Abstract Syntax Tree generation and inspection (static analysis)
  - Context inspection (dynamic analysis)
  - Other potentially interesting features (GDB JIT interface, live objects inspection, code disassembler, etc.) exported through a clean and well designed API

Analysis

- Static analysis
  - Abstract Syntax Tree (AST)
- Dynamic analysis
  - V8 debugger protocol
  - Libemu integration (shellcode detection and emulation)

Abstract Syntax Tree (AST)

- Static analysis
  - Static attack signatures
  - Interesting breakpoints identification for later dynamic analysis
  - Symbols identification for later dynamic analysis

- Easily built through V8 API
- Thug AST implementation is quite generic and extensible and allows easily building and inspecting the tree

Vulnerability Modules

- Python-based vulnerability modules
  - ActiveX controls
  - Core browser functionalities
  - Browser plugins

ActiveX

- Thug implements an ActiveX layer of its own for emulating ActiveX controls (just for Internet Explorer personalities)
- It makes use of (Python) vulnerability modules in order to emulate such ActiveX controls or just some of their methods and attributes
- The layer was designed in order to allow adding new ActiveX controls in a fast and easy way

Browser Plugins

- Drive-by download attacks target specific versions of the browser plugins so a properly designed low-interaction honeyclient should be able to emulate different browser plugins versions or to disable them

-A, --adobepdf=	Specify the Adobe Acrobat Reader version (default: 9.1.0)
-P, --no-adobepdf	Disable Adobe Acrobat Reader plugin
-S, --shockwave=	Specify the Shockwave Flash version (default: 10.0.64.0)
-R, --no-shockwave	Disable Shockwave Flash plugin
-J, --javaplugin=	Specify the JavaPlugin version (default: 1.6.0.32)
-K, --no-javaplugin	Disable Java plugin

Logging

- MITRE MAEC logging format
- JSON logging format (contributed by Avira)
- Exploit graph (contributed by Avira)
- “Flat” log files (not so exciting I know)
- MongoDB
- HPFeeds
  - *thug.events* channel (URL analysis results published in MAEC format)
  - *thug.files* channel (downloaded samples)

Logging

Blackhole 2.0 Exploit Kit

{

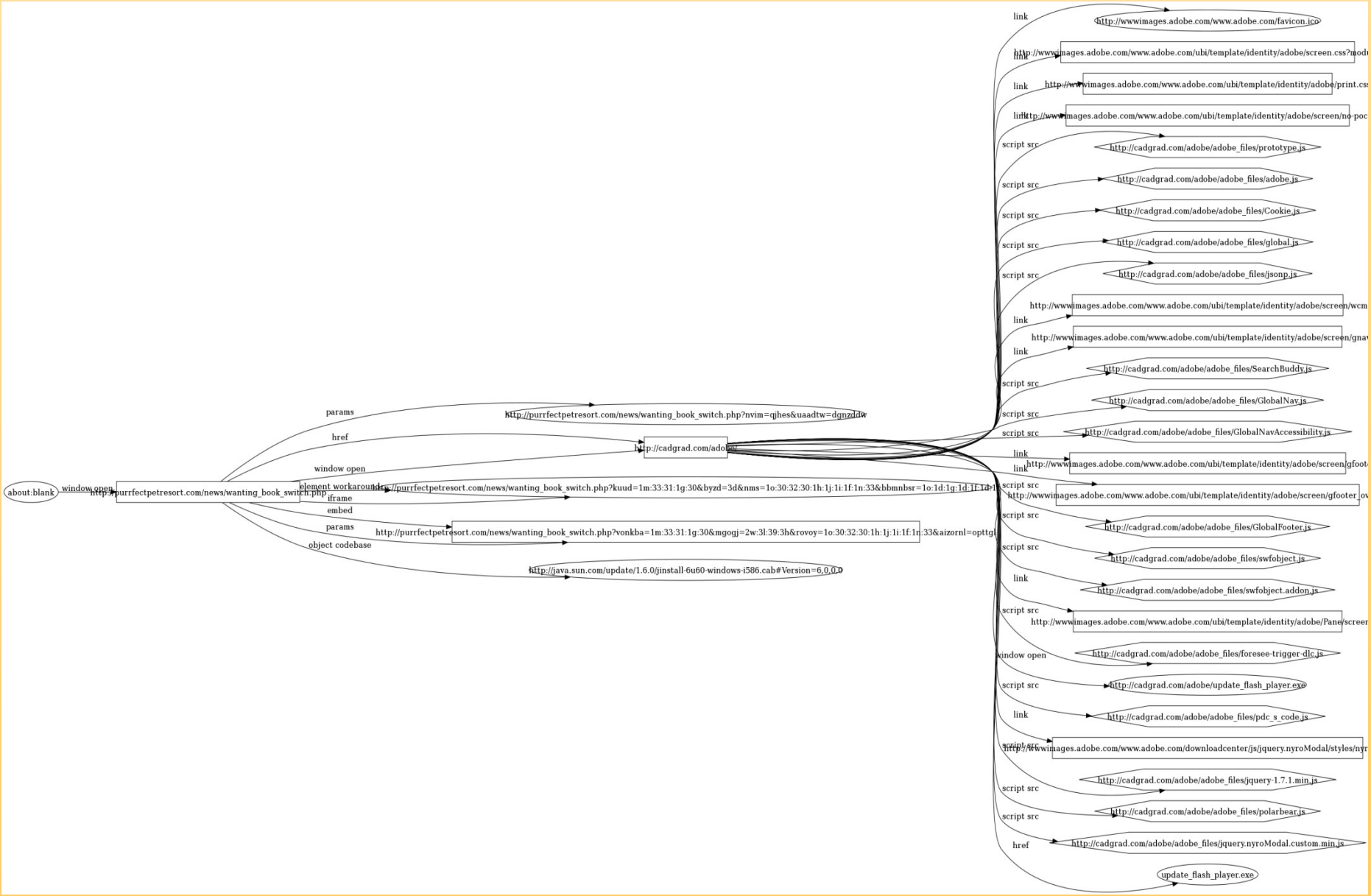
“timestamp”: “2013-04-13 13:43:54.307237”,

“cve”: “None”,

"description": [window open redirection] about:blank-> hxxp://purrfectpetresort.com/news/wanting_book_switch.php“,

"method": "Dynamic Analysis"

}


about:blank
window open
http://purrfectpetresort.com/news/wanting_book_switch.php
params
http://purrfectpetresort.com/news/wanting_book_switch.php?nvim=qjhes&uaadtw=dgnzddw
href
http://cadgrad.com/adobe/
window open
element workaround
http://purrfectpetresort.com/news/wanting_book_switch.php?kuud=1m:33:31:1g:30&byzd=3d&nms=1o:30:32:30:1h:1j:1i:1f:1n:33&bbmnbsr=1o:1d:1g:1d:1f:1a:1
iframe
embed
params
http://purrfectpetresort.com/news/wanting_book_switch.php?vonkba=1m:33:31:1g:30&mgogj=2w:3l:39:3h&rovoy=1o:30:32:30:1h:1j:1i:1f:1n:33&aizornl=opttg(
object codebase
http://java.sun.com/update/1.6.0/jinstall-6u60-windows-i586.cab#Version=6,0,0,0
link
http://wwwimages.adobe.com/www.adobe.com/favicon.ico
http://wwwimages.adobe.com/www.adobe.com/ubi/template/identity/adobe/screen.css?mod
http://wwwimages.adobe.com/www.adobe.com/ubi/template/identity/adobe/print.cs
http://wwwimages.adobe.com/www.adobe.com/ubi/template/identity/adobe/screen/no-poc
script src
http://cadgrad.com/adobe/adobe_files/prototype.js
http://cadgrad.com/adobe/adobe_files/adobe.js
http://cadgrad.com/adobe/adobe_files/Cookie.js
http://cadgrad.com/adobe/adobe_files/global.js
http://cadgrad.com/adobe/adobe_files/jsonp.js
http://wwwimages.adobe.com/www.adobe.com/ubi/template/identity/adobe/screen/wcm
http://wwwimages.adobe.com/www.adobe.com/ubi/template/identity/adobe/screen/gna
http://cadgrad.com/adobe/adobe_files/SearchBuddy.js
http://cadgrad.com/adobe/adobe_files/GlobalNav.js
http://cadgrad.com/adobe/adobe_files/GlobalNavAccessibility.js
http://wwwimages.adobe.com/www.adobe.com/ubi/template/identity/adobe/screen/gfoot
http://wwwimages.adobe.com/www.adobe.com/ubi/template/identity/adobe/screen/gfooter_ov
http://cadgrad.com/adobe/adobe_files/GlobalFooter.js
http://cadgrad.com/adobe/adobe_files/swfobject.js
http://cadgrad.com/adobe/adobe_files/swfobject.addon.js
http://wwwimages.adobe.com/www.adobe.com/ubi/template/identity/adobe/Pane/screen
http://cadgrad.com/adobe/adobe_files/foresee-trigger-dlc.js
window open
http://cadgrad.com/adobe/update_flash_player.exe
http://cadgrad.com/adobe/adobe_files/pdc_s_code.js
http://wwwimages.adobe.com/www.adobe.com/downloadcenter/js/jquery.nyroModal/styles/nyr
http://cadgrad.com/adobe/adobe_files/jquery-1.7.1.min.js
http://cadgrad.com/adobe/adobe_files/polarbear.js
http://cadgrad.com/adobe/adobe_files/jquery.nyroModal.custom.min.js
update_flash_player.exe

Logging


```
{
    "mimetype": "PE32 executable (GUI) Intel 80386 (stripped to external PDB), for MS Windows",
    "url": "hxxp://cadgrad.com/adobe/update_flash_player.exe",
    "flags": {},
    "sha256": "d59d9af4e9ec25431acfd8938895b5c3b728d818db024d76f5aa265e0b171f4f",
    "content-type": "application/octet-stream",
    "md5": "a3266663f644dc0c0df42e8da1404878",
    "size": 130560
}
```

Classifiers

- Classifiers support was introduced in Thug 0.4.24 and is based on Yara signatures

- Currently two classifiers exist:
  - URL classifier
  - Javascript classifier

URL Classifier

The URL classifier works on URL pattern matching trying to identify typical exploit kits URL i.e.

```
rule Blackhole_V2_2 : Exploit_Kit {
    meta:
            author = "Thorsten Sick"
    strings:
            $url = /\/closest\/\w{15,35}.php/ nocase
    condition:
            $url
}
```

Javascript Classifier

The Javascript classifier exploits the idea that even if the code is obfuscated Thug goes through all the deobfuscation stages. Working this way it can catch details which does not change so frequently in a typical exploit kit i.e.

```
rule PluginDetect : Multiple_Exploit_Kits {
    meta:
                author = "Angelo Dell'Aera"
    strings:
                $jar = "getjavainfo.jar" nocase
                $pdpd = "pdpd" nocase
                $getver = "getversion" nocase
    condition:
                ($jar or $pdpd) and $getver
}
```

Source code

Thug source code is publicly available at

https://github.com/buffer/thug

Contributions, comments and feedback welcome!

Thanks for the attention!

Questions?

Angelo Dell'Aera
<angelo.dellaera@honeynet.org>