HOW I MET YOUR BROWSER: GOING INCognito DOESN’T HIDE YOUR BROWSING FROM RAGAMUFFIN

#OSDFCon

OSDFCON 2017 – HERNDON (VIRGINIA)
AGENDA

- Introduction
  - What is Chrome Ragamuffin
  - What Chrome Ragamuffin is not
  - Why Chrome Ragamuffin should be useful
  - Goal
  - Tools of the trade

- Forensic Overview
  - Chrom(e|ium) overview
  - Objects we have focused on
  - Case Study
  - Chrome Ragamuffin architecture
  - State of art
  - Work in progress and future works
WHAT IS CHROME RAGAMUFFIN?

• Internal research project
• It is a memory forensics tool
• Three main functionalities:
  • Detailed overview about user navigation
  • Web page contents
  • Client-site attacks traces
WHAT CHROME RAGAMUFFIN IS NOT

- IDS/NIDS
- Browser extension
- Anti-virus
- SQLite databases analyzer
WHY CHROME RAGAMUFFIN SHOULD BE USEFUL

- Most used browser
- We hadn’t tools to analyze web browser internals
- It can find a lot of new artifacts
TOOLS OF THE TRADE

- Source code analysis
- Debugging and data structures analysis
- Data structures extraction
- to implement the Proof of Concept
FORENSIC OVERVIEW
GOOGLE CHROME OVERVIEW

Browser Process

Renderer

Blink

V8

Renderer

Blink

V8
OBJECTS WE HAVE FOCUSED ON
Browser process

Visible page

ifram1

ifram2

WebContents

NavigationController

entries

NavigationEntry

NavigationEntry

NavigationEntry

frame_tree

ifram1

ifram2

ifram3

ifram4
OBJECTS WE HAVE FOCUSED ON
Renderer process

Blink::Document

url_
title_
...
document_element_
...

Document Object Model

html
  head
  body
  Text
  div
  iframe
  p
  form

Blink::HTMLIFrameElement

m_tagName_
...
m_contentFrame_
m_URL_
...
OBJECTS WE HAVE FOCUSED ON
Browser process

net::HttpCache

mode_

disk_cache_

disk_cache:: MemBackendImpl

unordered_map MemoryEntryImpl

entries_

max_size_

current_size_

key
... 
...
... 
...
... 
...
... 
...
...

value
... 
...
... 
...
... 
...
... 
...
...
# OBJECTS WE HAVE FOCUSED ON

## Example 1

### Browser objects:

- Evidence: offset object, url, status code, method, transition, timestamp, restore type, page type, form params

<table>
<thead>
<tr>
<th>ID</th>
<th>Offset</th>
<th>Title</th>
<th>User typed url</th>
<th>Original request url</th>
<th>Status code</th>
<th>Method</th>
<th>PageState address</th>
<th>Transition</th>
<th>Referer</th>
<th>UTC Timestamp</th>
<th>Restore Type</th>
<th>Page type</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2.57191E+12</td>
<td>None</td>
<td><a href="http://192.168.0.11/example1.html">http://192.168.0.11/example1.html</a></td>
<td><a href="http://192.168.0.11/example1.html">http://192.168.0.11/example1.html</a></td>
<td>200</td>
<td>GET</td>
<td>None</td>
<td>User used the address bar to trigger this navigation</td>
<td>None</td>
<td>10/1/2017 16:12</td>
<td>Entry was not restored</td>
<td>NORMAL</td>
</tr>
</tbody>
</table>
OBJECTS WE HAVE FOCUSED ON

Example 1

Renderer objects:

- Evidence: PID of the **tab** which **contains** the **specific document**, document **offset**, URL of the document, **title**, `<html>` node address of the DOM tree
OBJECTS WE HAVE FOCUSED ON
Example 1

Renderer objects:
Evidence: By the “DOM start address” field we can get the entire Document Object Model tree in its dot (high-level structure of the page) and text notation (detailed contents)

Node tag: iframe
Node attributes: {'src': 'iframe1.html'}
Memory offset: 0x3e7ca5a3650
Contained document offset: 0x3e7ca5aa588

Node tag: iframe
Node attributes: {'src': 'iframe2.html'}
Memory offset: 0x3e7ca5a4570
Contained document offset: 0x3e7ca5a9220
# OBJECTS WE HAVE FOCUSED ON

**Cross-Site Request Forgery**

**Browser objects:**
- Evidence: offset object, url, status code, method, transition, timestamp, restore type, page type, form params

<table>
<thead>
<tr>
<th>Entry ID</th>
<th>Offset</th>
<th>Title</th>
<th>User typed url</th>
<th>Original request url</th>
<th>Status code</th>
<th>Method</th>
<th>PageState address</th>
<th>Transition</th>
<th>Referrer</th>
<th>UTC Timestamp</th>
<th>Restore type</th>
<th>Page type</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0x192d2a03c60</td>
<td>None</td>
<td><a href="http://192.168.1.124/notexists.html">http://192.168.1.124/notexists.html</a></td>
<td><a href="http://192.168.1.124/notexists.html">http://192.168.1.124/notexists.html</a></td>
<td>0</td>
<td>GET</td>
<td>None</td>
<td>Typed URL in the address bar</td>
<td>None</td>
<td>18/09/17 14.50</td>
<td>Entry was not restored</td>
<td>ERROR</td>
</tr>
<tr>
<td>3</td>
<td>0x192d2a02900</td>
<td>Test</td>
<td><a href="http://192.168.1.124/in">http://192.168.1.124/in</a> dex.html</td>
<td><a href="http://192.168.1.124/in">http://192.168.1.124/in</a> dex.html</td>
<td>200</td>
<td>GET</td>
<td>None</td>
<td>Typed URL in the address bar</td>
<td>None</td>
<td>18/09/17 14.50</td>
<td>Entry was not restored</td>
<td>NORMAL</td>
</tr>
</tbody>
</table>
OBJECTS WE HAVE FOCUSED ON
Cross-Site Request Forgery

POST Params:

- With the memory address, we can dump the PageState object which contains serialized information about the submitted form
# OBJECTS WE HAVE FOCUSED ON

Cross-Site Request Forgery

**Renderer objects:**

- Evidence: PID of the **tab** which **contains** the **specific document**, document **offset**, **URL** of the document, **title**, `<html>` node address of the DOM tree

<table>
<thead>
<tr>
<th>Pid</th>
<th>Document offset</th>
<th>URL</th>
<th>Title</th>
<th>DOM start address</th>
</tr>
</thead>
<tbody>
<tr>
<td>4916</td>
<td>0x11dad2f8380</td>
<td><a href="https://www.google.it/.../newtab?espv=2&amp;ie=UTF-8">https://www.google.it/.../newtab?espv=2&amp;ie=UTF-8</a></td>
<td>None</td>
<td>0x11dad2f9830</td>
</tr>
<tr>
<td>4916</td>
<td>0x11dad4825c0</td>
<td>chrome-search://most-visited%28questa%28pagina</td>
<td>None</td>
<td>0x11dad4825c0</td>
</tr>
<tr>
<td>2222</td>
<td>0x3416fde25c0</td>
<td>chrome://apps/</td>
<td>Nuova scheda</td>
<td>0x3416fde25c0</td>
</tr>
<tr>
<td>3268</td>
<td>0x31da1fe25c0</td>
<td><a href="https://it-it.facebook.com/">https://it-it.facebook.com/</a></td>
<td>Facebook: accedi o iscriviti</td>
<td>0x31da1fe25c0</td>
</tr>
<tr>
<td>3268</td>
<td>0x31da21101e98</td>
<td><a href="https://static.xx.facebook...ommon/refer_frame.php">https://static.xx.facebook...ommon/refer_frame.php</a></td>
<td>None</td>
<td>0x31da21101e98</td>
</tr>
<tr>
<td>5416</td>
<td>0x1e1842825c0</td>
<td>chrome://settings/</td>
<td>Impostazioni</td>
<td>0x1e1842825c0</td>
</tr>
<tr>
<td>5416</td>
<td>0x1e184327ab8</td>
<td>about:blank</td>
<td>None</td>
<td>0x1e184327ab8</td>
</tr>
<tr>
<td>5416</td>
<td>0x1e184329680</td>
<td>about:blank</td>
<td>None</td>
<td>0x1e184329680</td>
</tr>
<tr>
<td>5416</td>
<td>0x1e18432b248</td>
<td>about:blank</td>
<td>None</td>
<td>0x1e18432b248</td>
</tr>
<tr>
<td>3212</td>
<td>0x21661d41838</td>
<td>None</td>
<td>PROVA</td>
<td>0x21661d41838</td>
</tr>
<tr>
<td>3212</td>
<td>0x21661d425c0</td>
<td><a href="http://192.168.1.124/index.html">http://192.168.1.124/index.html</a></td>
<td>None</td>
<td>0x21661d425c0</td>
</tr>
</tbody>
</table>
OBJECTS WE HAVE FOCUSED ON
what do we get from those?

Renderer objects:
Evidence: By the “DOM start address” field we can get the entire Document Object Model tree in its dot (high-level structure of the page) and text notation (detailed contents)

Node tag: iframe
Node attributes: {'src': 'hgrueguoeir.html', 'style': 'display: none;'}
Memory offset: 0x216b1d43650
Contained document offset: 0x216b1d41838

- Node tag: html
  Node attributes: {lang=it}
  Memory offset: 0x2669a628b8b8
- Node tag: head
  Node attributes: {}
  Memory offset: 0x2669a628b920
- Node tag: body
  Node attributes: {class=test, id=123}
  Memory offset: 0x2669a628b980
- Node tag: h1
  Node attributes: {id=title}
  Memory offset: 0x2669a628bf9f0
- Node tag: Text
  Content: You've successfully changed your password
CHROME RAGAMUFFIN ARCHITECTURE

Implemented in two parts:

1. *libchrome_${release}.py* library
   - VTypes
   - WTF::StringImpl, Vector objects and other platform-specific data types

2. *chrome_ragamuffin.py* plugin
   - This is the main plugin.
   - signatures, validation, render the output
STATE OF ART

Other tools

- WebCapsule/ChromePic
  - **Instrumentation** of the web browser source code
  - Records and Replay **key logger**

- Chrome History (@superponible)
  - **SQLite databases** in memory (visited pages, cookies, search terms ecc.)
  - SQLite databases are **saved on disk**
STATE OF ART

Chrome Ragamuffin

• Pro
  o Agnostic approach
  o Whole address space (a lot of new artifacts)
  o Overcoming incognito mode
STATE OF ART

Chrome Ragamuffin

• Limitations
  o Garbage Collector (Olipan, Scavenger ecc.) collects unused objects
WORK IN PROGRESS

- HTTP cached Response Body (Work in progress)
- V8 (for now, Isolate, Heap, Spaces, Page Memories) (Almost-Work-in-progress)
- Automatic VTypes extraction
- Linux/macOSx support
THANKS

• Join the project on github! Search for cube0x8 ("cube" "zero" "x" "eight") (https://github.com/cube0x8/chrome_ragamuffin)

• Email: alessandro.devito@truel.it