OSXCollector
Automated forensic evidence collection & analysis for OS X

Jakub (Kuba) Sendor
@jsendor
whoami

- Joined Yelp security team in July 2014.
- Mostly involved in malware incident response.
- Also working on automating our security processes.
- Previously worked at SAP in Sophia Antipolis (France) in the Security & Trust research group.
- Graduated in 2011 from AGH University of Science and Technology in Kraków (Poland) and Telecom ParisTech/Institut Eurecom (France).
Yelp’s Mission:
Connecting people with great local businesses.
Yelp Stats:
As of Q2 2015

83M
83M
68%
32
>3k employees, most of them using Macs
OS X Grumpy Cat

Introducing OS X 10.FU
The world's most advanced operating system just got Grumpier
https://github.com/Yelp/osxcollector

**OSXCollector** is an open source forensic evidence collection & analysis toolkit for Mac OS X.
OSXCollector is easy to run

1 Python file
0 dependencies

$ sudo osxcollector.py --id DelayedHedgehog
Wrote 35394 lines.
Output in DelayedHedgehog-2015_01_20-19_38_38.tar.gz
$

Megan Carney @PwnieFan · Jan 13
Best line from osxcollector documentation: "Get creative with incident names, it makes it easier to laugh through the pain."
The output is JSON

JSON is beautiful.

JSON is easy to manipulate.

```json
{
  "file_path": "/System/Library/Extensions/Apple_iSight.kext/Contents/MacOS/Apple_iSight",
  "sha2": "19b7b85eaedb17d9565dce872f0d1ea8fc0761f508f28bedcc8606b828cbf614",
  "sha1": "99005b68295c202fd359b46cd1411acea96b2469",
  "md5": "b8cc164b6546e4b13768d8353820b216",
  "ctime": "2014-12-05 16:50:39",
  "mtime": "2014-09-19 00:16:50",
  "osxcollector_section": "kext",
  "osxcollector_incident_id": "DelayedHedgehog-2015_01_20-19_38_38",
  "osxcollector_plist_path": "/System/Library/Extensions/Apple_iSight.kext/Contents/Info.plist",
  "osxcollector_bundle_id": "com.apple.driver.Apple_iSight",
  "signature_chain": [
    "Software Signing",
    "Apple Code Signing Certification Authority",
    "Apple Root CA"
  ]
}
```
OS X stores lots of data in SQLite DBs

```python
# Dump a sqlite DB in a dozen lines of code
with connect(sqlite_db_path) as conn:
    conn.cursor.execute('SELECT * from sqlite_master WHERE type = "table"')
    table_names = [table[2] for table in tables.fetchall()]

for table in table_names:
    rows = conn.cursor.execute('SELECT * from {0}'.format(table_name))
    column_descriptions = [col[0] for col in conn.cursor.description]
    for row in rows.fetchall():
        record = dict([(key, val) for key, val in zip(column_descriptions, row)])
```

@jsendor
plist == property list
sometimes binary, sometimes plain text

$ /usr/libexec/PlistBuddy -c print shell.plist
Dict {
    ProgramArguments = Array {
        /usr/libexec/rshd
    }
    Sockets = Dict {
        Listeners = Dict {
            SockServiceName = shell
        }
    }
    Disabled = true
    Label = com.apple.rshd
    SessionCreate = true
    inetdCompatibility = Dict {
        Wait = false
    }
}

$ cat ssh.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0 //EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
    <dict>
        <key>Disabled</key>
        <true/>
        <key>Label</key>
        <string>com.openssh.sshd</string>
        <key>Program</key>
        <string>/usr/libexec/sshd-keygen-wrapper</string>
        <key>ProgramArguments</key>
        <array>
            <string>/usr/sbin/sshd</string>
            <string>-i</string>
        </array>
    </dict>
</plist>
OSXCollector uses Foundation

Foundation is a nice Objective-C wrapper.

```
import Foundation

# Look! Incredibly long objc style function names!
plist_nsdata, error_message = Foundation.NSData.dataWithContentsOfFile_options_error_(plist_path, Foundation.NSUncachedRead, None)

# Seriously, incredibly long function names!
plist_dict, _, _ = Foundation:NSPropertyListSerialization.propertyListFromData_mutabilityOption_format_errorDescription_(plist_nsdata, Foundation.NSPropertyListMutableContainers, None, None)
```
Forensic Collection

OS System Info  Applications  Web Browser Info

Kernel Extensions  Quarantines  Email Info

Downloads  Startup Items  Groups & Accounts
Common keys in entries

path, hashes, timestamps, signature chain, ...

```json
{
  "file_path": "/System/Library/Extensions/Apple_iSight.kext/Contents/MacOS/Apple_iSight",
  "sha2": "19b7b85eaedbe7d9565dce872f0d1ea8fc0761f508f28bedcc8606b828cbf614",
  "sha1": "99005b68295c202fd359b46cd1411acea96b2469",
  "md5": "b8cc164b6546e4b13768d8353820b216",
  "ctime": "2014-12-05 16:50:39",
  "mtime": "2014-09-19 00:16:50",
  "osxcollector_section": "kext",
  "osxcollector_incident_id": "DelayedHedgehog-2015_01_20-19_38_38",
  "osxcollector_plist_path": "/System/Library/Extensions/Apple_iSight.kext/Contents/Info.plist",
  "osxcollector_bundle_id": "com.apple.driver.Apple_iSight",
  "signature_chain": [
    "Software Signing",
    "Apple Code Signing Certification Authority",
    "Apple Root CA"
  ]
}```
Startup items run on boot

Malware running at startup is basically game over.

```
{
  "osxcollector_section": "startup",
  "osxcollector_subsection": "launch_agents",
  "md5": "dbd251d8a6e4da2419d75f5b18cf5078",
  "sha1": "bbb8016ad1026aea499fd47e21ffe95f9597aca",
  "sha2": "9c89666fd071abd203f044ab7b3fd416decafe4468ff2e28d72f94809e2",
  "file_path": "/Library/Application Support/GPGTools/uuid-patcher",
  "ctime": "2014-12-05 16:52:00",
  "mtime": "2014-11-30 15:49:40",
  "osxcollector_plist": "/System/Library/LaunchDaemons/ssh.plist",
  "program": "/usr/libexec/sshd-keygen-wrapper",
  "label": "com.openssh.sshd",
  "signature_chain": [],
  "osxcollector_incident_id": "DelayedHedgehog-2015_01_20-19_38_38",
}
```
Timestamps are important in forensics

Timestamps get stored in a lot of ways. OSXCollector normalizes them.

```
{
    "file_path": "/System/Library/Extensions/Apple_iSight.kext/Contents/MacOS/Apple_iSight",
    "sha2": "19b7b85eaed817d9565dce872f0d1ea8fc0761f508f28bedcc8606b828cbf614",
    "sha1": "99005b68295c202fd359b46cd1411ace96b2469",
    "md5": "b8cc164b6546e4b13768d8353820b216",
    "ctime": "2014-12-05 16:50:39",
    "mtime": "2014-09-19 00:16:50",
    "osxcollector_section": "kext",
    "osxcollector_incident_id": "DelayedHedgehog-2015_01_20-19_38_38",
    "osxcollector_plist_path": "/System/Library/Extensions/Apple_iSight.kext/Contents/Info.plist",
    "osxcollector_bundle_id": "com.apple.driver.Apple_iSight",
    "signature_chain": [
        "Software Signing",
        "Apple Code Signing Certification Authority",
        "Apple Root CA"
    ]
}
```
Hashes are still important in forensics

```json
{
  "file_path": "/System/Library/Extensions/Apple_iSight.kext/Contents/MacOS/Apple_iSight",
  "sha2": "19b7b85eaed17d9565dce872f0d1ea8fc0761f508f28bedcc8606b828cbf614",
  "sha1": "99005b68295c202fd359b46cd1411acea96b2469",
  "md5": "b8cc164b6546e4b13768d8353820b216",
  "ctime": "2014-12-05 16:50:39",
  "mtime": "2014-09-19 00:16:50",
  "osxcollector_section": "kext",
  "osxcollector_incident_id": "DelayedHedgehog-2015_01_20-19_38",
  "osxcollector_plist_path": "/System/Library/Extensions/Apple_iSight.kext/Contents/Info.plist",
  "osxcollector_bundle_id": "com.apple.driver.Apple_iSight",
  "signature_chain": [
    "Software Signing",
    "Apple Code Signing Certification Authority",
    "Apple Root CA"
  ]
}
```
Quarantines track downloaded content

They live forever in a plist.

```json
{
  "osxcollector_section": "quarantines",
  "osxcollector_username": "jsendor",
  "LSQuarantineAgentName": "Google Chrome",
  "LSQuarantineAgentBundleIdentifier": "com.google.Chrome",
  "LSQuarantineDataURLString": "https://cachefly.alfredapp.com/Alfred_2.5.1_308.zip",
  "LSQuarantineEventIdentifier": "6FA87446-1249-4578-83E4-4BBCF7AEA4A3",
  "LSQuarantineOriginURLString": "http://www.alfredapp.com/",
  "osxcollector_db_path": "/Users/ivanlei/Library/Preferences/com.apple.LaunchServices.QuarantineEventsV2",
  "osxcollector_table_name": "LSQuarantineEvent",
  "osxcollector_incident_id": "DelayedHedgehog-2015_01_20-19_38_38",
  "LSQuarantineTimeStamp": "2014-12-05 14:40:33"
}
```
xattr-wherefrom

No need to search around in browser history.

```
{
  ..
  "md5": "0b984ecc39d5b33e4f6a81ade4e8dbf1",
  "xattr-quarantines": [
    "0001:5541127e;Google Chrome;63B2C485-1F64-4ADE-A95C-72F7087FA172"
  ],
  "signature_chain": [],
  "xattr-wherefrom": [
    "http://trojans.evildownloads.com/Trojan.app",
    "http://trojans.evildownloads.com/latest-trojans/
  ],
  "osxcollector_incident_id": "DelayedHedgehog-2015_01_20-19_38_38",
  "file_path": "/Users/jdoe/Downloads/Trojan.app",
}
```
OS X doesn't care if startups and kext are signed

But I kinda do, so OSXCollector lists the signature chain.

```json
{
    "osxcollector_section": "startup",
    "osxcollector_subsection": "launch_agents",
    "md5": "dbd251d8a6e4da2419d75f5b18cf5078",
    "sha1": "bb8016ad1026ae499f4d7e21ffeb9f9597aca",
    "sha2": "9c89666fd071abd203f044ab7b3fd416decafe4468ff2e20a50b6d72f94809e2",
    "file_path": "/Library/Application Support/GPGTools/uuid-patcher",
    "ctime": "2014-12-05 16:52:00",
    "mtime": "2014-11-30 15:49:40",
    "osxcollector_plist": "/System/Library/LaunchDaemons/ssh.plist",
    "program": "/usr/libexec/sshd-keygen-wrapper",
    "label": "com.openssh.sshd",
    "signature_chain": [],
    "osxcollector_incident_id": "DelayedHedgehog-2015_01_20-19_38_38"
}
```
Forensic collection is hard work.

Forensic analysis is fun.

Part science, part art.
Manual analysis with `grep` and `jq` works pretty well

grep a time window

$ cat foo.json | grep '2014-01-01 11:3[2-8]'  

only urls in a time window

$ cat foo.json | grep '2014-01-01 11:3[2-8]' | jq 'select( has("url")).url'

grep a single user

$ cat INCIDENT32.json | jq 'select( .osxcollector_username=="jsendor")|.'

@jsendor
We can automate this!

step 1: analyze

We can automate this!

step 2: ???

step 3: profit

Well, here's some domains OpenDNS wouldn't recommend.

...  
- quarantines
  LSQuarantineDataURLString: "http://d2.genieo.com/im/partners/webpic2/installgenieo.dmg?campaign=wbpc_1&download_browser=Chrome"
  LSQuarantineTimeStamp: "2014-04-30 15:26:13"
  opendns-categorization: {"status": 0, "content_categories": ["Adware"], "suspicious": True, "security_categories": []}
  opendns-security: {"dga_score": -6.35631605112, "rip_score": 0.0, "asn_score": 0.0, "securerank2": -0.00813742053751, "attack": ",", "prefix_score": 0.0, "found": True, "threat_type": ","}
  opendns-link: "https://investigate.opendns.com/domain-view/name/w.genieo.com/view"

...  
- firefox history
  last_visit_date: "2015-01-11 23:44:56"
  url: "http://dl.pspvideosdownload.com/lp/?appid=12...
  vtdomain-domain: "dl.pspvideosdownload.com"
  vtdomain-detections: {"undetected_referrer_samples": 0, "detected_downloaded_samples": 2, "detected_referrer_samples": 0, "detected_urls": 100, "detected_communicating_samples": 0, "undetected_communicating_samples": 0}
Enter OSXCollector Output Filters
Automated analysis with output filters

- JSON in
  - find domains filter
  - check blacklists filter
  - construct browser history filter
  - VirusTotal hash lookup filter
  - OpenDNS domain reputation filter
  - Shadowserver hash lookup filter

- JSON out
  - recommend next steps filter
  - OpenDNS related domains filter
  - find related files filter

@jsendor
Automated analysis with output filters

- JSON in
  - find domains filter
  - check blacklists filter
  - construct browser history filter

- find related files filter
  - Shadowserver hash lookup filter
  - VirusTotal hash lookup filter

- JSON out
  - recommend next steps filter
  - OpenDNS domain reputation filter
  - OpenDNS related domains filter

- VirusTotal domain reputation filter
find domains filter

{  "url": "https://biz.yelp.com"
}

{  "url": "https://biz.yelp.com",
   "osxcollector_domains": [   "biz.yelp.com",
   "yelp.com"
  ]
}

a lot of filters add a single piece of info
Automated analysis with output filters

- JSON in
  - find domains filter
  - check blacklists filter
  - construct browser history filter
  - VirusTotal hash lookup filter
  - OpenDNS domain reputation filter
  - Shadowserver hash lookup filter
  - VirusTotal domain reputation filter
  - find related files filter
  - OpenDNS related domains filter

- JSON out
  - recommend next steps filter

@jsendor
check blacklist filter

Match any key.
Regex or exact match.
Built in smarts for turning domains into regex.

domain_blacklist.txt

- evil.com
- streaming-football.com
- downloads.com
Automated analysis with output filters

- JSON in
  - find domains filter
  - check blacklists filter
  - VirusTotal hash lookup filter
  - Shadowserver hash lookup filter
  - find related files filter

- JSON out
  - recommend next steps filter
  - construct browser history filter
  - OpenDNS domain reputation filter
  - VirusTotal domain reputation filter
  - OpenDNS related domains filter
VirusTotal hash lookup filter

API output filter base does the heavy lifting.
Support for rate limits & response caching issues
10s of requests at once.

@jsendor
Automated analysis with output filters

- JSON in
  - find domains filter
  - check blacklists filter
  - VirusTotal hash lookup filter
  - Shadowserver hash lookup filter
  - find related files filter
  - OpenDNS domain reputation filter
- JSON out
  - recommend next steps filter
  - construct browser history filter
  - OpenDNS domain reputation filter
  - VirusTotal domain reputation filter
  - OpenDNS related domains filter

@jsendor
OpenDNS related domains filter

{ "url": "https://www.evil.com",
  "osxcollector_related": {
    "domains": ["double-evil.com", "free-lunch.org", "torrent-malware.net"]
  }
}

Judge domains by the company they keep.
Domains related to suspicious domains are usually suspicious themselves.
Automated analysis with output filters

- **JSON in**
  - find domains filter
  - check blacklists filter
  - create browser history filter
  - recommend next steps filter

- **check blacklists filter**

- **VirusTotal hash lookup filter**

- **Shadowserver hash lookup filter**

- **find related files filter**

- **OpenDNS domain reputation filter**

- **VirusTotal domain reputation filter**

- **OpenDNS related domains filter**

- **JSON out**
OpenDNS domain reputation filter

Premium Cyber Threat Intel (CTI)

{  
  "url": "https://www.evil.com",
  "osxcollector_opendns": {
    "domain": "evil.com",
    "security": {
      "found": true,
      "dga_score": -3,
      "securerank2": -23,
      "asn_score": -57,
      "prefix_score": -62,
      "rip_score": -99
    }
  }
}
Automated analysis with output filters

- JSON in
  - find domains filter
  - check blacklists filter
  - construct browser history filter
- recommend next steps filter
- JSON out
  - VirusTotal hash lookup filter
  - OpenDNS domain reputation filter
  - Shadowserver hash lookup filter
  - find related files filter
  - OpenDNS related domains filter
Recommend next steps

This whole things started with just a few clues. Now look what I found.
- downloads
  - ctime: "2015-02-02 12:15:14"
  - file_path: "/Users/jdoe/Downloads/screenshot.scr"
  - mtime: "2015-01-16 19:20:06"
- xattr-quarantines: ["0001;54b95657;Google\x20Chrome;162C4043-647D-44A8-83C2-2B1F69C7861F"]
- xattr-wherefrom: ["https://evildownloads.com/docs/securesc/5552qjr011ks3i1r65nm9vjn073v4ahg/82mfdn9k8qmvo3ta2vja6hta3iink5i/1421431200000/00218636334715341180/12229357981017199890/0B-HDNU1GNnRAVjBtY1BqdVFrT2sf\ndownload\&h=01562916784096941731\&nonce=850uav3g55qiu\&user=12229357981017199890\&hash=78ffvfofh7rreq0bj86hqfhb7i8eq921", ""]
- related-files: ["screenshot.scr"]

Nothing hides from Very Readable Output Bot

If I were you, I'd probably update my blacklists to include:
- domain: "evildownloads.com"

That might just help things, Skippy!
Automated analysis with output filters

1. JSON in
   - find domains filter
2. JSON out
   - recommend next steps filter
3. check blacklists filter
4. construct browser history filter
5. VirusTotal hash lookup filter
6. OpenDNS domain reputation filter
7. Shadowserver hash lookup filter
8. VirusTotal domain reputation filter
9. OpenDNS related domains filter
10. find related files filter

@jsendor
Threat Intel API

https://github.com/Yelp/threat_intel

Query Threat Intel Feeds:

- VirusTotal
- OpenDNS
- Shadowserver
from threat_intel.opendns import InvestigateApi
investigate = InvestigateApi(<INVESTIGATE-API-KEY-HERE>, cache_file_name="/tmp/cache.opendns.json")

domains = ["google.com", "baidu.com", "bibikun.ru"]
investigate.security(domains)

{
  "baidu.com": {
    "found": true,
    "dga_score": 0,
    "rip_score": 0,
    ...
  }
}
ElastAlert

http://engineeringblog.yelp.com/
https://github.com/Yelp/osxcollector

Lemme know if you use it.
Send pull requests.