Incident Response tactics with Compromise Indicators

Vladimir Kropotov, Vitaly Chetvertakov, Fyodor Yarochkin
RuCrypto 2014

March 25, 2014

Outline

Introduction

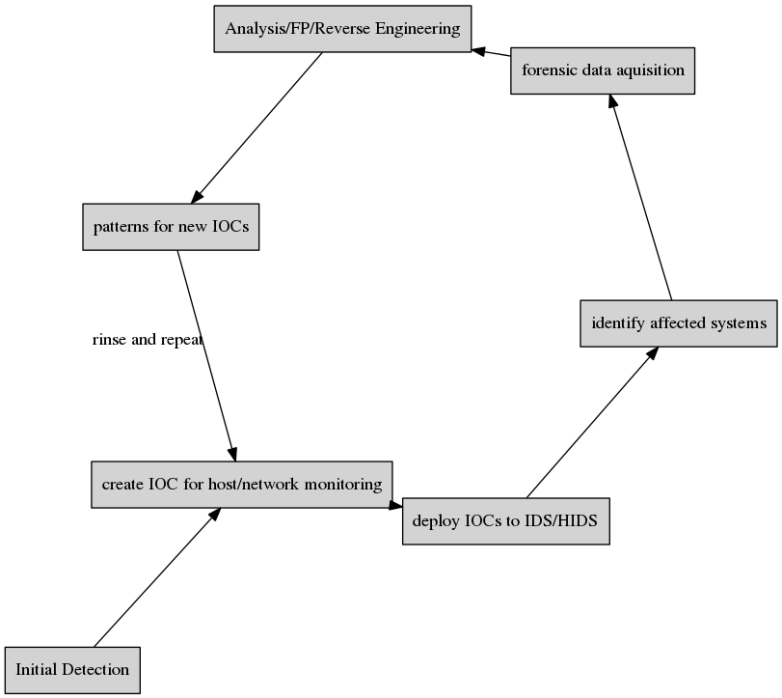
Indicators of Compromise

Indicator of compromise (IOC) in computer forensics is an artifact observed on network or in operating system that with high confidence indicates a computer intrusion.

`http://en.wikipedia.org/wiki/Indicator_of_compromise`

IOC workflow

A typical flow with Indicators of Compromise allows to integrate dynamic threat intelligence into detection process:



source: Sophisticated indicators for the modern threat landscape, 2012 paper

Standards: OpenIOC

OpenIOC - Mandiant-backed effort for unform representation of IOC (now FireEye) `http://www.openioc.org/`

```
-<ioc id="6d2a1b03-b216-4cd8-9a9e-8827af6ebf93" last-modified="2011-10-28T19:28:20">
   <short_description>Zeus</short_description>
   <description>Finds Zeus variants, twexts, sdra64, ntos</description>
   <keywords/>
   <authored_by>Mandiant</authored_by>
   <authored_date>0001-01-01T00:00:00</authored_date>
   <links/>
  -<definition>
   -<Indicator operator="OR" id="9c8df971-32a8-4ede-8a3a-c5cb2c1439c6">
    -<Indicator operator="AND" id="0781258f-6960-4da5-97a0-ec35fb403cac">
     -<IndicatorItem id="50455b63-35bf-4efa-9f06-aeba2980f80a" condition="contains">
        <Context document="ProcessItem" search="ProcessItem/name" type="mir"/>
        <Content type="string">winlogon.exe</Content>
      </IndicatorItem>
     -<IndicatorItem id="b05d9b40-0528-461f-9721-e31d5651abdc" condition="contains">
        <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Type" type="mir"/>
        <Content type="string">File</Content>
      </IndicatorItem>
     -<Indicator operator="OR" id="67505775-6577-43b2-bccd-74603223180a">
      -<IndicatorItem id="c5ae706f-c032-4da7-8acd-4523f1dae9f6" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Name" type="mir"/>
          <Content type="string">system32\sdra64.exe</Content>
        </IndicatorItem>
      -<IndicatorItem id="25ff12a7-665b-4e45-8b0f-6e5ca7b95801" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Name" type="mir"/>
          <Content type="string">system32\twain_32\user.ds</Content>
        </IndicatorItem>
      -<IndicatorItem id="fea11706-9ebe-469b-b30a-4047cfb7436b" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Type" type="mir"/>
          <Content type="string">\WINDOWS\system32\twext.exe</Content>
        </IndicatorItem>
```

Standards: Mitre

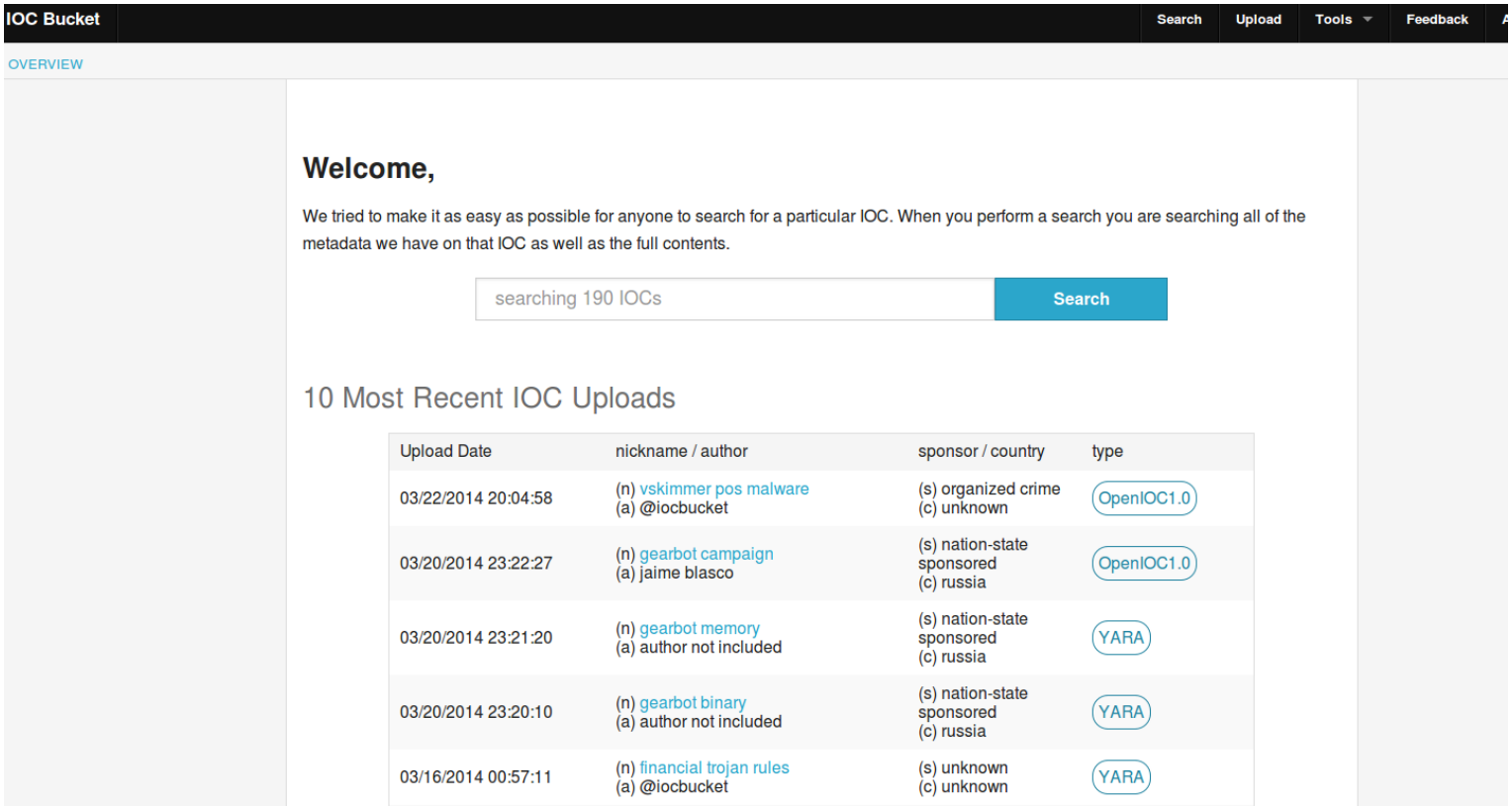
Mitre CybOX: `http://cybox.mitre.org/`
`https://github.com/CybOXProject/Tools`
`https://github.com/CybOXProject/openioc-to-cybox` Mitre CAPEC: `http://capec.mitre.org/` Mitre STIX: `http://stix.mitre.org/` Mitre TAXII `http://taxii.mitre.org/`

Open-source tools

OpenIOC manipulation
`https://github.com/STIXProject/openioc-to-stix`
`https://github.com/tklane/openiocscripts`
Mantis Threat Intelligence Framework
`https://github.com/siemens/django-mantis.git` Mantis supports STIX/CybOX/IODEF/OpenIOC etc via importers:
`https://github.com/siemens/django-mantis-openioc-importer`
Search splunk data for IOC indicators:
`https://github.com/technoskald/splunk-search`
Our framework: `http://github.com/fygrave/iocmap/`

Online Sharing of IOCs

http://iocbucket.com/



Policies on Sharing

Policies on sharing IOCs:

- what to be shared/can be shared
- who to share with
- when to share

Where to look for IOCs:

- Outbound Network Traffic
- User Activities/Failed Logins
- User profile folders
- Administrative Access
- Access from unsual IP addresses
- Database IO: excessive READs
- Size of responses of web pages
- Unusual access to particular files within Web Application (backdoor)
- Unusual port/protocol connections
- DNS and HTTP traffic requests
- Suspicious Scripts, Executables and Data Files

Challenges

Why we need IOCs? because it makes it easier to systematically describe knowledge about breaches.

- Identifying intrusions is hard
- Unfair game:
  - defender should protect all the assets
  - attacker only needs to 'poop' one system.
- Identifying targeted, organized intrusions is even harder
- Minor anomalous events are important when put together
- Seeing global picture is a mast
- Details matter
- Attribution is hard

Challenges

All networks are compromised

The difference between a good security team and a bad security team is that with a bad security team you will never know that you've been compromised.

An Example

A Network compromise case study:

- Attackers broke via a web vuln.
- Attackers gained local admin access
- Attackers created a local user
- Attackers started probing other machines for default user ids
- Attackers launched tunneling tools – connecting back to C2
- Attackers installed RATs to maintain access

Indicators

So what are the compromise indicators here?

- Where did attackers come from? (IP)
- What vulnerability was exploited? (pattern)
- What web backdoor was used? (pattern, hash)
- What tools were uploaded? (hashes)
- What users were created locally? (username)
- What usernames were probed on other machines

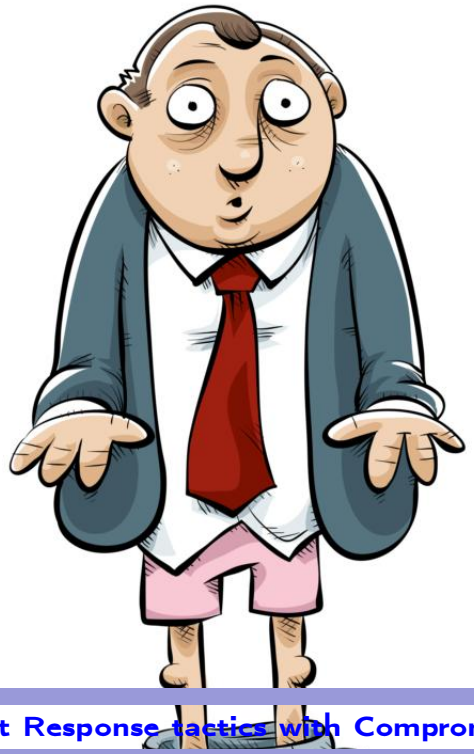
Good or Bad?

File Name	:	RasTls.exe
File Size	:	105 kB
File Modification Date/Time	:	2009:02:09 19:42:05+08:00
File Type	:	Win32 EXE
MIME Type	:	application/octet-stream
Machine Type	:	Intel 386 or later, and compatibles
Time Stamp	:	2009:02:02 13:38:37+08:00
PE Type	:	PE32
Linker Version	:	8.0
Code Size	:	49152
Initialized Data Size	:	57344
Uninitialized Data Size	:	0
Entry Point	:	0x3d76
OS Version	:	4.0
Image Version	:	0.0
Subsystem Version	:	4.0
Subsystem	:	Windows GUI
File Version Number	:	11.0.4010.7
Product Version Number	:	11.0.4010.7
File OS	:	Windows NT 32-bit
Object File Type	:	Executable application
Language Code	:	English (U.S.)
Character Set	:	Windows, Latin1
Company Name	:	Symantec Corporation
File Description	:	Symantec 802.1x Supplicant
File Version	:	11.0.4010.7
Internal Name	:	dot1xtray

It really depends on context

RasTls.DLL
RasTls.DLL.msc
RasTls.exe

http://msdn.microsoft.com/en-us/library/ms682586(v=VS.85).aspx
Dynamic-Link Library Search Order



Tools for Dynamic Detection of IOC

- Snort
- Yara + yara-enabled tools
- Moloch
- Splunk/Log search

Tools for Dynamic Detection

- Moloch
  - Moloch supports Yara (IOCs can be directly applied)
  - Moloch has tagger plugin:

```
# tagger.so
# provides ability to import text files with IP and/or h
# into a sensor that would cause autotagging of all matc
plugins=tagger.so
taggerIpFiles=blacklist,tag,tag,tag...
taggerDomainFiles=domainbasedblacklists, tag, tag, tag
```

Sources of IOCs

ioc collection `http://iocbucket.com`
Public blacklists/trackers could also be used as source:
`https://zeustracker.abuse.ch/blocklist.php?download=ipblocklist`
`https://zeustracker.abuse.ch/blocklist.php?download=domainblocklist`

where to mine IOC

- passive HTTP (keep your data recorded)
- passive DNS

These platforms provide ability to mine traffic or patterns from the past based on IOC similarity

show me all the packets similar to this IOC

We implemented a whois service for IOC look-ups

```
whois -h ioc.host.com attribute:value+attribute:value
```

Mining IOCs from your own data

- find and investigate incident
- Or even read paper
- determine indicators and test it in YOUR Environment
- use new indicators in the future
  see IOC cycle we mentioned earlier

Example

If event chain leads to compromise

`http://liapolasens[.]info/indexm.html`

`http://liapolasens[.]info/counter.php?t=f&v=win%2011,7,700,169&a=true`

`http://liapolasens[.]info/354RIcx`

`http://liapolasens[.]info/054RIcx`

What to do?

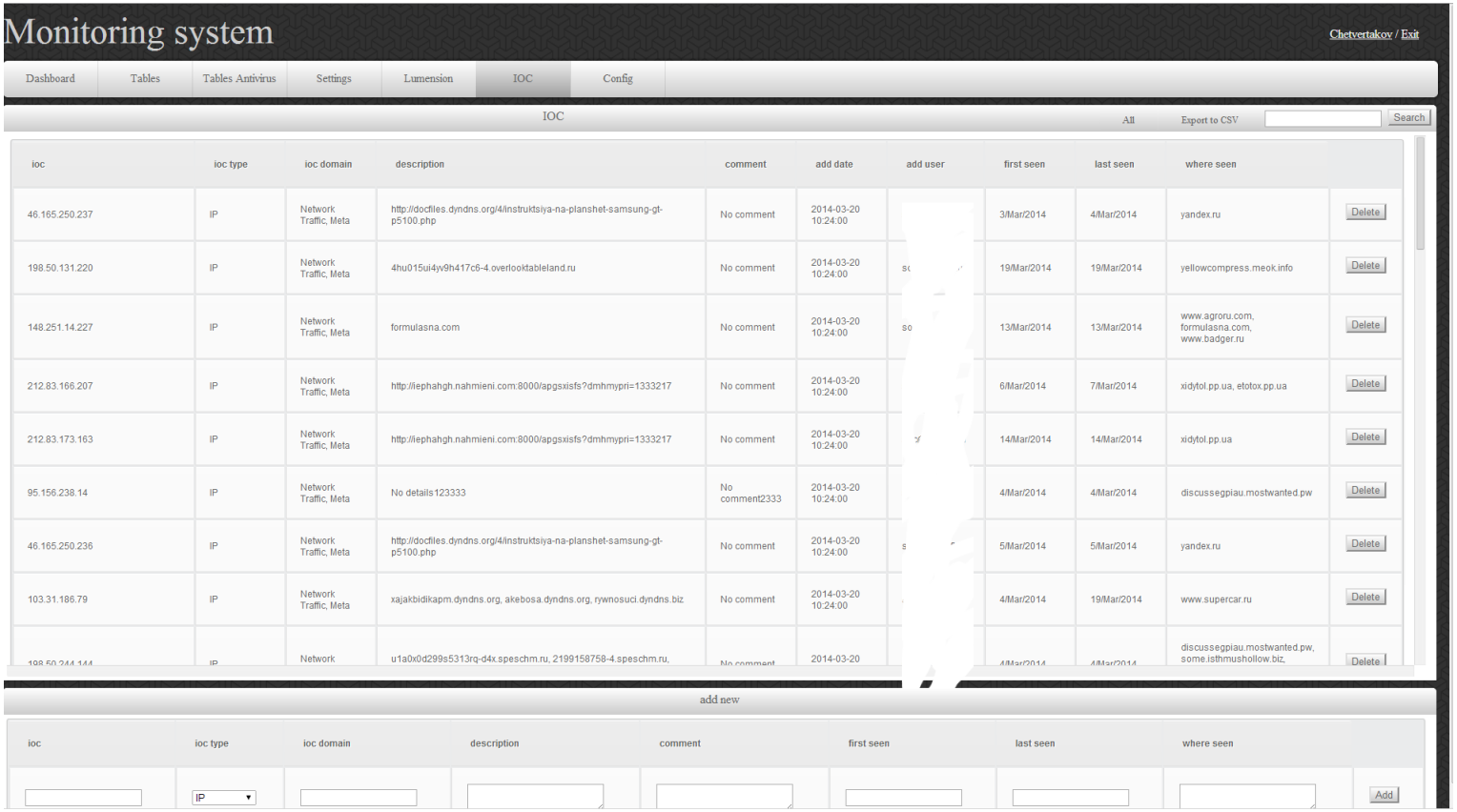
Use YARA, or tune your own tools

```
rule susp_params_in_url_kind_of_fileless_bot_drive_by
{
        meta:
        date = "oct 2013"
        description = "Landing hxxp://jdatastorelame.info/indexm.html  04.10.2013 13:14
        description1 =  " Java Sploit hxxp://jdatastorelame.info/054RIwj      "

    strings:
        $string0 = "http"
        $string1 = "indexm.html"
        $string2 = "054RI"

    condition:
        all of them
}
```

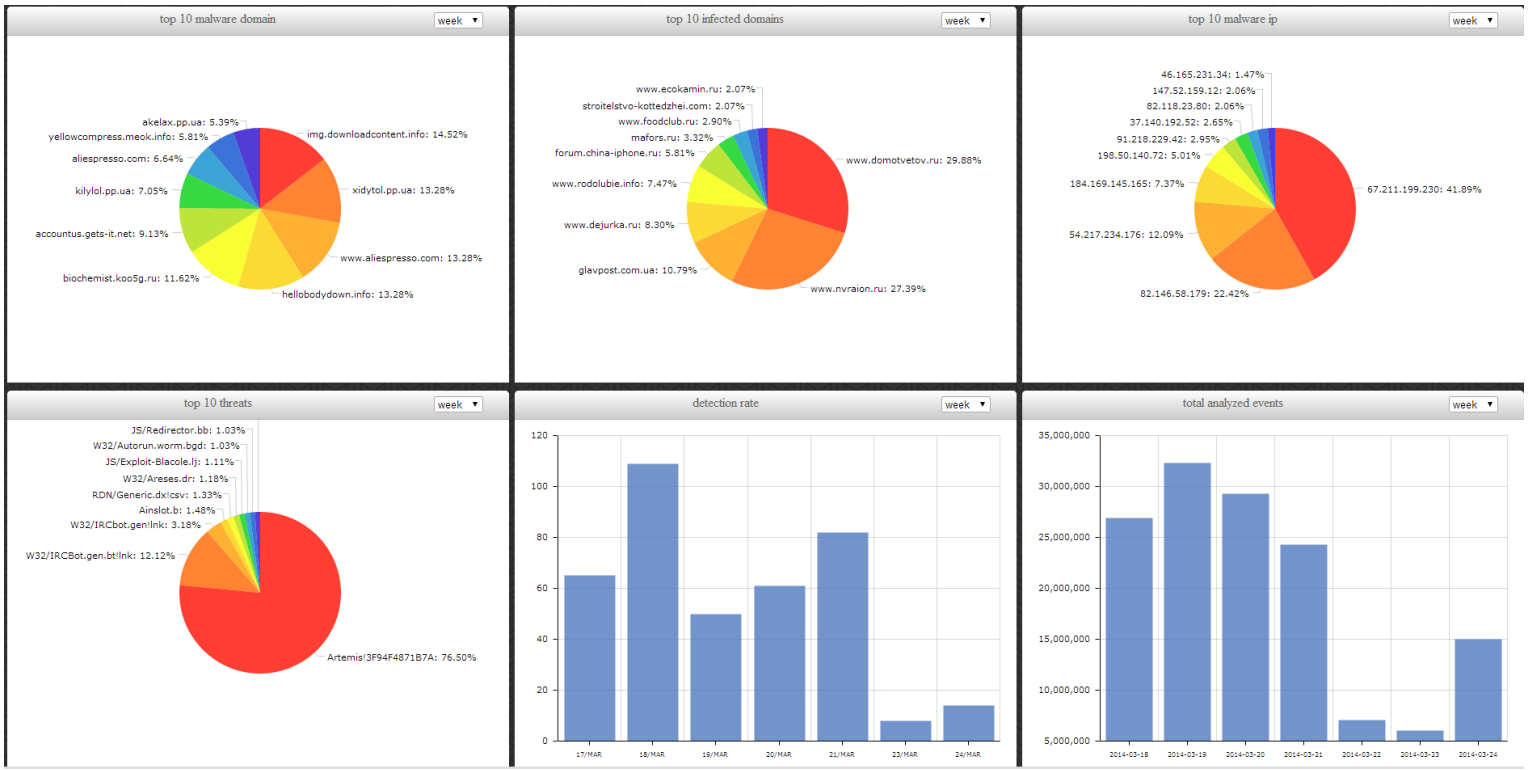
Use snort to catch suspicious traffic:

```
# many plugX deployments connect to google DNS when not in use
alert tcp !$DNS_SERVERS any -> 8.8.8.8 53 (msg:"APT possible PlugX Google DNS TCP
port 53 connection attempt"; classtype:misc-activity; sid:500000112;
rev:1;)
```

IOC management portal



and every manager loves graphs :p



Q and A

Or contact us at . . .