

APT Detection with Host-Based Intrusion Detection System and Intelligent Systems

Seong Oun Hwang

Department of Computer and Information Communication Engineering, Hongik University
Sejong-ro 2639, Jochiwon, Sejong, Republic of Korea
Email: sohwan@hongik.ac.kr

Abstract—Recently, Advanced Persistent Threat (APT) attacks have targeted many institutions, such as governments and companies. APT refers to a type of offensive attacks, which have been performed for a long time using unique attack vectors and malware specifically developed for the target organization. Due to its complicated and sophisticated nature, this threat can be very hard to detect compared to other types of attacks. In this paper, we propose a new method to detect APT attacks by profiling user activities based on Indicator of Compromise (IOC) and chasing malware activities.

Keywords—OSSEC; APT; IOC; HIDS.

I. INTRODUCTION

OSSEC [2] is a free, Open Source host-based intrusion detection system (HIDS) SECurity. It performs functions such as integrity checking (i.e., check if a file is modified, deleted, or added to the agent and send the information to the server), process information checking (i.e., check if a process is terminated, or started), and APT [1] monitoring (i.e., monitor the APT attacks). OSSEC systems largely consist of two parts: agent and manager. The agent is a small program, or collection of programs, installed on a system to be monitored. The agent collects information and forwards it to the manager for analysis and correlation. The manager is the central piece of the OSSEC deployment. It stores the file integrity information, logs, events, and system auditing entries. All the rules, decoders and major configuration options are stored centrally in the manager, which makes it easier to administer in case of a large number of agents.

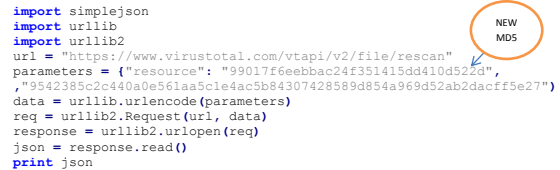
The rest of this paper is organized as follows. In Section II, we propose our OSSEC-based intrusion detection system. In Section III, we describe how to combine the proposed system with other intelligent systems. Finally, we give conclusions in Section IV.

II. PROPOSED SYSTEM

In this section, we implement our host-based intrusion detection system using the OSSEC system model.

A. File Integrity Check

- Get a new MD5 (Message Digest 5) value of a file when it is modified.
- Submit the new MD5 value to Virustotal [3], as shown in Figure 1. In order to scan a file using Virustotal, we need to have both the MD5 hash value and Virustotal key.



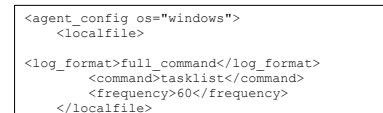
```
import simplejson
import urllib
import urllib2
url = "https://www.virustotal.com/vtapi/v2/file/rescan"
parameters = {"resource": "99017f6eebbac24f351415dd410d522d",
              "9542385c2c440a0e561aa5c1e4ac5b84307428589d854a969d52ab2dacff5e27"}
data = urllib.urlencode(parameters)
req = urllib2.Request(url, data)
response = urllib2.urlopen(req)
json = response.read()
print json
```

Figure 1. MD5 Scan by Virustotal API.

B. Monitoring of Running Processes

Step 1: Accepting remote commands. The first step is to configure the agent log collector option to accept remote commands from the manager. That can be done by editing "internal_options.conf" file (usually located at "C:\Program Files(x86)\ossec-agent\internal_options.conf") and setting the variable logcollector.remote_commands to 1.

Step 2: Specifying the command to list running processes. This is a configuration that can be done both at the agent and at the manager side (using the shared directory). It only depends on how many agents you want to use this command. In our case, we edit "var/ossec/etc/shared/agent.conf" configuration file, and have these settings pushed to our Windows agents. The command we use to list processes in Windows Operating



```
<agent_config os="windows">
  <localfile>

  <log_format>full_command</log_format>
  <command>tasklist</command>
  <frequency>60</frequency>
</localfile>
```

Figure 2. Agent Configuration.

System is "tasklist", as shown in Figure 2. There are other options, such as using wmic, but this one is sufficient. For Unix systems, "ps" can be used.

Step 3: Creating local rules. In this step, we edit our "var/ossec/rules/local_rules.xml" file to add rules that will trigger an alert if our critical process is not running. For the purpose of this example, we will use "wordpad.exe", as shown in Figure 3, but, of course, it could be any other name.

The first rule (id "100050") will trigger a level "7" alert every time tasklist command is executed, unless (as defined in rule "100051") the output matches the string "wordpad.exe". If this is the case, the alert level is set to "0", which means that no alert would be triggered.

Now, we just need to save these changes and restart the

```

<rule id="100050" level="7">
  <if_sid>530</if_sid>
  <match>"ossec: Output: 'tasklist'"</match>
  <description>Critical process not
  found.</description>
  <group>process_monitor,</group>
</rule>
<rule id="100051" level="0">
  <if_sid>100050</if_sid>
  <match>wordpad.exe</match>
  <description>Processes running as
  expected.</description>
  <group>process_monitor,</group>
</rule>

```

Figure 3. Local Rules.

manager for them to be applied. We can do that by running the "ossec-control restart" command.

C. APT Monitoring

APT monitoring can be performed using the APT detection system model, as shown in Figure 4.

Input part: This system is fed with an MD5 hash value of file handling processes which are collected through OSSEC agents along with URL. These MD5 inputs are not determined to be malicious or not yet, and will be under further testing at the Analyzing step.

Audit part: In this part, the MD5 values of files and processes are submitted to Virustotal and the Virustotal's responses are passed to Analyzing part.

Analyzing part: In this part, we check if a suspicious MD5 matches any of existing detection information in Normal Profile database or IOC database [4].

Report part: The report module provides information on the suspicious MD5/file to the security administrator and stores the analysis result in Signature Profile information database. In addition, it stores suspicious URLs to malicious URL (MURL) database.

Output part: It provides signature information in the form of OpenIOC [5] to security organizations or vendors. Figure 5 shows one of the FireEye publicly shared IOCs [4].

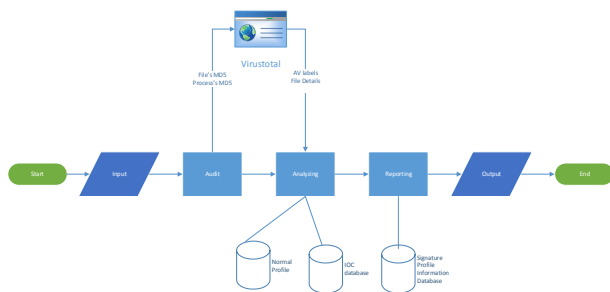


Figure 4. APT Detection System Model.

III. INTEGRATION WITH INTELLIGENT SYSTEMS

The above OSSEC-based intrusion detection system only cannot efficiently cope with malware which are so frequently updated and distributed in a short time that the existing anti-virus vendors including Virustotal cannot make the corresponding malware signatures appropriately. To overcome this limitation, we combine the above system with our intelligent systems such as Payload Chase System (PCS) and Malicious Site Detection System (MSDS) together. PCS periodically

```

xmime="http://schemas.mandiant.com/2010/IOC" id="7b9e87c5-b619-4613-b862-0145614359a"
last-modified="2015-05-13T18:16:33Z"
<short description>BACKDOOR: (FAMILY) </short description>
<description>This IOC contains indicators detailed in the whitepaper "Hiding in
Plain Sight: FireEye and Microsoft Bypass Chinese APT Group's Obfuscation Tactic". The
whitepaper can be read here: https://www.fireeye.com/blog/threat-
research/2015/05/hiding_in_plain_sight.html. This IOC contains indicators for the
BACKDOOR malware family that is attributed to APT7.</description>
<keywords>
  <keyword>FireEye</keyword>
  <keyword>Microsoft</keyword>
  <keyword>APT7</keyword>
</keywords>
<authored by>FireEye</authored by>
<authored date>2014-10-15T21:02:19</authored date>
<links>
  <link rel="category">Backdoor</link>
  <link rel="threatcategory">APT</link>
  <link rel="threatgroup">APT7</link>
  <link rel="license">Apache 2.0</link>
</links>
<definition>
  <indicator id="5d3b0faa-aaf4-4e07-9688-4b886279ab4" operator="OR">
    <indicator item id="84b9a7e-470e-4215-b3ee-e4070b4623" condition="is">
      <content document="FileItem" search="FileItem/MD5sum" type="md5"/>
      <content type="md5">da5eb504e518a26e67959af34612</content>
    </indicator item>
    <indicator item id="fddcebd2-39cd-4e30-907f-cd3f0b9945f3" condition="is">
      <content document="FileItem" search="FileItem/MD5sum" type="md5"/>
      <content type="md5">522fcb48d471249975da208a1cc0a</content>
    </indicator item>
    <indicator item id="69526072-b5ba-4afe-9247-821ddfa93db" condition="is">
      <content document="ProcessItem" search="ProcessItem/HandleList/HandleName"
      type="md5"/>
      <content type="string">358bd08946</content>
    </indicator item>
    <comment>Process Handle Type: events</comment>
    </indicator item>
    <indicator item id="77521051-2713-438b-ab19-c7686f020aba" condition="is">
      <content document="ProcessItem" search="ProcessItem/HandleList/HandleName"
      type="md5"/>
      <content type="string">Pop_0_Management</content>
    </indicator item>
    <comment>Process Handle Type: events</comment>
    </indicator item>
    <indicator item id="dc9b0e69-8b9d-4412-ab38-2ad7171d442" condition="contains">
      <content document="DomainEntryItem" search="DomainEntryItem/RecordName" type="md5"/>
      <content type="string">translate.wordpress.com</content>
    </indicator item>
    <indicator item id="82b1ac0e-af1a-4a6f-9c6f-90c2bf9db67c" condition="contains">
      <content document="DomainEntryItem" search="DomainEntryItem/RecordName" type="md5"/>
      <content type="string">news.juachad.net</content>
    </indicator item>
    <indicator item id="98b14a1d-3840-4d47-87d2-c1a188c49f77" condition="is">
      <content document="FileItem" search="FileItem/MD5sum" type="md5"/>
      <content type="md5">cd14af30b3729a70b04563b3c1ae</content>
    </indicator item>
    <indicator item id="7b02a5b8-f85a-48aa-8447-04b49f8b004" condition="is">
      <content document="PortItem" search="PortItem/remotepIP" type="md5"/>
      <content type="IP">69.80.72.165</content>
    </indicator item>
    <indicator item id="0c4522eb-a19d-411a-9867-cb3bd6c31e" condition="is">
      <content document="PortItem" search="PortItem/remotepIP" type="md5"/>
      <content type="IP">110.45.151.43</content>
    </indicator item>
  </indicator>
</definition>
</IOC>

```

Figure 5. APT IOC.

checks the MURL database and collects new malware downloadable at the distribution page by malware producers in advance, executes them in a controllable environment if they are determined to be malicious by MSDS, monitors their changes in time and records the detailed information such as period of alteration, IP address of the distribution, geographical information, attack patterns to identify and track further the underlying attacker group.

IV. CONCLUSION AND FUTURE WORK

Since the OSSEC system model is based on the signature approach, it cannot efficiently cope with new malware, such as malware variants whose signatures are not published yet. To overcome this problem, we proposed a new host-based intrusion detection method by profiling user activities based on IOC and chasing malware activities periodically. Our future work is to completely develop the OSSEC system model and collect more APT samples enough to classify them using machine learning techniques. If we finally classify APT malware with acceptable accuracy and performance, we could overcome the limitation of the signature approach taken by the conventional anti-virus vendors.

ACKNOWLEDGMENT

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2014R1A1A2054174).

REFERENCES

- [1] R. Jasek, M. Kolarik and T. Vymola, "Apt detection system using honeypots," In Proceedings of the 13th International Conference on Applied Informatics and Communications (AIC'13), WSEAS Press, pp. 25–29, 2013.
- [2] <http://ossec.github.io>, accessed January 2017.
- [3] <https://virustotal.com>, accessed January 2017.
- [4] <https://github.com/fireeye/iocs>, accessed January 2017.
- [5] <http://openioc.org>, accessed January 2017.