TOWARDS SYSTEMATIC HONEYTOKEN FINGERPRINTING

SINCONF 2020

SHREYAS SRINIVASA, JENS MYRUP PEDERSEN, EMMANOUIL VASILOMANOLAKIS
AALBORG UNIVERSITY, DENMARK
Honeypot are deception systems that emulate the services of an end system

Honeytoken is an umbrella term for honeypot-like entities/resources that can be deployed on a network or a system

Honeytokens emulate a resource and hence are light-weight and flexible

Honeytokens are efficient to detect indirect attacks (malware) and direct attacks like unauthorized access

Popular honeytokens include the open source service Canarytokens [1]
Honeytoken operation example
Related work: Honeypot Fingerprinting

- The process of determining that the vulnerable end system is indeed a honeypot

- Honeypot Fingerprinting relies on [2] [3]:
  - observing for static response,
  - partial or
  - invalid response due to limited simulation or library dependency

- This the first attempt towards Honeytoken Fingerprinting
Honeytokens are classified based on operation levels – System, Network, Data and File

For example: a fake user access information in a database that operates at the data level

Fingerprinting techniques are based on these operational levels

<table>
<thead>
<tr>
<th>Honeytoken</th>
<th>Deceptive Entity/Resource</th>
<th>Alerting Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honeyentries [4],[12]</td>
<td>Table data set</td>
<td>DB Monitor</td>
</tr>
<tr>
<td>Honeyword [14]</td>
<td>Password</td>
<td>DB Monitor</td>
</tr>
<tr>
<td>Honeyaccount [8]</td>
<td>User-account</td>
<td>Event Logger</td>
</tr>
<tr>
<td>Honeyfile [17]</td>
<td>File-Google Sheets</td>
<td>Session Log</td>
</tr>
<tr>
<td>Honeyfile [10]</td>
<td>File</td>
<td>Event Logger</td>
</tr>
<tr>
<td>Honeypatch [1], [2]</td>
<td>Vulnerability</td>
<td>Session Log</td>
</tr>
<tr>
<td>HoneyURL [17]</td>
<td>URL</td>
<td>DNS Trigger</td>
</tr>
<tr>
<td>CanaryTrap [7]</td>
<td>Email</td>
<td>Email</td>
</tr>
<tr>
<td>Honeyport [10]</td>
<td>Network port</td>
<td>Session Log</td>
</tr>
<tr>
<td>CanaryToken [26]</td>
<td>File-pdf, docx</td>
<td>DNS Trigger</td>
</tr>
<tr>
<td>CanaryToken [26]</td>
<td>Directory</td>
<td>DNS Trigger</td>
</tr>
<tr>
<td>CanaryToken [26]</td>
<td>URL</td>
<td>DNS Trigger</td>
</tr>
<tr>
<td>Honeybits[15]</td>
<td>Email</td>
<td>DNS Trigger</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alerting Mechanism</th>
<th>Operating Level</th>
<th>Fingerprinting Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Monitor</td>
<td>Data</td>
<td>Modified Date</td>
</tr>
<tr>
<td>Event Logger</td>
<td>System</td>
<td>Last Used, grep search</td>
</tr>
<tr>
<td>Session Log</td>
<td>Application</td>
<td>Grep search</td>
</tr>
<tr>
<td>DNS Trigger</td>
<td>Application, Network</td>
<td>Reverse Engineering Network Sniffing</td>
</tr>
</tbody>
</table>
Proof of concept

- We fingerprint honeytokens generated through the open source Canarytoken service
- Canarytokens provide honeytokens:
  - files(pdf, docx, exe, dll),
  - directories,
  - URLs,
  - image embeds
  - Etc.
- We propose fingerprinting techniques for the pdf, docx and the directories through
  - decomposition
  - reverse engineering techniques
- We exploit the alerting mechanism
- Employ reverse engineering
- Rename the file extension from .docx to .zip (compressed)
- Unzip the zip folder to find xml files
- In the footer.xml file, we find a DNS call made to a Canary Tokens domain
PDF & Windows Directory Honeytoken Fingerprinting

- exploit the hardcoded URL in the embedded triggering mechanism
- Decompose the pdf file by parsing it (python)
- The pdf file contains an embedded hidden object that makes DNS call to a Canarytokens domain

- Directory honeytoken contains a hidden .ini file
- The .ini file is configured to make a DNS call when the directory is accessed
PDF, Directory Honeytoken Fingerprinting

(base) C:\AAU_NetSec\tokengrabber>python folder.py --d C:\Employee_PaySlips
Found: C:\Employee_PaySlips\My Documents\desktop.ini
Canary token detected in file: C:\Employee_PaySlips\My Documents\desktop.ini

(base) C:\AAU_NetSec\tokengrabber>python pdf-parser.py -o 16 -O C:\Important\Salary.pdf
This program has not been tested with this version of Python (3.7.6)
Should you encounter problems, please use Python version 3.6.3

obj 16 0
Containing /ObjStm: 14 0
Type:
Referencing:
<<
  /S /URI
  /URI (http://example.com/callback?token=4f939c83-1e5b-4452-a44f-699e4452ba5f)
>>

(base) C:\AAU_NetSec\tokengrabber>python docx.py -f C:\Important\Contract.docx
C:\AAU_NetSec\tokengrabber\temp\word\footer2.xml
CanaryToken detected
Passive DNS Fingerprinting

- Alerting mechanism works by triggering DNS requests
- Packet Sniffing can parse all requests made to the Canarytokens domain
- Disadvantage: Have to access the honeytoken to confirm the DNS calls
Future Work

Extending the fingerprinting techniques to detect

- System level honeytokens (employing *inode*)
- Database level honeytokens
- User-account based honeytokens
Thank You!

Contact:
Shreyas Srinivasa
Email: shsr@es.aau.dk
References

