The WOMBAT API

Querying a global network of advanced honeypots

Stefano Zanero (Politecnico di Milano)
Paolo Milani Comparetti (Technical University of Vienna)

Black Hat DC, 2-3 February 2010
Knowledge is the key for victory

- Knowing your enemy is the key to success
  - “He will win who knows when to fight and when not to fight... He will win who, prepared himself, waits to take the enemy unprepared. Hence the saying: If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle.” [Sun-Tsu]
  - Perhaps the most often quoted, and less often practiced, sentence in history

- Understanding is the key to (re)acting sensibly
Disappearance of the worms

- In 2001 we were all worried of worms getting wormier
  - “In July 2001, Code Red spread to $HUGE\_INT$ systems within $SMALL\_INT$ hours; the worldwide economic impact was estimated to be $INSANE\_FIGURE$ billions. SQL Slammer was even faster. We'll see an even greater increase in the speed and destructive capabilities of threats.

- The trend was so clear:
  - 2001: Li0n, Code Red, Nimda
  - 2002: Slapper, Klez
  - 2003: SQL Slammer, Blaster, SoBig
  - 2004: Sober, MyDoom, Witty, Sasser
  - I have even an iDefense t-shirt with this list on it!
Opportunities for cybermayhem

- Why didn't the /bin/ladens of the digital world target the infrastructure?
  - FX's and Michael Lynn's works showed the potential to attack routers directly
  - Even with a traditional worm, windows of opportunity:
    - June 2003: MS03-026, RPC-DCOM Vulnerability (Blaster) + Cisco IOS Interface Blocked by IPv4 Packets
    - April 2004: MS04-011, LSASS Vulnerability (Sasser) + TCP Vulnerabilities in Multiple IOS-Based Cisco Products (resets)

- Yet, no worm. Should we just relax?
  - Worms have handed over the scene to botnets
Rise of the robots

- Bots, bots everywhere
  - When I was young (1998), bots were IRC warriors' stuff
  - We used to call remote control trojans “zombies”, and they were usually DDoS tools (2000-2)

- Today's bots are different
  - Intelligent, evolving, with complex C&C infrastructures
  - Larger botnets (10k common, 1M+ seen)
  - Phishing & spamming are more difficult to track than DDoS

- How do we track them? How do we analyze them?
  - Worm explosive propagation vs. bot slow and steady diffusion: there's no network telescope that can see them
Current initiatives

- Efforts by vendors
  - ATLAS (Arbor)
  - DeepSight (Symantec, formerly SecurityFocus)

- Community and no-profit efforts
  - Dshield and the Internet Storm Center (SANS)
  - Network Telescopes
  - The HoneyNet project
  - Leurrecom project
  - MWCollect Alliance
Enter the WOMBAT
Main objectives and principles

Data acquisition

Data enrichment

Threat analysis

New collection practices

Storage Analysis

Meta-data Analysis

Malware analysis

Context analysis

Honeypots

Crawlers

Security tools updates (signatures, …)

Security practices updates (policies, …)
Project aims and innovation

- **Improved data gathering tools**
  - Malware samples
  - Malicious websites
  - Network attacks
  - Mobile viruses

- **Tools and techniques for characterization of threats**
  - Automation of malware analysis (behavioral and static)
  - Malicious websites analysis

- **Framework and tools for qualitative threat evolution assessment and root cause analysis**
Introducing the WAPI idea

- Creation of an infrastructure for data sharing
  - Creation of a set of standard API (WOMBAT API), as opposed to integration on a single database

- WAPI requirements for the data provider
  - Control which content to present to the clients, and how
  - Enrich or modify the dataset without needing to modify all the clients

- WAPI requirements for the client
  - Need for a common “language” to request data from the datasets
  - Need for programming primitives to easily retrieve information on the fly while performing analysis tasks

- We sought and seek international collaboration
  - Already integrated data from other sources
WAPI design challenges

- A set of API that work on heterogeneous, pre-existing data sets
  - So, they are not an ontology
  - They are not a data format specification

- Language, database and platform-independent
  - A set of SOAP enabled methods which easily allow a client to traverse a hierarchy of objects, characterized by attributes, methods and references
  - Reference implementation in Python (SOAPpy), but completely language-independent
  - Reflection allows to hide details of the implementation
WAPI: the main idea

- The WAPI calls allow to explore a dataset:
  - Which objects are offered by the dataset?
  - What are the methods for a given object?
  - What are the references for a given object?
  - What are the attributes and the attribute values for a specific object instance?
  - Does an object instance exist in the dataset?
  - Call a method
  - Follow a reference
  - Get the documentation for an object
A simple WAPI dataset

- Objects: the dataset “concepts”
- Object instances: a specific instance of an object
- Attributes: information on an object instance.
- References: “special methods” that return lists of object instances.
  - The edges of the graph
  - Allow the user to explore the different objects
  - e.g.: Domain name and whois information → DNS relations → IP addresses → geolocation
- Methods: more expensive calculations on an object
  - e.g.: check whether a given IP address is blacklisted
WAPI-enabled data sources

- Virustotal
  - AV results with 40+ antivirus engines
- Anubis
  - Malware analysis in a monitored sandbox
- HARMUR (client-side)
  - Historical Archive of Malicious URLs
- WEPAWET (client-side)
  - Analysis of malicious scripts on a given URL
WAPI-enabled data sources

- Shelia (client-side)
  - Email + URL inspection and Windows malware analysis
- HoneySpider Network (client-side)
  - Crawler, focuses on drive-by download attacks
- BlueBat (mobile)
  - Bluetooth honeypots
- SGNet
  - A scalable network of LIH and HIH
WAPI reference client

- Python wrappers that offer each WAPI object as a normal python object.
- Object instantiation:
  - Check that the object exists
  - Retrieve the list of methods and references
  - Retrieve the attributes and their values
  - Retrieve the documentation
- Method call:
  - Calls the method over WAPI
- Reference call:
  - Returns a list of wrappers for the returned WAPI objects
Connecting to the WAPI datasets

- harmur : success
- virustotal : success
- wepawet : success
- anubis : success
- hsn : success
- shelia : success
- sgnet : success
- forth : success

You are connected to 8 WAPI datasets!
VirusTotal

- Scan results from 40+ AVs
- Web submission of files
- Very large amount of daily submissions
- AV detection statistics on the malware sample, and much more
- Example: is this malware sample recognized by my AV solution? Was it in the past?
```
In [55]: md5
Out[55]: '9ed9de912153d3f7777ce89cfd5aa2ec'

In [56]: virustotal.get_file(md5=md5)
Out[56]: [<virustotal.file object id '9ed9de912153d3f7777ce89cfd5aa2ec'>]

In [57]: file.get_last_analysis()[0].av_positives_report
Out[57]:
{'AVG': ['BackDoor.Turkojan', '8.5.0.423', '2009.10.27'],
 'AhnLab-V3': ['Win-Trojan/Turkojan.307712', '5.0.0.2', '2009.10.26'],
 'AntiVir': ['BDS/Backdoor.Gen', '7.9.1.44', '2009.10.27'],
 'Antiy-AVL': ['Backdoor/Win32.Turkojan.gen', '2.0.3.7', '2009.10.27'],
 'Authentium': ['W32/Agent.Aw.gen!Eldorado', '5.1.2.4', '2009.10.27'],
 'Avast': ['Win32:Turkojan-BZ', '4.8.1351.0', '2009.10.26'],
 'BitDefender': ['Trojan.Generic.230172', '7.2', '2009.10.27'],
 'CAT-QuickHeal': ['Backdoor.Turkojan.r', '10.00', '2009.10.27'],
 'ClamAV': ['Trojan.Agent-14143', '0.94.1', '2009.10.27'],
 'Comodo': ['Backdoor.Win32.Turkojan.il0', '2746', '2009.10.27'],
 'DrWeb': ['Trojan.Rent.166', '5.0.0.12182', '2009.10.27'],
 'F-Prot': ['W32/Agent.AW.gen!Eldorado', '4.5.1.85', '2009.10.26'],
 'F-Secure': ['Backdoor:W32/Turkojan.gen!A', '9.0.15370.0', '2009.10.27'],
 'Fortinet': ['W32/Turkojan.R!tr.bdr', '3.120.0.0', '2009.10.26'],
 'GData': ['Trojan.Generic.230172', '19', '2009.10.27'],
 'Ikarus': ['Downloader.Delphi', 'T3.1.1.72.0', '2009.10.27'],
 'Jiangmin': ['Backdoor/Turkojan.c', '11.0.800', '2009.10.26'],
 'K7AntiVirus': ['Backdoor.Win32.Turkojan.r', '7.10.879', '2009.10.24']
```
Anubis

- Monitored sandbox
- Web submission of files
- Large amount of daily submissions
- Detailed information on the malware behavioral analysis and the corresponding behavioral clustering
- Example: there is a suspicious registry key in my machine. Is any malware known to Anubis performing this action?
In [42]: malware=anubis.malware(md5=md5)[0]

In [43]: task=malware.tasks()[0]

In [44]: task.file_events()[0].dump()

<wobject 'file_event.34841407'>

  File Event object
  <attributes>
    action:modified
    identifier:34841407
    is_directory:0
    name:C:\cmsetac.dll
    result_id:5391203
  </attributes>
  <methods>
  </methods>
  <references>
    tasks(start_date,end_date) : Get the anubis analysis tasks which had this file event.
    This can be filtered by a start and end date.
    @param start_date (default=None) format "YYYY-mm-dd"
    @param end_date (default=None) format "YYYY-mm-dd"

In [45]: [f.name for f in task.file_events() if f.action=='created']
Out[45]: ['C:\cmsetac.dll', 'C:\nttdtcstp.dll']

In [46]:
- Allows to explore the characteristics of the code injection attacks observed by the SGNET deployment
- Example: have you ever seen this malware? How did it propagate?
Specific event associated to a successful code injection attack as detected by the SGNET deployment

(attributes)
- download_address: 85._________
- download_filename: msoft55831.exe
- download_hostname: e18:_________.adsl.alicedsl.de
- download_port: 1945
- download_protocol: ftp
- download_uri: ftp://dleza:dleza@85._________:1945/msoft55831.exe
- dst_addr: 85._________
- dst_port: 135
- identifier: 133197
- path: win2krich:135T:751:|1|6
- src_addr: 85._________
- src_port: 39348
- ts_end: 2008-03-23 14:08:32
- ts_start: 2008-03-23 14:08:25

(methods)
- shellcode(): Returns the shellcode associated to the code injection attack, hex encoded

(references)
- activity(): Points to the activity class corresponding to this code injection event.
HARMUR

- Detailed information on the temporal evolution of malicious websites, on their threats and their location
- Example: where is this suspicious domain hosted? Is it malicious? Was it moved?
In [92]: domain_name="illill1.com.mx"

In [93]: domain=harmur.domain(domain=domain_name)[0]

In [94]: domain.threats()[5].dump()
<wobject 'threat.95'>

<attributes>
  <help>
    help:
    id:Processes Started
    identifier:95
    rating:UNKNOWN
    ts_first_seen:2009-06-21 04:08:44
    ts_last_seen:2009-09-02 19:01:22
    type:BREXP
    type_description:Browser exploit
    url_source:shasta
    url_tags:['Browser exploit']
  </help>
  <methods>
  <references>
    content()
    domain()
    threat_class()
Wepawet

- Detailed exploit information on each analyzed URL
- Example: analyze a site and get the exploit characteristics. How many other sites contain this specific exploit?
An exploit instance.

<attributes>
  added_at: 2008-10-27 11:33:27
  identifier: 13209
  name: QuickTime RTSP
  reference_id: CVE-2007-0015
  task_id: 158017
  vendor: Apple
<methods>
<references>
tasks(): Get all the other tasks that contained this exploit.

A (possible) payload referenced during the analysis of a resource.

<attributes>
anubis_url: http://anubis.iseclab.org/?action=result&task_id=108cf49df1631f8a49c10fcaa351578c
In [10]: wepawet_task.payloads()[0].dump()
<wobject 'payload.33768'>
  A (possible) payload referenced during the analysis of a resource.
  <attributes>
    anubis_url:http://anubis.iseclab.org/?action=result&task_id=108cf49df1631f8a49c10fcaa351578c
    hash:a88b202fc2f319493c82a0a2b9b4d316
    identifier:33768
    status:0
    task_id:158017
  type:MS-DOS executable PE for MS Windows (GUI) Intel 80386 32-bit
  url:http://hildjxvdves.com/cgi-bin/index.cgi?ECVCEzzEZzZZsZrZZMCOArZEUcZEZOVMOfbZMMkTczVMCZZZZzZkZlZZZZZZZZZFFZ
  virustotal_url:http://www.virustotal.com/analysis/159dbe3675d0ba1dc53f67ebd44863dac7450d57dc09dc68bf0cadadd4457bf-1247833388
  <methods>
  <references>
    tasks(): Get all tasks that referenced this payload.

In [11]:
A more comprehensive demo
WAPI-enabling your data

- We want your data = we made it easy to WAPI-enable it
- Define the dataset as a collection of python classes following some simple naming conventions
  - Extend wapi.object.WObject
  - Define the type name as a static WTYPE
  - Custom constructor WObject.init()
  - Attributes have prefix W_
  - Methods have prefix WM_
  - References have prefix WR_ and return list of WObjects
Example

class Address(WObject):
    
    """
    This is the object documentation
    """
    def init(self):
        #query your database to retrieve information
        address, location = query_db(self.W_identifier)

        #define the attributes
        self.W_address = address
        self.W_location = location

    def WM_issspammer(self):
        """
        This is the method documentation
        """
        #query the database to check whether it's a blacklisted IP
        return isblacklisted(self.W_identifier)
Conclusions

- We introduced the WOMBAT project aims
  - Integrating data on threats and attacks
  - Gathering data on novel threats
  - Incorporating contextual features

- We designed the WAPI to enable easier data sharing
  - More control for the data provider
  - Transparency to updates
  - Integration in a single client

- We want your data!
  - Please contact us for sharing agreements
Questions?

Thanks for your attention!

www.wombat-project.eu

Stefano Zanero stefano.zanero@polimi.it

Paolo Milani Comparetti pmilani@iseclab.org

WOMBAT is partially supported by the European Commission through project IST-216026 funded by the 7th Framework Program. The opinions here expressed are those of the authors, and do not necessarily reflect the views of the European Commission.