WOMBAT

(or: what are the baddies doing in my network?)

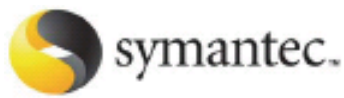
Herbert Bos – Vrije Universiteit Amsterdam
www.cs.vu.nl/~herbertb/

Who/what is Wombat?

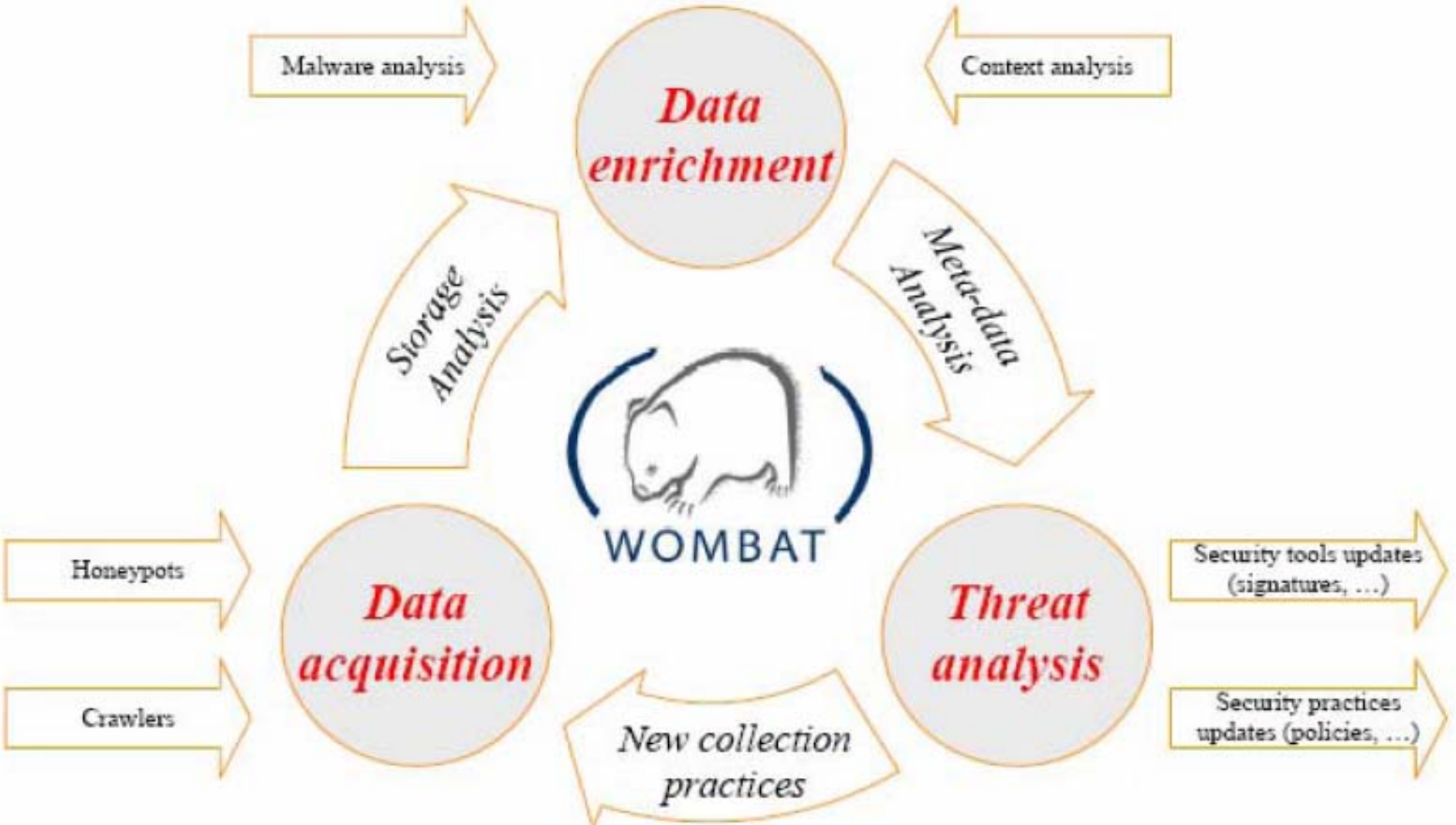
WOMBAT

Worldwide Observatory of Malicious Behaviours and Attack Threats

European Commission
Information Society and Media



Objectives


Malware analysis
Context analysis
Data enrichment
Storage Analysis
Meta-data Analysis
WOMBAT
Honeypots
Crawlers
Data acquisition
Threat analysis
Security tools updates (signatures, …)
Security practices updates (policies, …)
New collection practices

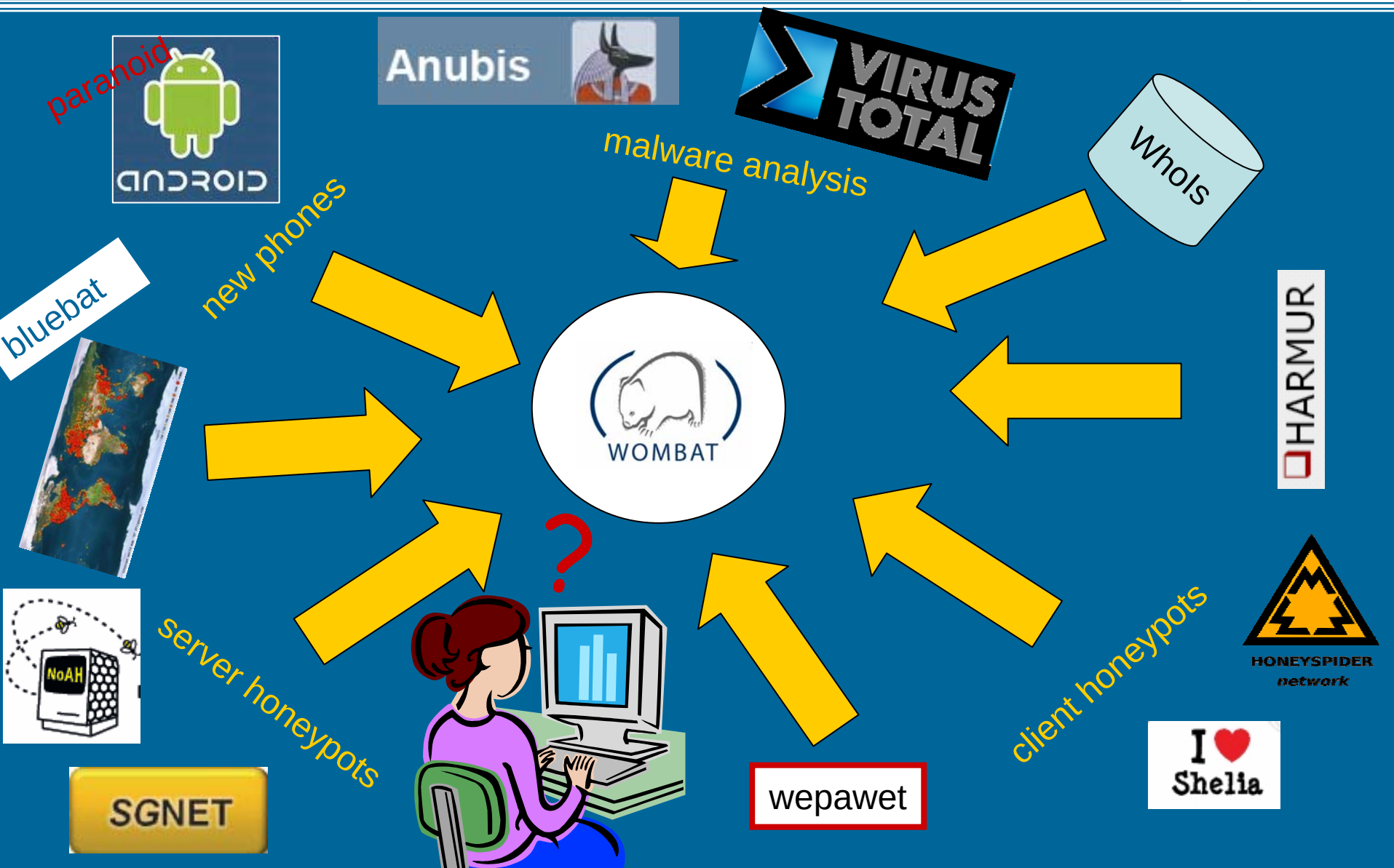
Outcomes

- New data gathering tools
  - –new features (high interaction, real-time analysis)
  - –new targets (wireless, bluetooth, RFID, …)
- Tools + techniques for characterization of malware
  - –malware-based analysis AND Contextual analysis
- Framework and tools for qualitative threat analysis
  - –early warning systems
  - –who is behind these attacks?

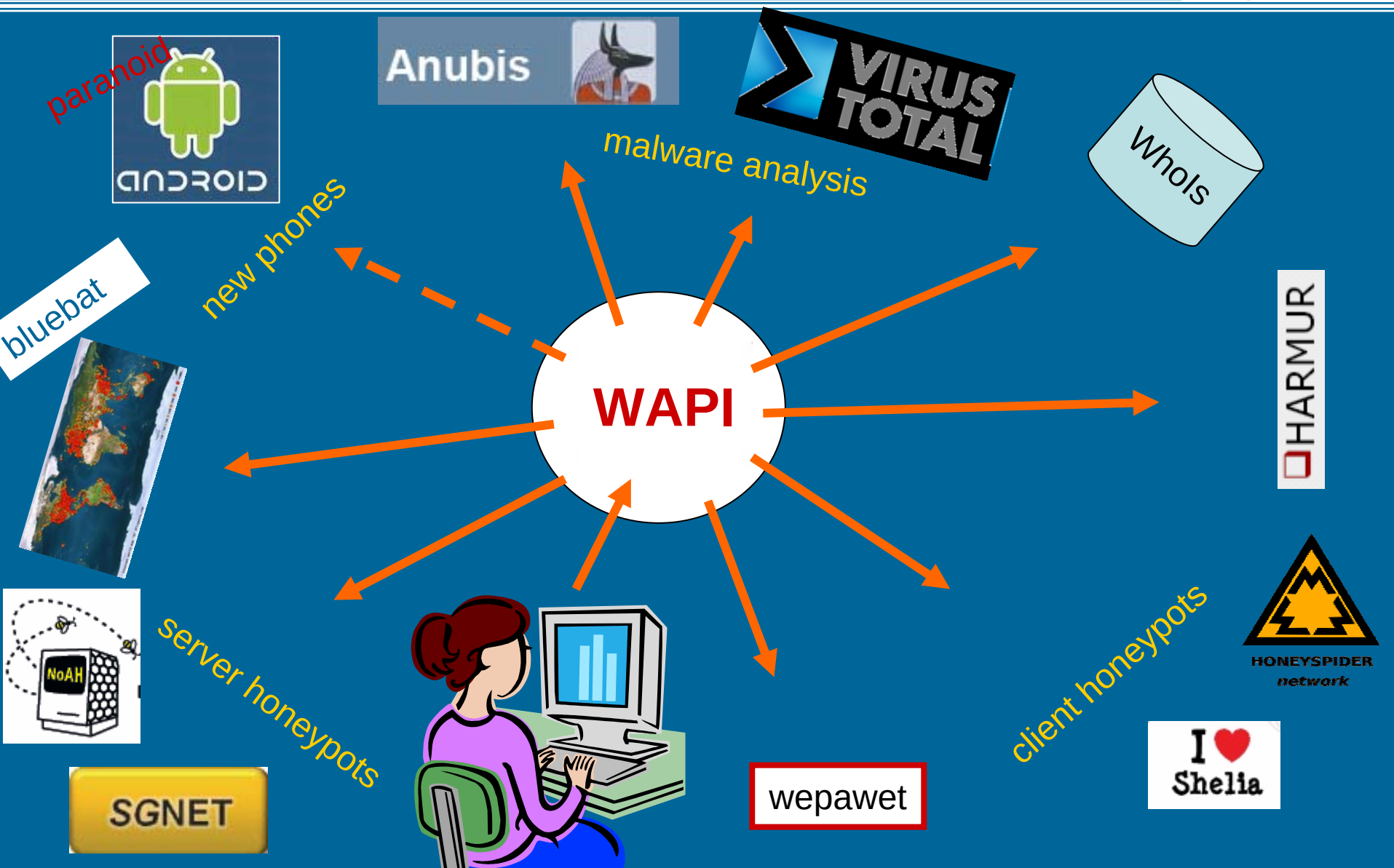
we need data!

access different sources!

Data

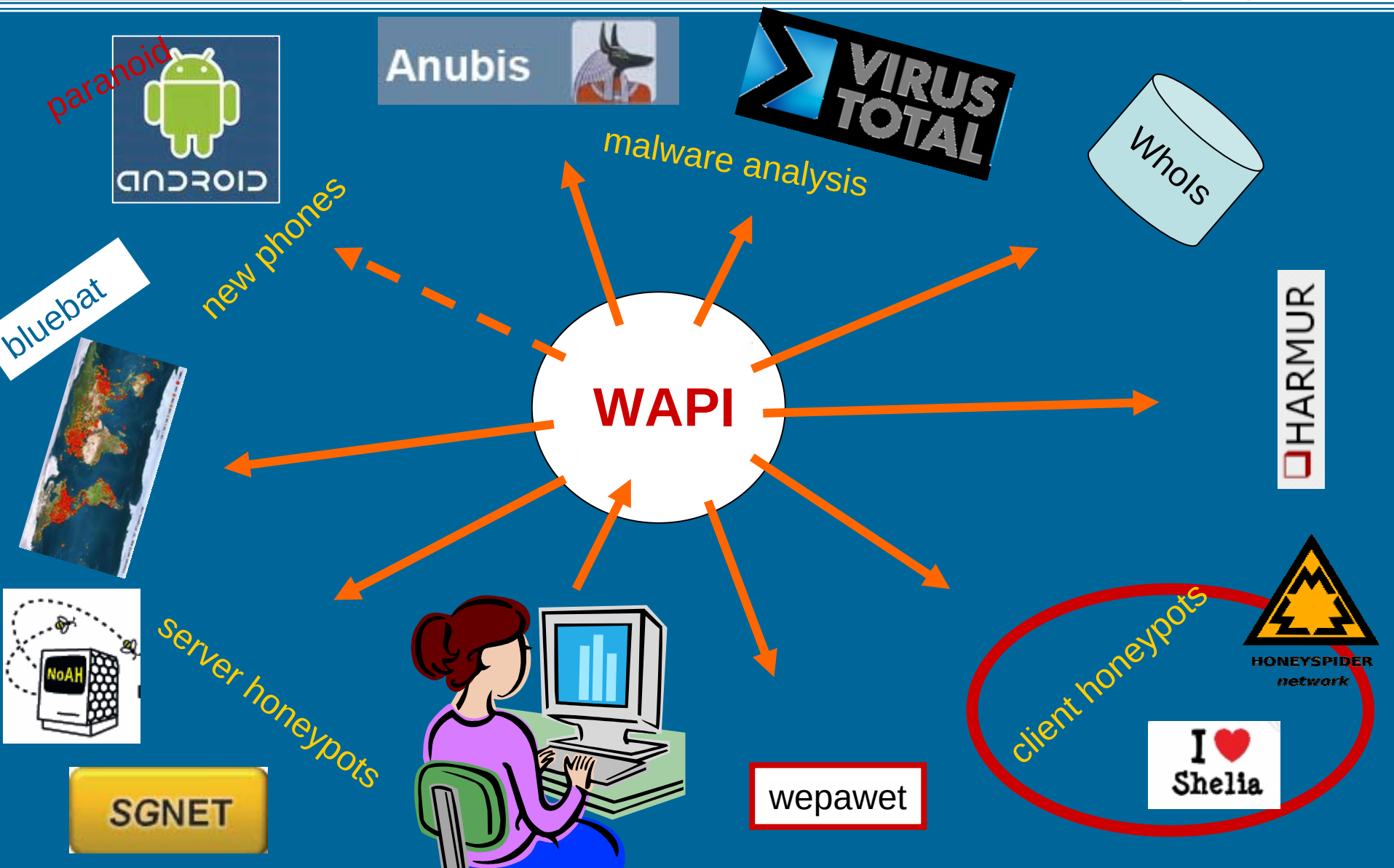

paranoid
android
Anubis
VIRUS TOTAL
malware analysis
WhoIs
new phones
bluebat
WOMBAT
HARMUR
NoAH
server honeypots
?
client honeypots
HONEYSPIDER network
SGNET
wepawet
I ♥ Shelia

Data access: WAPI



More concrete, please

More concrete, please



Client honeypot

From Wikipedia, the free encyclopedia

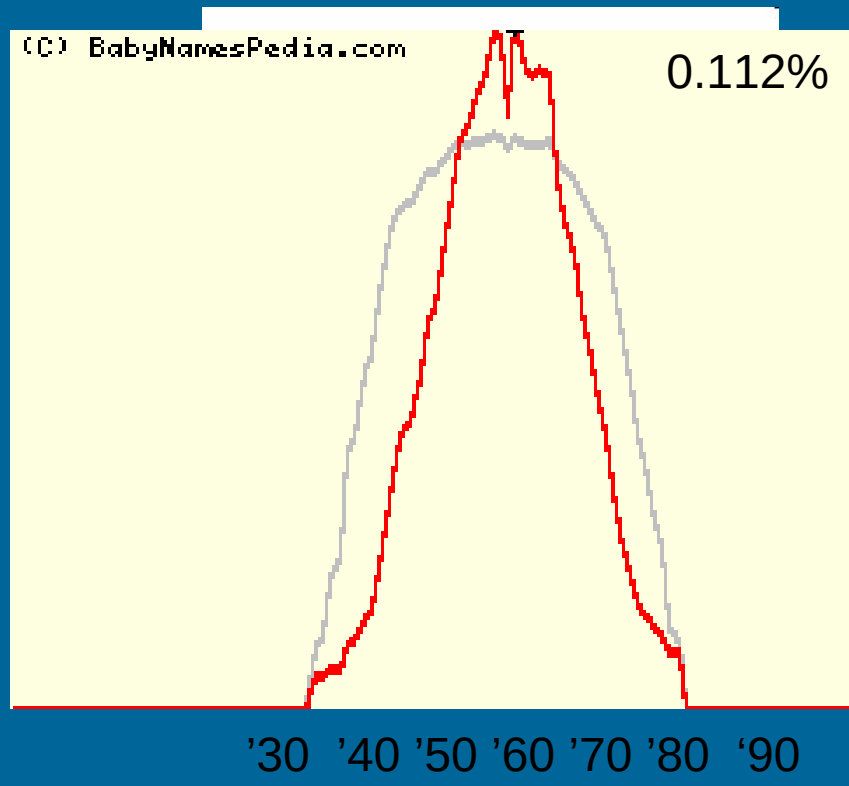
Honeypots are security devices whose value lie in being probed and compromised. Traditional honeypots are servers (or devices that expose server services) that wait passively to be attacked. **Client Honeypots** are active security devices in search of malicious servers that attack clients. The client honeypot poses as a client and interacts with the server to examine whether an attack has occurred. Up until now, the focus of client honeypots have been web browsers, but any client that interacts with servers can be part of a client honeypot (for example ftp, ssh, email, etc).

There are several terms that are used to describe client honeypots. Besides client honeypot, which is the generic classification, honeyclient is the other term that is generally used and accepted. However, there is a subtlety here, as "honeyclient" is actually a homograph that could also refer to the first open source client honeypot implementation (see below), although this should be clear from the context.

Shelia

- Joan Robert Rocaspana
- Georgios Portokalidis
- Philip Homburg
- Herbert Bos

Latin: "Blind"



"Great, but what's with the name?"

Shelia

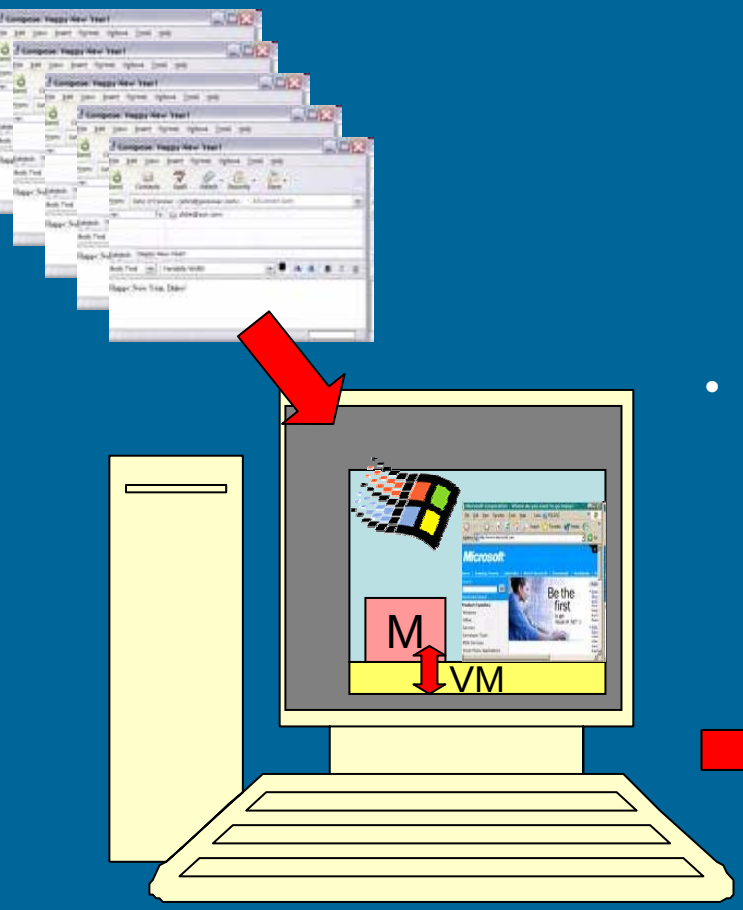
- High-interaction client honeypot
- for Windows
- Goals:
  - no false positives
  - ease of management

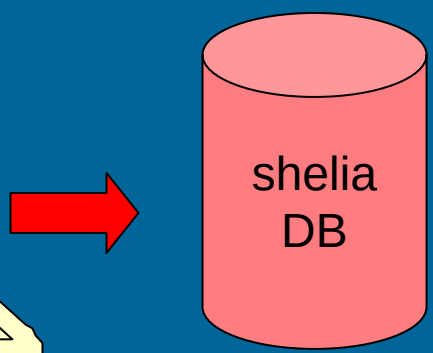
3 main phases

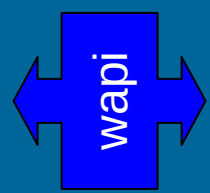
- client emulation
  - “blindly follow all links, open all attachments”
- attack detection
  - “did we see any sensitive actions from memory areas where there should not be code?”
- log
  - “If (attack) store as much info as possible”

Big picture

- shelia mgmt server on host
  - starts VM with shelia mgmt client
    - client listens on socket for target objects (URLs and attachments)
    - launches Shelia detector with appropriate app
    - returns results to server
  - retrieves urls and attachments from DB to pass to client
    - order by timestamp and priority
    - and by type (default: attachments first, but can be modified)
  - periodically restarts VM (also when connection is lost)
    - to ensure we stay clean
  - writes results in DB
- DB can be filled in many different ways
  - IMAP client
  - manual / file parser

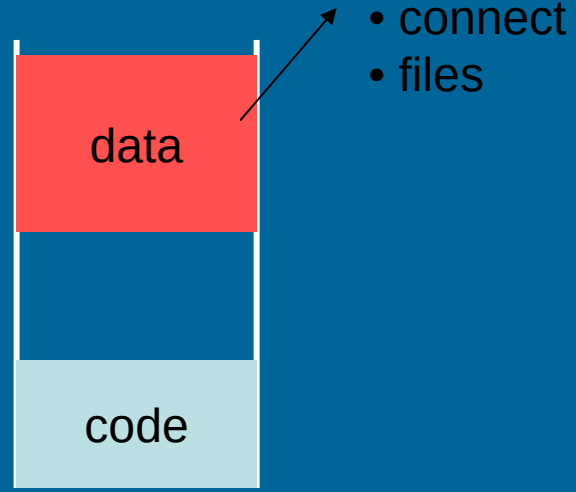






avoid false positives

- we do not treat the system as a black box
  - e.g., we do not look at FS changes
- instead, we explicitly detect badness
  - change the registry
  - open network connections
  - write or execute files



  - if called from area not supposed to contain code!

Log

- execute payload (shellcode)
- try to download the malware
- log in DB

Alerts
- timestamp
- target application
- target URL/object
- payload (shellcode)

Attachment
- file name
- md5
- data

Malware
- binary
- MD5, sha-1, sha256
- filename of malware on FS

Calls
- function name
- function description
- arguments

URLs
- timestamp entered
- processed

Example

```
coogee: ../src>PYTHONPATH=../../SOAPpy/ python wapi_client.py -c democonf --no-ipython


__          __     _____ _____
\ \        / /\   |  __ \_   _|
 \ \  /\  / /  \  | |__) || |
  \ \/  \/ / /\ \ |  ___/ | |
   \  /\  / ____ \| |    _| |_
    \/  \/_/    \_\_|   |_____|
      The WOMBAT API (version 1.0)

Connecting to the WAPI datasets
-> harmur :  success
-> virustotal :  success
-> wepawet :  success
-> anubis :  success
-> hsn :  success
-> shelia :  success
-> sgnet :  success
-> forth :  success
-> utils :  success
You are connected to 9 WAPI datasets!
Welcome to the wombat wapi client
>>>
```

example 2

let us see how it works


```
>>> whelp(shelia)
<wobject 'dataset.shelia'>
 Shelia dataset.
 <attributes>
  identifier:shelia
 <methods>
 <references>
  alert(alert_id) : Returns the WAPI alert object: with a given alert_id (shelia internal key)
  alerts() : Returns all WAPI alert objects in the database:
  alerts_by_target(target) : Returns the WAPI alert objects: that match a URL/filename (SQL) patte
  malware_by_filename(fn) : Returns the WAPI malware objects: that match a filename (SQL) pattern
  malware_by_md5(md5) : Returns the WAPI malware objects: with a given md5
  malware_by_sha1(sha1) : Returns the WAPI malware objects: with a given sha1
  malware_by_sha256(sha256) : Returns the WAPI malware objects: with a given sha256
  urls(url) : Returns the WAPI url objects: that match a URL (SQL) pattern
>>>
```

find alerts caused by target containing ‘http’


```
>>> shelia.alerts_by_target(target="http")
[<shelia.alert object id '2'>, <shelia.alert object id '5'>, <shelia.alert object id '6'>,
<shelia.alert object id '7'>, <shelia.alert object id '8'>, <shelia.alert object id '9'>,
<shelia.alert object id '10'>, <shelia.alert object id '11'>, <shelia.alert object id '15'>,
<shelia.alert object id '17'>, <shelia.alert object id '18'>, <shelia.alert object id '19'>,
<shelia.alert object id '20'>, <shelia.alert object id '21'>, <shelia.alert object id '22'>,
<shelia.alert object id '23'>, <shelia.alert object id '24'>, <shelia.alert object id '25'>,
<shelia.alert object id '26'>, <shelia.alert object id '27'>, <shelia.alert object id '28'>,
<shelia.alert object id '29'>, <shelia.alert object id '30'>, <shelia.alert object id '31'>,
<shelia.alert object id '32'>, <shelia.alert object id '33'>, <shelia.alert object id '34'>,
<shelia.alert object id '35'>, <shelia.alert object id '36'>, <shelia.alert object id '37'>,
<shelia.alert object id '38'>, <shelia.alert object id '39'>, <shelia.alert object id '40'>,
<shelia.alert object id '41'>, <shelia.alert object id '42'>, <shelia.alert object id '43'>,
<shelia.alert object id '44'>, <shelia.alert object id '45'>, <shelia.alert object id '46'>,
<shelia.alert object id '47'>, <shelia.alert object id '48'>, <shelia.alert object id '49'>,
<shelia.alert object id '50'>, <shelia.alert object id '51'>, <shelia.alert object id '52'>,
<shelia.alert object id '53'>, <shelia.alert object id '54'>, <shelia.alert object id '55'>,
<shelia.alert object id '57'>, <shelia.alert object id '62'>, <shelia.alert object id '63'>,
<shelia.alert object id '64'>, <shelia.alert object id '65'>, <shelia.alert object id '66'>]
>>>
```

let us pick one


```
>>> shelia.alerts_by_target(target="http")[8].dump()
<wobject 'alert.15'>
 An alert raised by Shelia
 <attributes>
  addr:202571238
  appl:C:\Program Files\Internet Explorer\iexplore.exe
  identifier:15
  is64bit:0
  payload_pid:1184
  shelia_version:1.2.1
  target:http://azadars.com
  target_pid:1184
  timestamp:2009-08-24 10:51:53
 <methods>
  data()
  payload()
 <references>
  calls() : Returns the WAPI call objects associated with the alert
  malware() : Returns the WAPI malware objects associated with the alert
>>>
```

what about the malware?


```
>>> shelia.alerts_by_target(target="http")[8].malware()
[<shelia.malware object id '1'>]
>>> shelia.alerts_by_target(target="http")[8].malware()[0].dump()
<wobject 'malware.1'>
 Malware object.
 <attributes>
  fn:C:\DOCUME~1\user\LOCALS~1\Temp\update.exe
  identifier:1
  length:28160
  md5:00b23b08657a153fcde4e0891e2484bb
  sha1:522674387e1a8e2d3ab5f7c11ecd9db7e5904dc4
  sha256:b851756487f055bb746cae506e5ffc016f88a07177ab7bfc5b8be7208cbc8156
 <methods>
  binary() : Returns the actual malware
 <references>
  alerts() : Returns the WAPI alerts objects associated with the malware
```

Example (2)

- Looking at HTTP proxy access logs, we notice that some of our clients perform every 20 minutes HTTP requests to the following URL:

  `http://ijmkkyjves.net/iE=eQBHE8cNe8DRM.`

- This behavior is suspicious
  ➔ is the machine infected by a malware?
  if so, what is it?

Let's look at what Anubis says!


```
http=anubis.http_traffic(destination="ijmkkyjves.net")
malware = [h.tasks()[0].malware()[0] for h in http]
cc_stats = set([(m.md5,m.file_size,m.mime_type) for m in cc_malware])
cc_md5=set([m.md5 for m in cc_malware])
print cc_stats
```

```
[('25daf7f2d35c942b4454ba5cc30f98d6', 27648, 'application/x-dosexec'),
 ('30475a021b535c335d107eb572209090', 28160, 'application/x-dosexec'),
 ('8891e825c5d1ae7e128439f14e1b0aa6', 27648, 'application/x-dosexec'),
 ('95da18a176d6f58c1d77ca87cd82f221', 28160, 'application/x-dosexec'),
 ('9ee7dfbaae3671449ad2f3d6cbc38619', 27648, 'application/x-dosexec'),
 ('d0a29f3e05a3de4619bdbb105fa23c63', 27648, 'application/x-dosexec'),
 ('dd848c42013209e542d24fc71998de15', 28160, 'application/x-dosexec')]
```

What do we know about these samples? Let's ask VT

```
for md5 in cc_md5:
        print "=== %s"%md5
        print virustotal.get_file(md5=md5)[0] \
                .get_last_analysis()[0].av_positives_report
```

```
=== 25daf7f2d35c942b4454ba5cc30f98d6
{'Prevx': ['Medium Risk Malware', '3.0', '2009.07.20'], 'NOD32': ['a variant of
    Win32/Kryptik.UL', '4261', '2009.07.20'], 'GData': ['Trojan.Generic.2187991', '19',
    '2009.07.20'], 'Symantec': ['Trojan.Mebroot', '1.4.4.12', '2009.07.20'], 'McAfee-
    GW-Edition': ['Heuristic.BehavesLike.Win32.Suspicious.H', '6.8.5', '2009.07.20'],
    'Sunbelt': ['Trojan.Mebroot', '3.2.1858.2', '2009.07.19'], 'BitDefender':
    ['Trojan.Generic.2187991', '7.2', '2009.07.20'], 'K7AntiVirus':
    ['Trojan.Win32.Malware.1', '7.10.796', '2009.07.18'], 'Panda': ['Trj/CI.A',
    '10.0.0.14', '2009.07.19']}
=== 30475a021b535c335d107eb572209090
{'F-Secure': ['Backdoor.Win32.Sinowal.fci', '8.0.14470.0', '2009.07.31'], 'Prevx':
    ['Medium Risk Malware', '3.0', '2009.08.01'], 'GData': ['Win32:Fraudo ', '19',
    '2009.08.01'], 'Symantec': ['Trojan.Mebroot', '1.4.4.12', '2009.08.01'],
    'McAfee+Artemis': ['Artemis!30475A021B53', '5694', '2009.07.31'], 'McAfee-GW-
    Edition': ['Heuristic.BehavesLike.Win32.Suspicious.H', '6.8.5', '2009.08.01'], 'a-
    squared': ['Backdoor.Win32.Sinowal!IK', '4.5.0.24', '2009.08.01'], 'Avast':
    ['Win32:Fraudo', '4.8.1335.0', '2009.07.31'], 'nProtect':
    ['Trojan/W32.Agent.28160.FT', '2009.1.8.0', '2009.08.01'], 'Kaspersky':
    ['Backdoor.Win32.Sinowal.fci', '7.0.0.125', '2009.08.01'], 'Microsoft':
    ['PWS:Win32/Sinowal.gen!P', '1.4903', '2009.08.01'], 'Ikarus':
    ['Backdoor.Win32.Sinowal', 'T3.1.1.64.0', '2009.08.01'], 'Antiy-AVL':
    ['Backdoor/Win32.Sinowal.gen', '2.0.3.7', '2009.07.31'], 'AntiVir':
    ['TR/PSW.Sinowal.28160P.1', '7.9.0.238', '2009.07.31'], 'K7AntiVirus':
    ['Backdoor.Win32.Sinowal.fci', '7.10.808', '2009.08.01'], 'AVG': ['PSW.Sinowal.Z',
    '8.5.0.406', '2009.08.01'], 'Panda': ['Trj/CI.A', '10.0.0.14', '2009.08.01']}
=== 8891e825c5d1ae7e128439f14e1b0aa6
{'Symantec': ['Trojan.Mebroot', '1.4.4.12', '2009.07.19'], 'Panda': ['Suspicious file',
    '10.0.0.14', '2009.07.19'], 'McAfee-GW-Edition':
    ['Heuristic.BehavesLike.Win32.Suspicious.H', '6.8.5', '2009.07.19']}
```

etc

Back to the proxylogs

```
domains=["google.com",
   "facebook.com",
   "baidu.cn",
   "adobe.com",
   "bandwidthplace.com",
   "azadars.com"]
```

Let's see what Shelia knows about them

```
>>> for d in domains:
...   shtarget = shelia.alerts_by_target (target=d)
...   if len (shtarget) > 0:
...     print "Result for " + d
...     shtarget[0].dump()
...
Result for azadars.com
<wobject 'alert.15'>
 An alert raised by Shelia
 <attributes>
  addr:202571238
  appl:C:\Program Files\Internet Explorer\iexplore.exe
  identifier:15
  is64bit:0
  payload_pid:1184
  shelia_version:1.2.1
  target:http://azadars.com
  target_pid:1184
  timestamp:2009-08-24 10:51:53
 <methods>
  data()
  payload()
 <references>
  calls() : Returns the WAPI call objects associated with the alert
  malware() : Returns the WAPI malware objects associated with the alert
```

…and we can just keep digging

- What is the shellcode used in the attack?
- What is the malware downloaded by the attack?
- What Windows registry did the code modify?
- What files did it create?
- Do any of the other dataset have malware that is similar?

Conclusion I

- Shelia works. It is different. It is available.

  Download from:

  www.cs.vu.nl/~herbertb/misc/shelia

- WAPI allows you to combine Shelia data with other sources

Sharing info is crucial

- WAPI-like solutions help
  - deal with trust
  - deal with availability
  - deal with heterogeneity

- WOMBAT are pretty cool. You can find some at:

  http://wombat-project.eu/