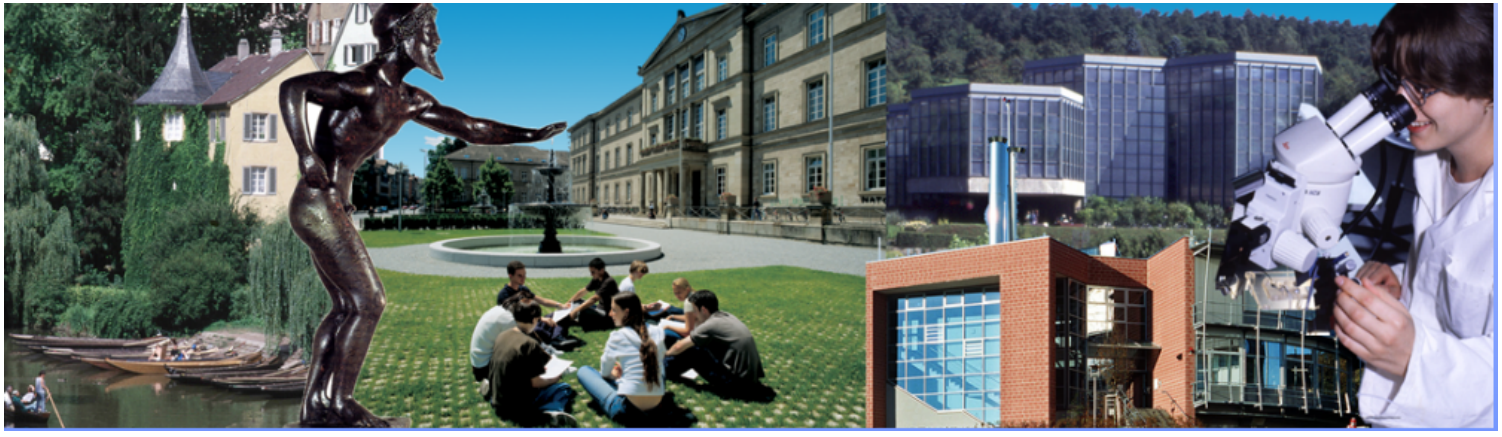
Eberhard Karls
Universität
Tübingen



Intrusion Detection and Malware Analysis

Malware analysis

Pavel Laskov
Wilhelm Schickard Institute for Computer Science

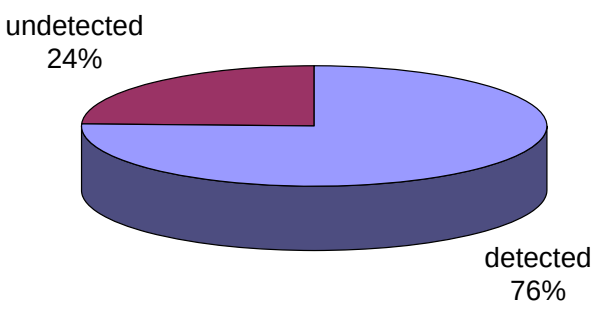
What do we want to know about malware?

- Is it recognized by existing antivirus products?
- What is its functionality?
  - How is malware distributed?
  - What other harmful functions does malware carry out?
- What are relationships between various classes of malware? Do they share common techniques? How do they evolve?

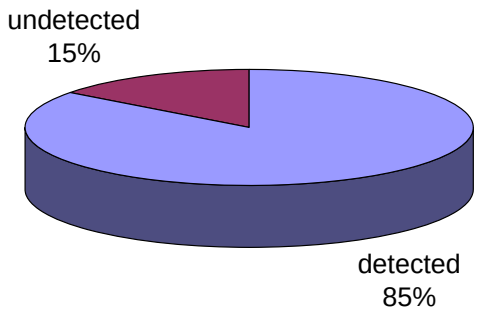
How much of malware is unknown?

- Experiment:
  - Current instances of malware were collected from a Nepenthes honeypot.
  - Files were scanned with Avira AntiVir.
- Results:

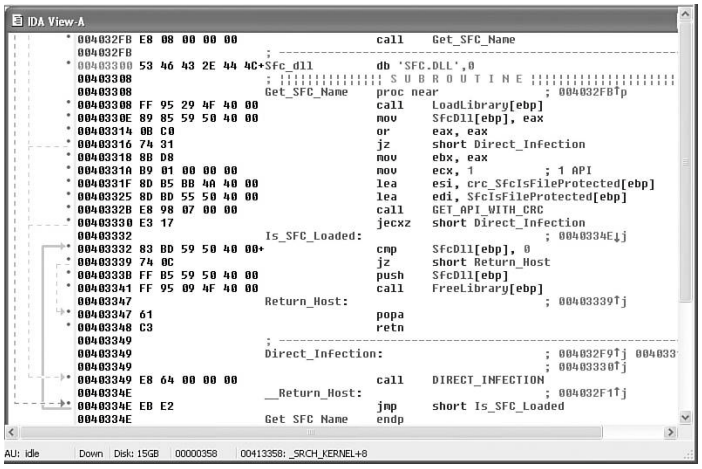
First scan:

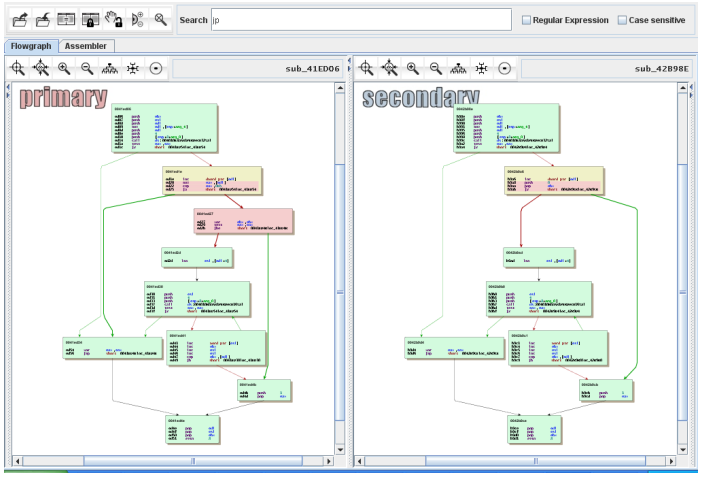


Second scan:



After four weeks 15% of malware instances were still not recognized!





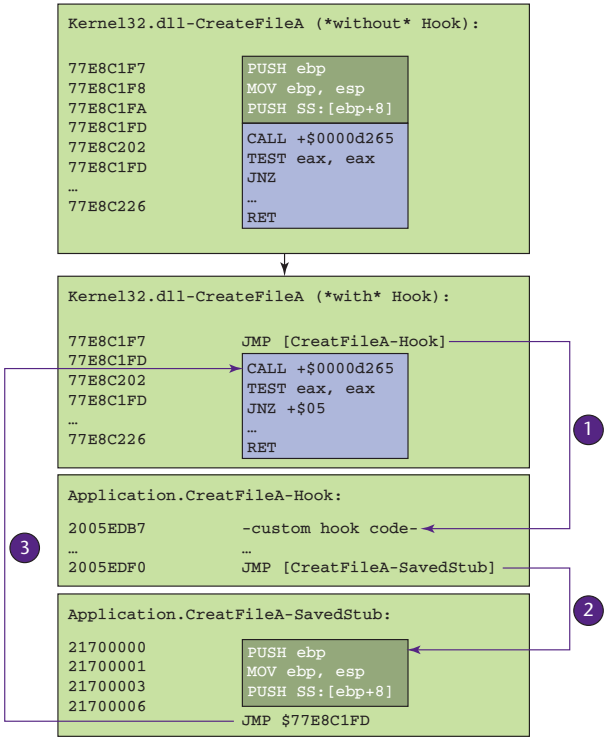
- Binary disassemblers (IDA Pro, OllyDbg)
  - $(+)$ precise understanding of malware behavior
  - $(-)$ extensive manual effort
- Static flow analysers (BinDiff, BinNavi)
  - $(+)$ visual representation of malware program flow
  - $(-)$ prone to obfuscation

Modern malware analysis tools

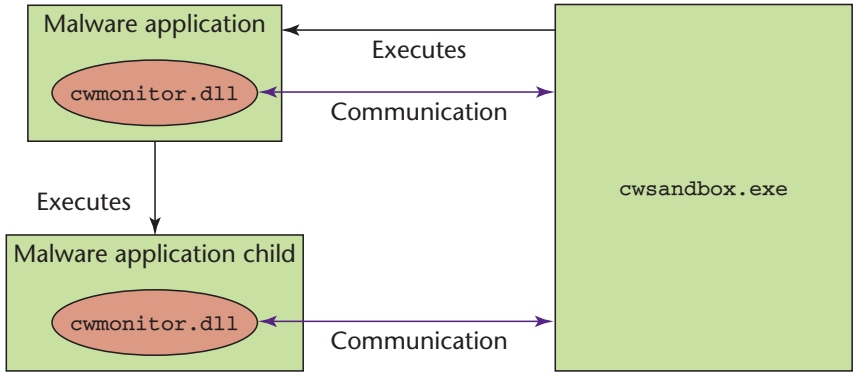
- Sandboxes
- Behavior analysers
- Clustering and classification methods
- Signature generation tools

CWSandbox: key ideas

- API hooking: intercept Windows API calls by overwriting API functions' code loaded in the process memory.
- DLL injection: insertion of hook functions into the running processes.
- Recording of persistent system changes collected by a hook in an XML report.



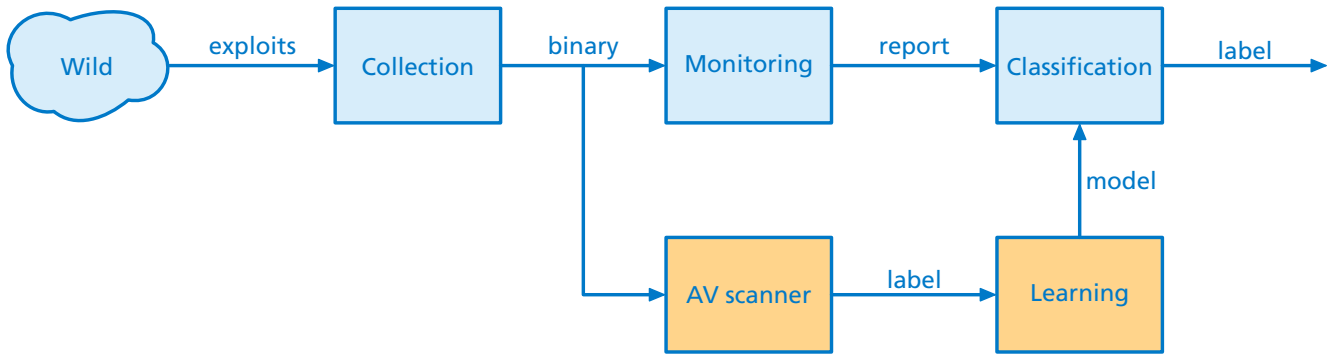
CWSandbox: system architecture



- During the initialization of a malware binary `cwmonitor.dll` is injected into its memory to carry out API hooking.
- DLL intercepts all API calls and reports them to CWSandbox.
- The same procedure is repeated for any child or infected process.

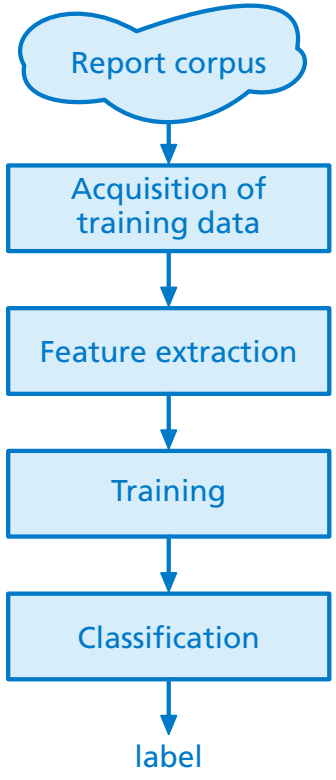
Case study: Malware classification

- Goals:
  - Detect variations of known malware families.
  - Detect previously unknown families.
  - Determine essential features of specific families.
- Main ideas:
  - Use AV scanners to assign labels to malware binaries.
  - Use machine learning to classify binaries among known malware families.



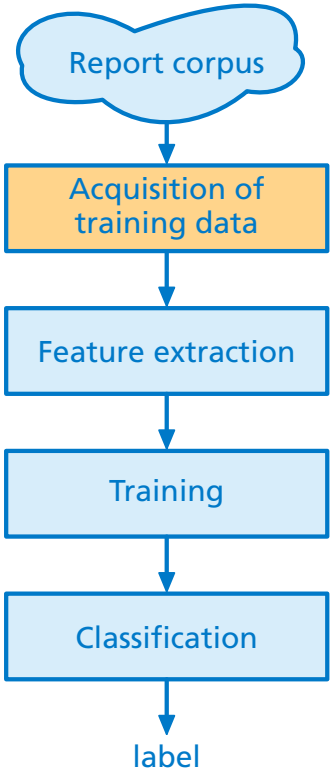
Malware classification: an overview

- Acquisition of training data: creation of balanced data sets for multi-class classification.
- Feature extraction: embedding of reports in a feature space.
- Training: optimizaiton and model selection.
- Classification: combining results of single family classifiers.



Malware classification: data acquisition

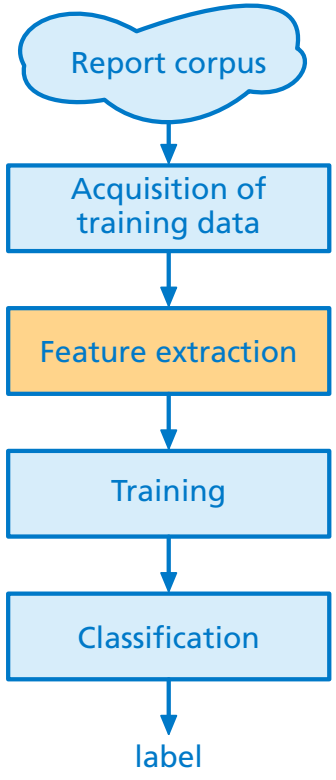
- Binaries are collected from Nepenthes and from spam-traps, and are analyzed by CWSandbox.
- Labels are assigned by running *Avira AntiVir*. Unrecognized binaries are discarded.
- 14 classes are considered for training: 1 backdoor, 2 trojans and 11 worms.



Malware classification: feature extraction

- Operational features: a set of all strings contained between delimeters “<” and “>”.
- “Wildcarding”: removal of potentially random attributes.

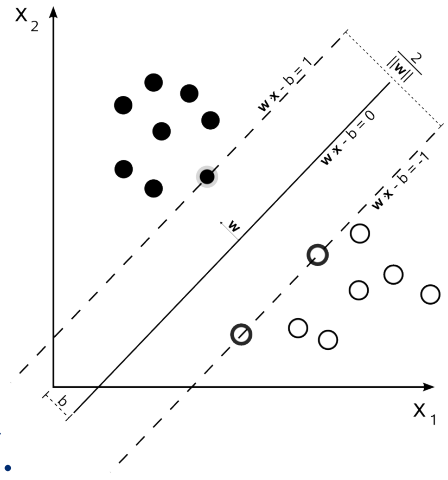
```
<copy_file filetype="File" srcfile="c:\1ae8b19ecea1b65705595b245f2971ee.exe"
  dstfile="C:\WINDOWS\system32\urdvxc.exe"
  creationdistribution="CREATE_ALWAYS" desiredaccess="FILE_ANY_ACCESS"
  flags="SECURITY_ANONYMOUS" fileinformationclass="FileBasicInformation"/>
<set_value key="HKEY_CLASSES_ROOT\CLSID\{3534943...2312F5C0&}"
  data="lsslwhxtettntbkr"/>
<create_process commandline="C:\WINDOWS\system32\urdvxc.exe /start"
  targetpid="1396" showwindow="SW_HIDE"
  apifunction="CreateProcessA" successful="1"/>
<create_mutex name="GhostBOT0.58b" owned="1"/>
<connection transportprotocol="TCP" remoteaddr="XXX.XXX.XXX.XXX"
  remoteport="27555" protocol="IRC" connectionestablished="1" socket="1780"/>
<irc_data username="XP-2398" hostname="XP-2398" servername="0"
  realname="ADMINISTRATOR" password="r0flc0mz" nick="[P33-DEU-51371]"/>
```



Malware classification: training

- For each known malware family, train a Support Vector Machine for separating this family for the others:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=0}^{M}\xi_i \\ \text{s.t.} \quad & y_i((w\cdot x_i)+b) \geq 1-\xi_i,\ i=1,\ldots,M. \\ & \xi_i \geq 0 \end{aligned}$$



- Determine the optimal parameter C by cross-validation

Classification of unknown malware binaries

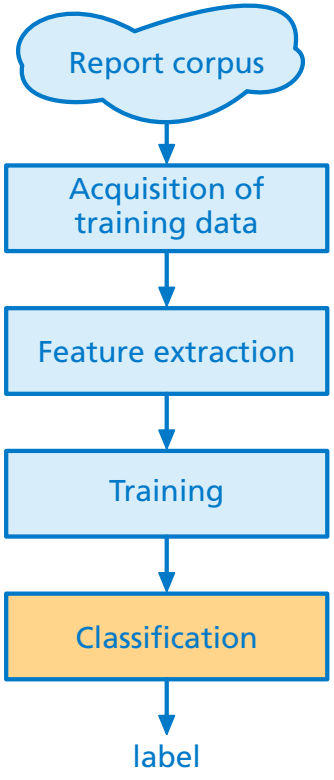
- Maximum distance: a label is assigned to a new report based on the highest score among the 14 classfiers:

$$d(x) = (w \cdot x) + b$$

- Maximum "likelihood": estimate conditional probability of class "+1" as:

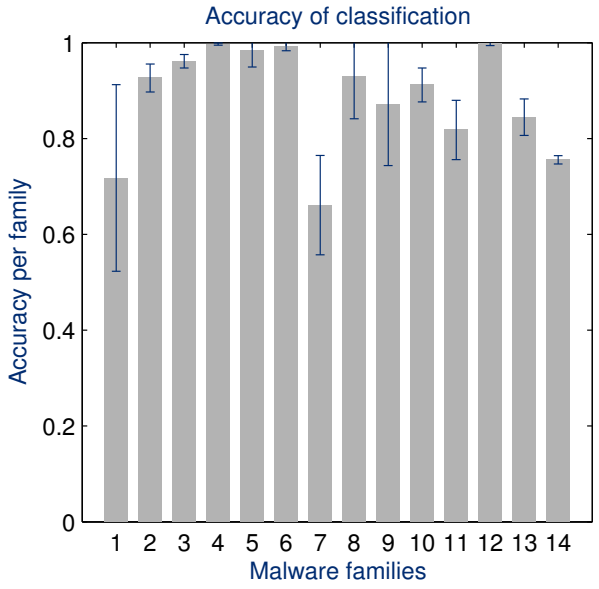
$$P(y = +1 \mid d(x)) = \frac{1}{1 + \exp(Ad(x) + B)}$$

where parameters A and B are estimated by a logistic regression fit on an independent training data set.

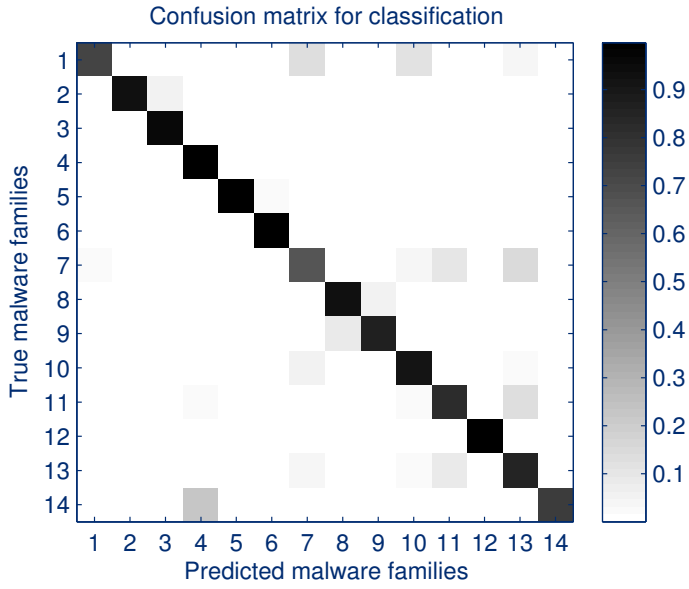


Results: known malware instances

- Test binaries are drawn from the same 14 families recognized by *AntiVir*.



(a) Accuracy per malware family

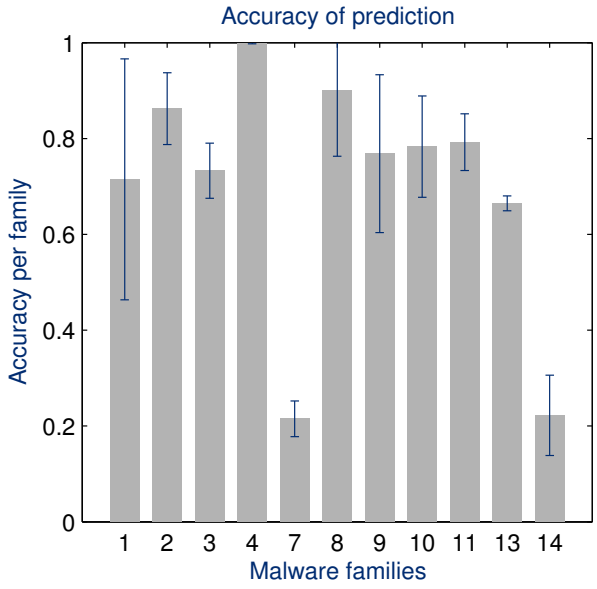


(b) Confusion of malware families

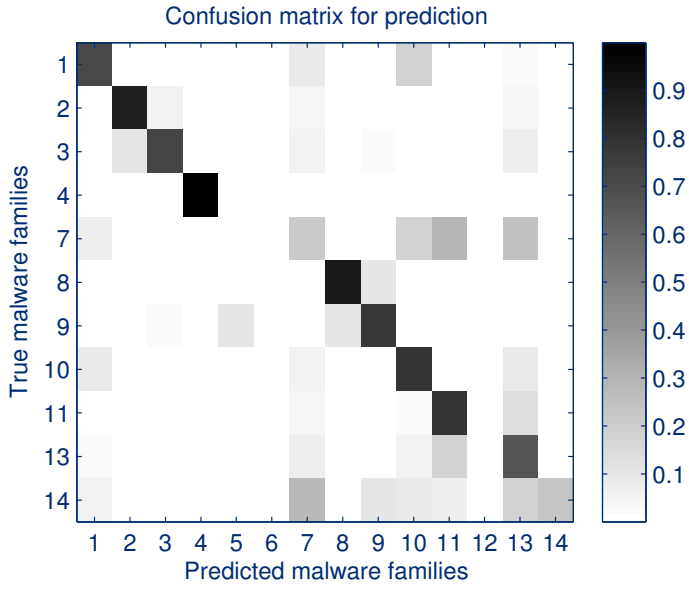
- Average accuracy: 88%.

Results: unknown malware instances

- Test binaries are drawn from the same 14 families recognized by *AntiVir* four weeks later.



(c) Accuracy per malware family



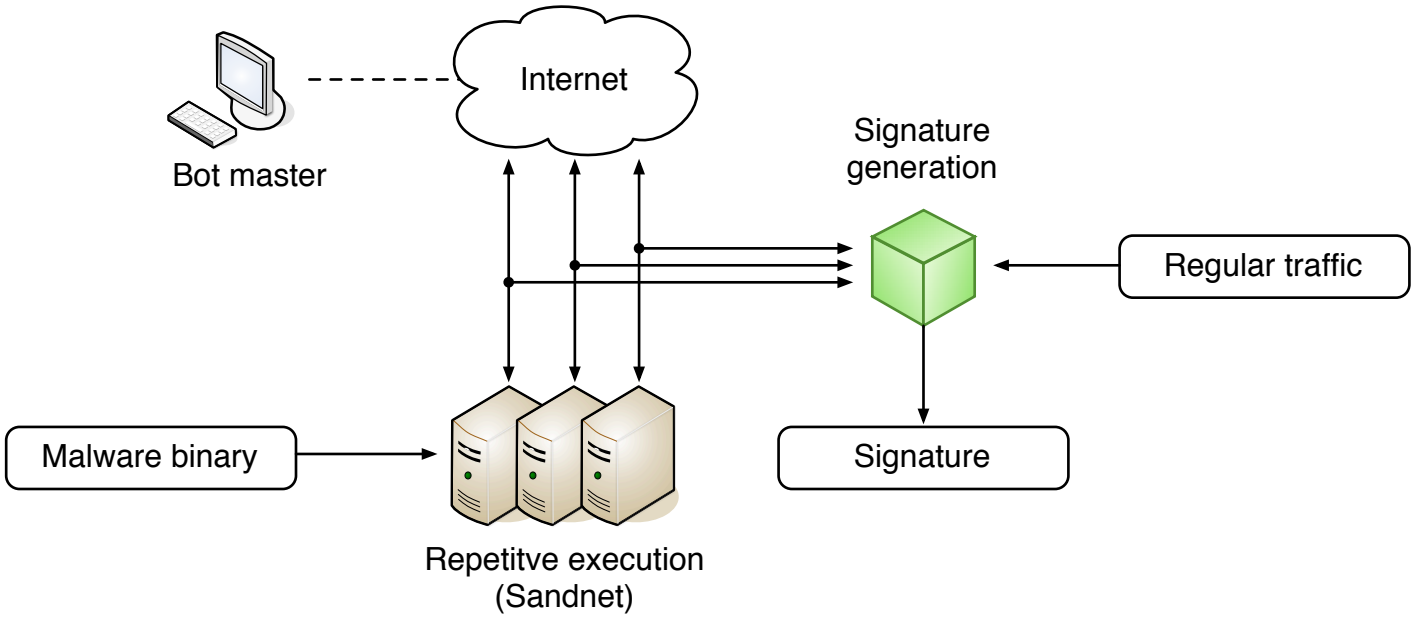
(d) Confusion of malware families

- Average accuracy: 69%.

Botzilla: analysis of malware communication

Operational goals:

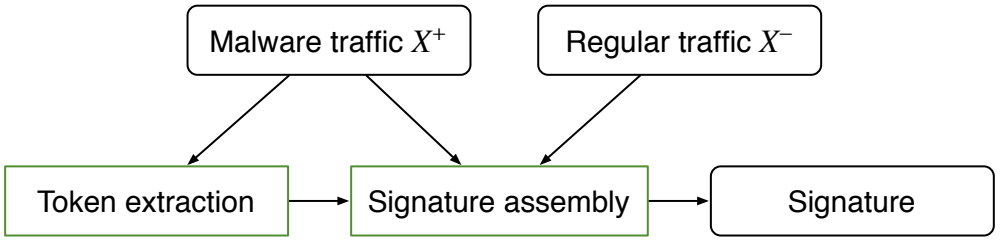
- Generation of signatures for malware communicatioin
- Detection of “drop-zones” and compromized machines



Execution of malware in a sandnet

- A sandnet is a monitored network of sandbox machines with various network configuration.
- Captured malware binaries are executed multiple times in a sandnet using various network settings (IP addresses, operating systems, date and time).
- All outgoing communication is captured in a tcpdump.
- Security precautions:
  - Redirection of SMTP traffic to a network sink
  - Blocking of DCOM and SMB requests (known vulnerabilities)

Signature generation



- Natural clustering: multiple executions of the same binary
- Extraction of (quasi)-distinct tokens using suffix trees
- Signature assembly: false positive check on regular traffic (real or simulated)
- Signature filtering: elimination of redundant signatures from similar binaries

Offline evaluation of Botzilla

- Procedure:
  - Signatures are generated for 43 classes of malware collected from a honeypot.
  - Test traffic is generated by running the same malware again and merging it with normal traffic from two sources (DARPA and University of Erlangen).
- Results:

	DARPA	Erlangen
Average detection rate	0.81	0.72
Average false alarm rate	0	$0.51 \cdot 10^{-6}$

Online evaluation of Botzilla

- Procedure:
  - Generated signatures were deployed on a large network at the University of Erlangen (ca. 50,000 hosts) using a Vermont network monitorin sensor.
  - Traffic statistics: ca. 840M flows in 4 days.
  - Alerts were manually evaluated.
- Results:

# malicious flows detected	219
# false alarms	295
Detection rate	0.43
False alarm rate	$3.51 \cdot 10^{-7}$

Signature examples

Banload keylogger:

```
tokens :
  0.5 : "SER asaasa510%0d%0aPASS 3330881%0d%0aTYPE I%0d%0aSYST%0d%0aPORT 10,0,"
  0.5 : "220 ProFTPD 1.2.9 Server (ProFTPD) [1.1.63]%0d%0a331 Password required for"
        "asaasa510.%0d%0a230 User asaasa510 logged in.%0d%0a200 Type set to I%0d%0a215 "
  0.5 : "UNIX Type: L8%0d%0a500 Illegal PORT command%0d%0a"
threshold : 1.00
```

Storm worm:

```
tokens :
  ef bf bd 50 00 ef bf bd 0a                  : 0.920
  ef bf bd ef bf bd ef bf bd ef bf bd ef ...  : 0.545
  ef bf bd 0d ef bf bd 0d ef bf bd 0d ef ...  : 0.339
  40 ef bf bd 3c ef bf bd 50 00 ef bf bd 0c   : 1.000
  40 ef bf bd 3c ef bf bd 50 00 ef bf bd 1b   : 0.679
  40 ef bf bd 3c ef bf bd 50 00 ef bf bd ...  : 0.581
threshold : 1.00
```

Lessons learned

- Malware analysis significantly facilitates malware understanding and the development of protection mechanisms.
- The main technique of malware analysis is execution of malware in a specially instrumented environment.
- Further analysis techniques require extensive machine learning and string matching instrumentation.

Recommended reading

Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Düssel, and Pavel Laskov.

Learning and classification of malware behavior.

In *Detection of Intrusions and Malware, and Vulnerability Assessment, Proc. of 5th DIMVA Conference*, pages 108–125, 2008.

Konrad Rieck, Guido Schwenk, Tobias Limmer, Thorsten Holz, and Pavel Laskov.

Botzilla: Detecting the phoning homeöf malicious software.

In *Proc. of 25th ACM Symposium on Applied Computing (SAC)*, March 2010.

(to appear).

Carsten Willems, Thorsten Holz, and Felix Freiling.

CWSandbox: Towards automated dynamic binary analysis.

IEEE Security and Privacy, 5(2):32–39, 2007.