Analysis and Classification of Android Malware

Nov 13, 2017

Department of Computer Science and Engineering
University of Bologna

Lorenzo Cavallaro

<lorenzo.cavallaro@rhul.ac.uk>

Research partially supported by the UK EPSRC grants EP/K033344/1 and EP/L022710/1

Royal Holloway, University of London

- Founded in 1879 by Thomas Holloway
  - → Entrepreneur and Philanthropist
  - → Holloway's pills and ointments
- ~10,000 students across 3 Faculty (Arts and Social Sciences, Management and Economics, and Science)
- Egham–still commuting distance to London!
- Featured in Avengers: Age of Ultron :-)

Academic Centre of Excellence in Cyber Security Research

- 1 of 14 in the UK, since its incipit in 2012

Centre for Doctoral Training in Cyber Security

- 1 of 2 in the UK, since its incipit in 2012

SYSTEMS SECURITY RESEARCH LAB

s2lab.isg.rhul.ac.uk

S^2Lab | People | Projects | Publications | Reading Group | Teaching | Vacancies

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

The research carried out at the Systems Security Research Lab (S^2Lab) within the Information Security Group (ISG) at Royal Holloway University of London focuses on devising novel techniques to protect systems from a broad range of threats, including those perpetrated by malicious software. In particular, we aim at building practical tools and provide security services to the community at large. Our research, kindly sponsored by the UK Engineering and Physical Sciences Research Council (EPSRC), the European Council Framework Programme 7 (EU FP7), and Intel Security (McAfee Labs UK), crosses the boundaries of a number of different Computer Science-related topics, such as operating systems, computer architecture, program analysis, and machine learning, making our challenging journey always exciting.

Learn About Our Research Projects

ROYAL HOLLOWAY UNIVERSITY OF LONDON

s2lab.isg.rhul.ac.uk

Systems Security Research Lab

- Founded on Sep 2014
- 2 Faculty staff, ~10 researchers (PhD and UROP students, visiting scholars)
- Engagement with industry to promote research impact
  E.g., VISA Research, Android Security @ Google, and McAfee Labs

Sep 2014– Funding in excess of 1.5M GBP (Principal Investigator)

→ Equipment 70+TB storage (repl.), 150+ vCores, 1.5+TB RAM, Tesla K40 GPU

→ Program analysis and machine learning infrastructure

→ Android security, malware analysis/detection, vulnerability discovery/analysis, automatic exploit generation

Vision

S2Lab's underpinning research builds on program analysis and machine learning to address threats against the security of computing systems

- Practical tools to provide security services to the community at large
- http://s2lab.isg.rhul.ac.uk

EPSRC Pioneering research and skills

intel Security

ROYAL HOLLOWAY UNIVERSITY OF LONDON

National Cyber Security Centre a part of GCHQ

NVIDIA

Selected Highlights

- Malware analysis and (open challenges in) ML classification
  [USENIXSec17] Detecting Concept Drift in Malware Classification Models
  [IEEE TIFS17] Understanding Android App Piggybacking: A Systematic Study of Malicious Code Grafting
  [NDSS15] CopperDroid: Automatic Reconstruction of Android Malware Behaviors
- Software understanding and hardening
  [NDSS17] Stack Object Protection with Low Fat Pointers
- Automatic exploit generation
  [ACM CCS-PLAS17] Modular Synthesis of Heap Exploits
- Vulnerability discovery
  [arXiv17] BabelView: Evaluating the Impact of Code Injection Attacks in Mobile Webviews

$ WHOAMI

- Antifork Research
- s0ftpj
- Metro Olografix

$ WHOAMI

- Antifork Research
- s0ftpj
- Metro Olografix

- BSc & MSc in Computer Science
- PhD in Computer Science (Computer Security)

$ WHOAMI

- Antifork Research
- s0ftpj
- Metro Olografix

- BSc & MSc in Computer Science
- PhD in Computer Science (Computer Security)

- 2006-2008: Visiting PhD Scholar – Prof. R. Sekar
- Systems security (mem err, taint tracking, anomaly detection)

- 2008-2010: PostDoc – Profs G. Vigna & C. Kruegel
- Malware analysis & detection (mostly botnet)

- 2010-2012: PostDoc – Prof. A. S. Tanenbaum
- OS Dependability (MINIX3) & Systems Security

$ WHOAMI

- Antifork Research
- s0ftpj
- Metro Olografix

- BSc & MSc in Computer Science
- PhD in Computer Science (Computer Security)

- 2006-2008: Visiting PhD Scholar – Prof. R. Sekar
- Systems security (mem err, taint tracking, anomaly detection)

- 2008-2010: PostDoc – Profs G. Vigna & C. Kruegel
- Malware analysis & detection (mostly botnet)

- 2010-2012: PostDoc – Prof. A. S. Tanenbaum
- OS Dependability (MINIX3) & Systems Security

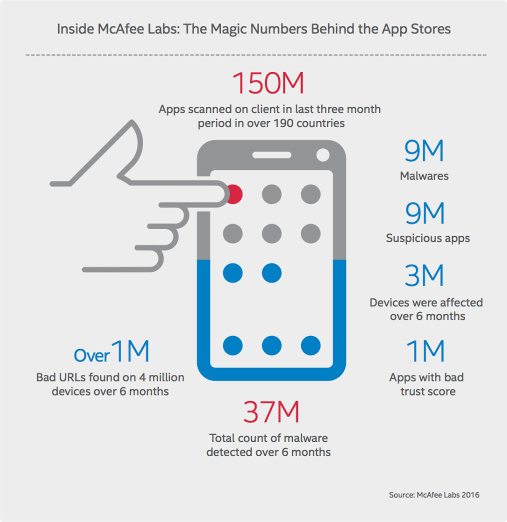
2016–Reader (Associate Professor) of Information Security

Google says there are now 1.4 billion active Android devices worldwide

BY JOHN CALLAHAM Tuesday, Sep 29, 2015 at 12:13 pm EDT

7 Comments

THE RISE IN ANDROID MALWARE



McAfee Labs routinely scans major app stores for infected systems as well as apps with suspicious behavior.

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

STATUS QUO

Year	Method	Venue	Type Det	Type Class	Feature	# Malware	DR/FP (%)	ACC (%)	
2014	DroidAPIMiner	SecureComm	✓	–	API,PKG,PAR	3,987	99/2.2	–	
2014	DroidMiner	ESORICS	✓	✓	CG,API	2,466	95.3/0.4	92	
2014	Drebin	NDSS	✓	–	PER,STR,API,INT	5,560	94.0/1.0	–	
2014	DroidSIFT	ACM CCS	✓	✓	API-F	2,200	98.0/5.15	93	
2014	DroidLegacy	ACM PPREW	✓	✓	API	1,052	93.0/3.0	98	
2015	AppAudit	IEEE S&P	✓	–	API-F	1,005	99.3/0.61	–	
2015	MudFlow	ICSE	✓	–	API-F	10,552	90.1/18.7	–	
2015	Marvin	ACM COMPSAC	✓	–	PER, INT, ST, PN	15,741	98.24/0.0	–	
2015	RevealDroid	TR GMU	✓	✓	PER,API,API-F,INT,PKG	9,054	98.2/18.7	93	
2017	MaMaDroid	NDSS	✓	–	Abstract APIs Markov Chain	80,000	99/1	–	
2017	DroidSieve	ACM CODASPY	✓	✓	Syntactic- & Resource-centric	100,000	99.7/0	–	
2016	Madam	IEEE TDSC	✓	–	SYSC, API, PER, SMS, USR	2,800	96/0.2	–	–

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

STATUS QUO

Year	Method	Venue	Type Det	Type Class	Feature	# Malware	DR/FP (%)	ACC (%)	
2014	DroidAPIMiner	SecureComm	✓	–	API,PKG,PAR	3,987	99/2.2	–	
2014	DroidMiner	ESORICS	✓	✓	CG,API	2,466	95.3/0.4	92	
2014	Drebin	NDSS	✓	–	PER,STR,API,INT	5,560	94.0/1.0	–	
2014	DroidSIFT	ACM CCS	✓	✓	API-F	2,200	98.0/5.15	93	
2014	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1,052	93.0/3.0	98	
2015	AppAudit	IEEE S&P	✓	–	API-F	1,005	99.3/0.61	–	
2015	MudFlow	ICSE	✓	–	API-F	10,552	90.1/18.7	–	
2015	Marvin	ACM COMPSAC	✓	–	PER, INT, ST, PN	15,741	98.24/0.0	–	
2015	RevealDroid	TR GMU	✓	✓	PER,API,API-F,INT,PKG	9,054	98.2/18.7	93	
2017	MaMaDroid	NDSS	✓	–	Abstract APIs Markov Chain	80,000	99/1	–	
2017	DroidSieve	ACM CODASPY	✓	✓	Syntactic & Resource-centric	100,000	99.7/0	–	
2016	[illegible]	[illegible]	[illegible]	[illegible]	[illegible], SMS, USR	2,800	96/0.2	–	–

- Most focus on statically-extracted features
  - → Issues with obfuscation, dynamically & native code
- How far would dynamically-extracted features go?

RQ1 Automatic reconstruction of apps behaviors
(As fewer features as possible with rich semantics)

RQ2 Machine learning with high-accuracy results
(Challenging contexts: classification, sparse behaviors)

RQ3 Decaying machine learning models: concept drift
(Evaluate the quality of a classifier to identify drifting objects)

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

RQ1—CopperDroid

Automatic Reconstruction of Android Apps Behaviors

DYNAMIC ANALYSIS FOR ANDROID

- DroidScope/DECAF[1]
  - → Dalvik VM method, asm insn, and system call tracing
  - → 2-level VMI to get to Dalvik VM semantics
- Droidbox[2] and TaintDroid[3]
- Other approaches generally built on top of the tools above

[1] https://github.com/sycurelab/DECAF/tree/master/DroidScope/qemu
[2] https://github.com/pjlantz/droidbox
[3] http://www.appanalysis.org/

SYSTEM CALL-CENTRIC ANALYSIS

- Established technique to characterize process behaviors[4]
- Identifying state-modifying actions crucial to analysis

[4] https://www.cs.unm.edu/%2E/forrest/publications/acsac08.pdf

SYSTEM CALL-CENTRIC ANALYSIS

- Established technique to characterize process behaviors[4]
- Identifying state-modifying actions crucial to analysis

Can it be applied to Android?

- Android architecture is different from traditional devices
- State-modifying actions manifest at multiple abstractions
  - → OS interactions (e.g., filesystem, network, process)
  - → Android-specific behaviors (e.g., SMS, phone calls)

[4] https://www.cs.unm.edu/%2E/forrest/publications/acsac08.pdf

Key Insight

System calls provide the right semantic abstraction given the reconstruction of Inter-Component Communications (ICC) behaviors

- ICC (aka Binder transactions) are carried out as **ioctl** system calls
  - → CopperDroid automatically unmarshalls such calls and reconstruct Android app behaviors[a]
  - → No modification to the OS
  - → It works automatically across the Android fragmented ecosystem

[a] Kimberly Tam, Salahuddin J. Khan, Aristide Fattori, and Lorenzo Cavallaro. **CopperDroid: Automatic Reconstruction of Android Malware Behaviors**. In 22nd Annual Network and Distributed System Security Symposium (NDSS), 2015

[4] https://www.cs.unm.edu/%2E/forrest/publications/acsac08.pdf

THE BINDER PROTOCOL

IPC/RPC

- Binder protocols enable fast inter-process communication
- Allows apps to invoke other app component functions
- Binder objects handled by Binder Driver in kernel
  - → Serialized/marshalled passing through kernel
  - → Results in input output control (`ioctl`) system calls

Android Interface Definition Language (AIDL)

- AIDL defines which/how services can be invoked remotely
- Describes how to marshal method parameters

IPC BINDER: AN EXAMPLE

Application

```
PendingIntent sentIntent = PendingIntent.getBroadcast(SMS.this,
0, new Intent("SENT"), 0);
SmsManager sms = SmsManager.getDefault();
sms.sendTextMessage("7855551234", null, "Hi␣There", sentIntent, null);
```

IPC BINDER: AN EXAMPLE

Application

android.telephony.SmsManager

```
public void sendTextMessage(...) {
    ...
    ISms iccISms = ISms.Stub.asInterface(ServiceManager.getService("isms"));
    if (iccISms != null)
        iccISms.sendText(destinationAddress, scAddress, text, sentIntent, deliveryIntent);
    ...
```

IPC BINDER: AN EXAMPLE

Application

↓

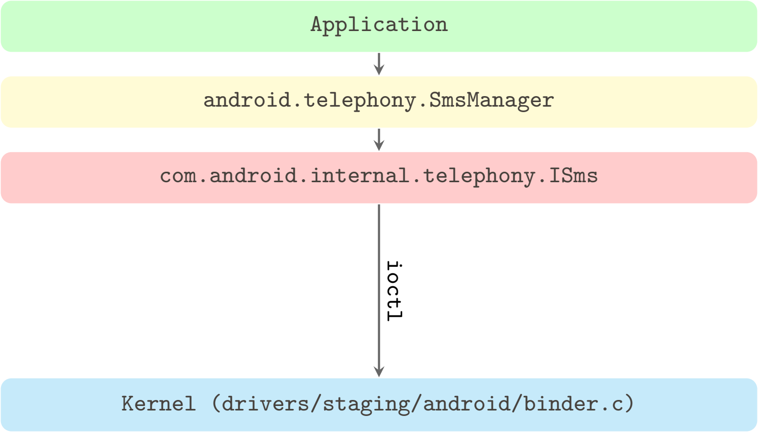
android.telephony.SmsManager

↓

com.android.internal.telephony.ISms

```
public void sendText(...) {
    android.os.Parcel _data = android.os.Parcel.obtain();
    try {
        _data.writeInterfaceToken(DESCRIPTOR);
        _data.writeString(destAddr);
        ...
        mRemote.transact(Stub.TRANSACTION_sendText, _data, _reply, 0);
    }
```

IPC BINDER: AN EXAMPLE



SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

IPC BINDER: AN EXAMPLE

Application

↓

android.telephony.SmsManager

↓

com.android.internal.telephony.ISms

```
ioctl(4, 0xc0186201, ...
  \x4b\x00\x00\x00\x49\x00\x20\x00\x74\x00\x61\x00
  \x6b\x00\x65\x00\x20\x00\x70\x00\x6c\x00\x65\x00
  \x61\x00\x73\x00\x75\x00\x72\x00\x65\x00\x20\x00
  \x69\x00\x6e\x00\x20\x00\x68\x00\x75\x00\x72\x00
  \x74\x00\x69\x00\x6e\x00\x67\x00\x20\x00\x73\x00 ...)
```

↓

Kernel (drivers/staging/android/binder.c)

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

IPC BINDER: AN EXAMPLE

Application

↓

android.telephony.SmsManager

↓

com.android.internal.telephony.ISms

```
ioctl(/dev/binder, BINDER_WRITE_READ, ...
  InterfaceToken = com.android.internal.telephony.ISms,
  method: sendText,
  destAddr = "7855551234",
  scAddr = null,
  text = "Hi␣There",
  sentIntent = Intent("SENT"),
  deliverIntent = null)
```

Kernel (drivers/staging/android/binder.c)

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

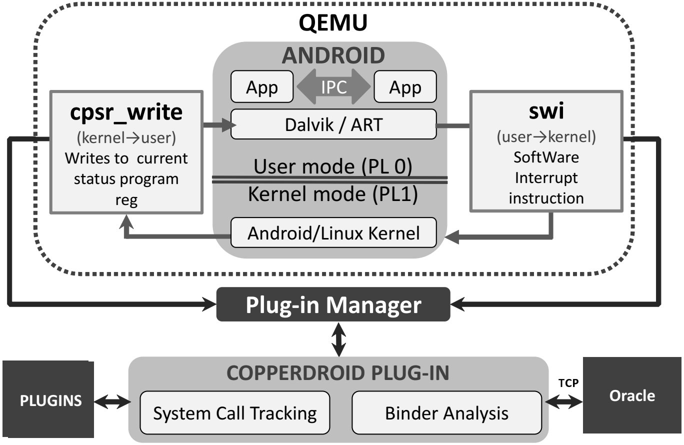
A system call induces a User -> Kernel transition

- On ARM invoked through the `swi` instruction (SoftWare Interrupt)
- `r7`: invoked system call number
- `r0-r5`: parameters
- `lr`: return address

CopperDroid's Approach

- instruments QEMU's emulation of the `swi` instruction
- instruments QEMU to intercept every `cpsr_write` (Kernel → User)
- Perform traditional VMI to associate system calls to threads

TRACING SYSTEM CALLS ON ANDROID ARM THROUGH QEMU

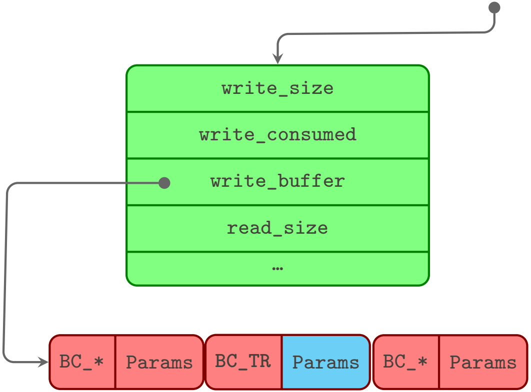


SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

CopperDroid inspects the Binder protocol in detail by intercepting a subset of the `ioctl`s issued by userspace Apps.

```
ioctl(binder_fd, BINDER_WRITE_READ, &binder_write_read);
```

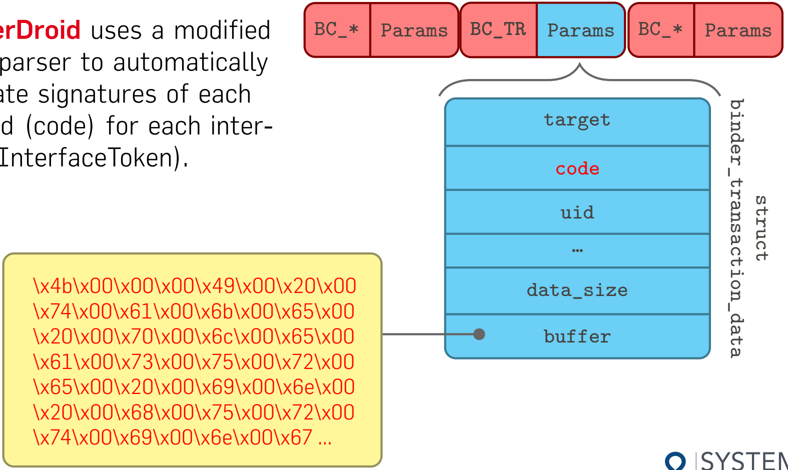


SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

CopperDroid analyzes `BC_TRANSACTION`s and `BC_REPLY`s

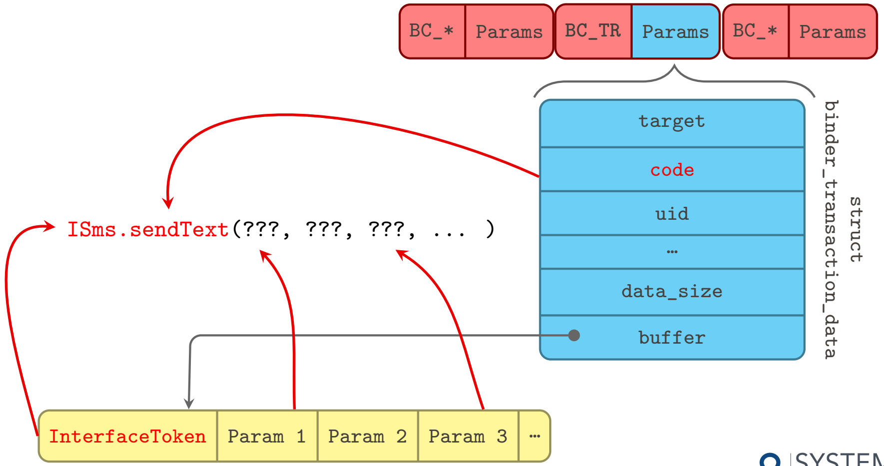
CopperDroid uses a modified AIDL parser to automatically generate signatures of each method (code) for each interface (InterfaceToken).



SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

CopperDroid analyzes `BC_TRANSACTION`s and `BC_REPLY`s

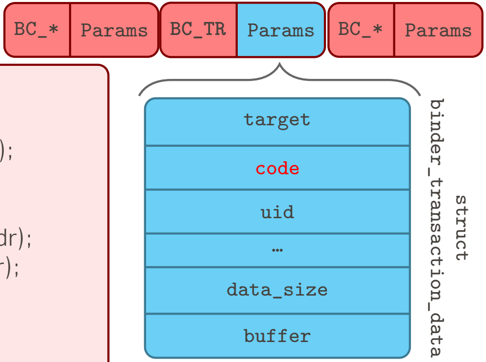


SYSTEMS SECURITY RESEARCH LAB
ROYAL HOLLOWAY UNIVERSITY OF LONDON

AUTOMATIC BINDER UNMARSHALLING

CopperDroid analyzes `BC_TRANSACTION`s and `BC_REPLY`s

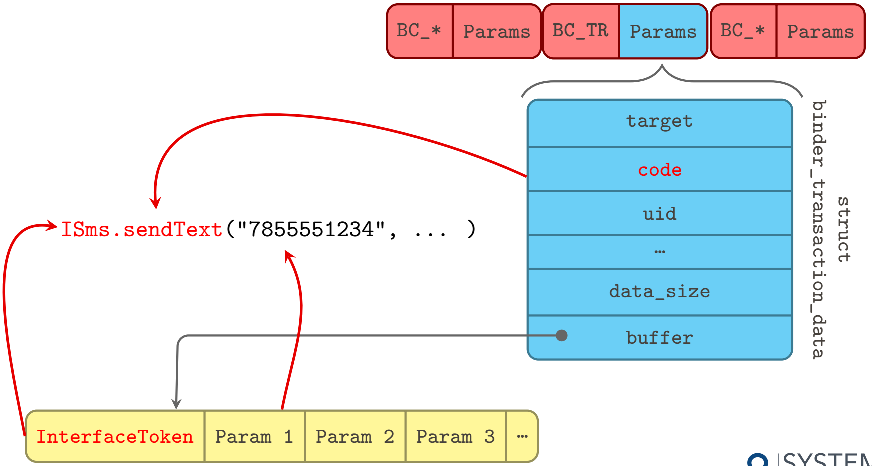
```
public void sendText(...) {
   android.os.Parcel _data =
       android.os.Parcel.obtain();
   try {
      ...
      _data.writeString(destAddr);
      _data.writeString(srcAddr);
      _data.writeString(text);
      ...
      mRemote.transact(
            Stub.TRANSACTION_sendText,
            _data, _reply, 0);
   }
```



SYSTEMS SECURITY RESEARCH LAB
ROYAL HOLLOWAY UNIVERSITY OF LONDON

AUTOMATIC BINDER UNMARSHALLING

CopperDroid analyzes `BC_TRANSACTION`s and `BC_REPLY`s

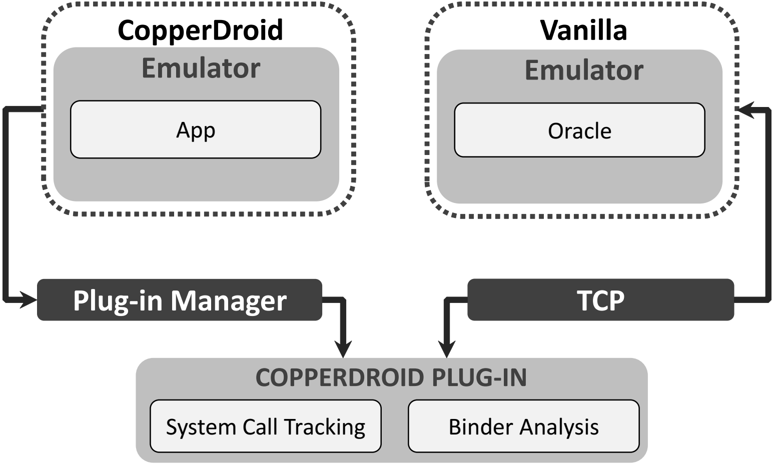


SYSTEMS SECURITY RESEARCH LAB
ROYAL HOLLOWAY UNIVERSITY OF LONDON

AUTOMATIC ANDROID OBJECTS UNMARSHALLING

- Primitive types (e.g., String text)
  - → A few manually-written procedures
- Complex Android objects
  - → 300+ Android objects–manual unmarshalling: does not scale & no scientific
  - → Finds object CREATOR field
  - → Use reflection (type introspection, then intercession)
- IBinder object reference
  - → A handle (pointer) sent instead of marshalled object
  - → Look earlier in trace to map each handle to an object

CopperDroid's Oracle unmarshalls all three automatically



SYSTEMS SECURITY RESEARCH LAB
ROYAL HOLLOWAY UNIVERSITY OF LONDON

AUTOMATIC UNMARSHALLING ORACLE: SMS EXAMPLE

TYPE	"string", "string", "string", "PendingIntent", "PendingIntent"
DATA	\x0A \x00 \x00 \x00 \x34 \x00 \x38 \x00 \x35 \x00 \x35 \x00 \x35 \x00 \x35 \x00 \x31 \x00 \x32 \x00 \x33 \x00 \x34 \x00 \x00 \x00 \x00 \x08 \x00 \x00 \x00 \x48 \x00 \x69 \x00 \x20 \x00 \x74 \x00 \x68 \x00 \x65 \x00 \x72 \x00 \x65 \x00 \x85*hs \x7f \x00 \x00 \x00 \xa0 \x00 \x00 \x00 \x00 \x00 \x00 \x00 ...
OUTPUT	

SYSTEMS SECURITY RESEARCH LAB
ROYAL HOLLOWAY UNIVERSITY OF LONDON

AUTOMATIC UNMARSHALLING ORACLE: SMS EXAMPLE

TYPE	"string", "string", "string", "PendingIntent", "PendingIntent"
DATA	\x0A \x00 \x00 \x00 \x34 \x00 \x38 \x00 \x35 \x00 \x35 \x00 \x35 \x00 \x35 \x00 \x31 \x00 \x32 \x00 \x33 \x00 \x34 \x00 \x00 \x00 \x00 \x00 \x08 \x00 \x00 \x00 \x48 \x00 \x69 \x00 \x20 \x00 \x74 \x00 \x68 \x00 \x65 \x00 \x72 \x00 \x65 \x00 \x85*hs \x7f \x00 \x00 \x00 \xa0 \x00 \x00 \x00 \x00 \x00 \x00 \x00 ...
OUTPUT	telephony.ISms.sendText(String destAddr = "7855551234", ...)

- Type[0] = `Primitive` `"string"`
- Use `ReadString()` (and increment data offset by length of string)

AUTOMATIC UNMARSHALLING ORACLE: SMS EXAMPLE

TYPE	`"string", "string", "string", "PendingIntent", "PendingIntent"`
DATA	`\x0A \x00 \x00 \x00 \x34 \x00 \x38 \x00 \x35 \x00 \x35 \x00 \x35 \x00 \x35 \x00 \x31 \x00 \x32 \x00 \x33 \x00 \x34 \x00 \x00 \x00 \x00 \x00 \x08 \x00 \x00 \x00 \x48 \x00 \x69 \x00 \x20 \x00 \x74 \x00 \x68 \x00 \x65 \x00 \x72 \x00 \x65 \x00 \x85*hs \x7f \x00 \x00 \x00 \xa0 \x00 \x00 \x00 \x00 \x00 \x00 \x00 ...`
OUTPUT	`com.android.internal.telephony.ISms.sendText( String destAddr = "7855551234", String srcAddr = null, String text = "Hi there", ... )`

- Type[1] and Type[2] are also **`Primitive`** **`"string"`**
- Use **`ReadString()`** (and increment data offset by length of strings)

AUTOMATIC UNMARSHALLING ORACLE: SMS EXAMPLE

TYPE	"string","string", "string", **"PendingIntent"**, "PendingIntent"
DATA	\x0A \x00 \x00 \x00 \x34 \x00 \x38 \x00 \x35 \x00 \x35 \x00 \x35 \x00 \x35 \x00 \x31 \x00 \x32 \x00 \x33 \x00 \x34 \x00 \x00 \x00 \x00 \x00 \x08 \x00 \x00 \x00 \x48 \x00 \x69 \x00 \x20 \x00 \x74 \x00 \x68 \x00 \x65 \x00 \x72 \x00 \x65 \x00 \x85*hs \x7f \x00 \x00 \x00 \xa0 \x00 \x00 \x00 \x00 \x00 \x00 \x00 ...
OUTPUT	com.android.internal.telephony.ISms.sendText(String destAddr = "7855551234", String srcAddr = null, String text = "Hi there", Intent sentIntent { type = BINDER_TYPE_HANDLE, flags = 0x7F | FLAT_BINDER_FLAG_ACCEPT_FDS handle = 0xa, cookie = 0x0 }, ...)

- Type[3] = `IBinder` `"PendingIntent"`
- Unmarshal using `com.Android.Intent` (AIDL) and increment buffer pointer
- Handle points to data to be unmarshalled in a previous Binder (`ioctl`) call

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

AUTOMATIC UNMARSHALLING ORACLE: SMS EXAMPLE

TYPE	"string","string", "string", **"PendingIntent"**, "PendingIntent"
DATA	\x0A \x00 \x00 \x00 \x34 \x00 \x38 \x00 \x35 \x00 \x35 \x00 \x35 \x00 \x35 \x00 \x31 \x00 \x32 \x00 \x33 \x00 \x34 \x00 \x00 \x00 \x00 \x00 \x08 \x00 \x00 \x00 \x48 \x00 \x69 \x00 \x20 \x00 \x74 \x00 \x68 \x00 \x65 \x00 \x72 \x00 \x65 \x00 **\x85*hs \x7f \x00 \x00 \x00 \xa0 \x00 \x00 \x00 \x00 \x00 \x00 \x00 ...**
OUTPUT	com.android.internal.telephony.ISms.sendText(String destAddr = "7855551234", String srcAddr = null, String text = "Hi there", Intent sentIntent { **Intent("SENT")** }, ...)

- Each handle is paired with a parcelable object
- CopperDroid sends each handle and parcelable object to the Oracle

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

Outputs Observed from CopperDroid

FILESYSTEM TRANSACTIONS


```
"class": "FS ACCESS",
"low": [
    {
        "blob": "{'flags': 131072, 'mode': 1, 'filename': u'/etc/media_codecs.xml'}",
        "id": 187369,
        "sysname": "open",
        "ts": "1455718126.798",
    },
    {
        "blob": "{'size': 4096L, 'filename': u'/etc/media_codecs.xml'}",
        "id": 187371,
        "sysname": "read",
        "ts": "1455718126.798",
        "xref": 187369
    },
    {
        "blob": "{'filename': u'/etc/media_codecs.xml'}",
        "id": 187389,
        "sysname": "close",
        "ts": "1455718126.799",
        "xref": 187369
    }
],
"procname": "/system/bin/mediaserver"
```

NETWORK TRANSACTIONS


```
"class": "NETWORK ACCESS",
"low": [
    {
        "blob": "{'socket domain': 10, 'socket type': 1, 'socket protocol': 0}",
        "id": 62,
        "sysname": "socket",
        "ts": "1445024980.686",
    },
    {
        "blob": "{'host': '::ffff:134.219.148.11', 'port': 80, 'returnValue': 0}",
        "id": 63,
        "sysname": "connect",
        "ts": "1445024980.687",
    },
    {
        "blob": "=%22%27GET+%2Findex.html+HTTP%2F1.1%5C%5Cr%5C%5CnUser-Agent%3A+Dalvik%2F1.6.0+%28Linux%3B+U%3B+Android+4.4.4%3B+sdk+Build%2FKK%29%5C%5Cr%5C%5CnHost%3A+s2lab.isg.rhul.ac.uk%5C%5Cr%5C%5CnConnection%3A+Keep-Alive%5C%5Cr%5C%5CnAccept-Encoding%3A+gzip%5C%5Cr%5C%5Cn%5C%5Cr%5C%5Cn%27%22",
        "id": 164,
        "sysname": "sendto",
        "ts": "1445024980.720",
    },
],
"procname": "com.cd2.nettest.nettest",
"subclass": "HTTP"
```

NETWORK TRANSACTIONS


```
"class": "NETWORK ACCESS",
"low": [
    {
        "blob": "{'socket domain': 10, 'socket type': 1, 'socket protocol': 0}",
        "id": 62,
```


- Composite behaviors (e.g., filesystem and network transactions)
- We perform a value-based data flow analysis by building a system call-related DDG and def-use chains
  - → Each observed system call is initially considered as an unconnected node
  - → Forward slicing inserts edges for every inferred dependence between two calls
    - → Nodes and edges are annotated with the system call argument constraints
    - → Annotations needed for the creation of def-use chains
  - → Def-use chains relate the output value of specific system calls to the input of (non-necessarily adjacent) others


```
],
"procname": "com.cd2.nettest.nettest",
"subclass": "HTTP"
```


SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

BINDER TRANSACTIONS


```
"class": "SMS SEND",
"low": [
    {
        "blob": {
            "method": "sendText",
            "params": [
                "callingPkg = com.load.wap",
                "destAddr = 3170",
                "scAddr = null",
                "text = 999287346 418 Java (256) vip 2012-02-25 17:47:56 newoperastore.ru y"
            ]
        },
        "method_name": "com.android.internal.telephony.ISms.sendText()",
        "sysname": "ioctl",
        "ts": "1444337887.816",
        "type": "BINDER"
    }
],
"procname": "com.load.wap"
```

COMING UP

- The unmarshalling oracle is the bottleneck
  - → Solution: program analysis on the Android OS framework
    - → Forward and backward slicing on Parcelable Android classes
    - → Type system
    - → Working prototype for the most common Parceleable Android classes
- Full Binder behavior reconstruction
  - → Devil is in the details: piggybacked Binder requests & replies, references, etc
- Python bindings (CopperDroid plug-in handles parceling and unparceling), e.g.:

```
def SendTextTransaction(Task, Object):
    Object.SetText("01234")
```

  - → It eases other analyses, e.g.:
    - → ICC fuzzing
    - → ICC policy enforcement (requires porting to the device to be useful)
    - → Information leakage detection via differential behavioral analysis
- RESTful API is being integrated, alive "soon"

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

RQ2—DroidScribe

Classifying Android Malware with Runtime Behavior

- Runtime behaviors as discriminator of maliciousness
  - → Independent of any syntactic artifact
  - → Visible in managed and native code alike
- Family Identification
  - → Crucial for analysis of threats and mitigation planning

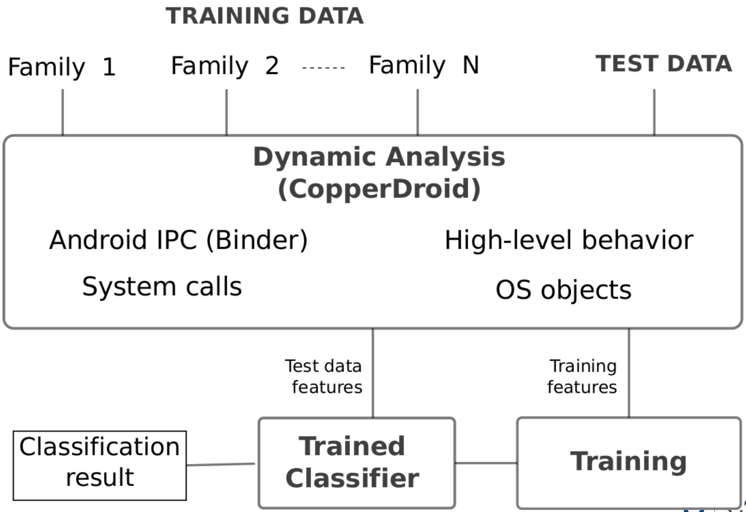
Goal Dynamic analysis for classification **challenging conditions**

Our contributions[5]

- RQ2.1: What is the best level abstraction?
- RQ2.2: Can we deal with sparse behaviors?

[5] Dash et al., ``DroidScribe: Classifying Android Malware Based on Runtime Behavior'', in IEEE S&P Workshop MoST 2016

OVERVIEW OF THE CLASSIFICATION FRAMEWORK

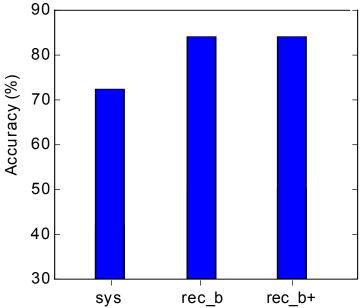


SYSTEMS SECURITY RESEARCH LAB
ROYAL HOLLOWAY UNIVERSITY OF LONDON

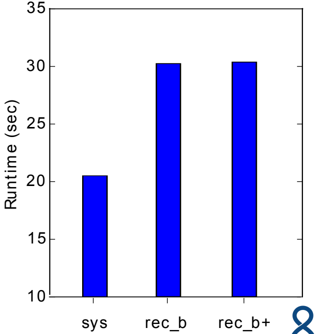
SYSTEM-CALLS VS. ABSTRACT BEHAVIORS

RQ2.1 What is the best level of abstraction?

- Experiments on the Drebin dataset (5,246 malware samples).
- Reconstructing Binder calls adds 141 meaningful features.
- High level behaviors added 3 explanatory features.



(a) Accuracy



(b) Runtime

SET-BASED PREDICTIONS

- Dynamic analysis is limited by code coverage
- Classifier has only partial information about behaviors
- Identify when malware cannot be reliable classified into only one family
  - → Based on a measure of the statistical confidence
- Helpful human analyst by identifying the top matching families, supported by statistical evidence

Conformal Predictor (CP)

- A statistical learning algorithm for classification tasks
- Provides statistical evidence on the results

Credibility

Supports how good a sample fits into a class

Confidence

Indicates if there are other good choices

Robust Against Outliers

Aware of values from other members of the same class

COMPUTING P-VALUES

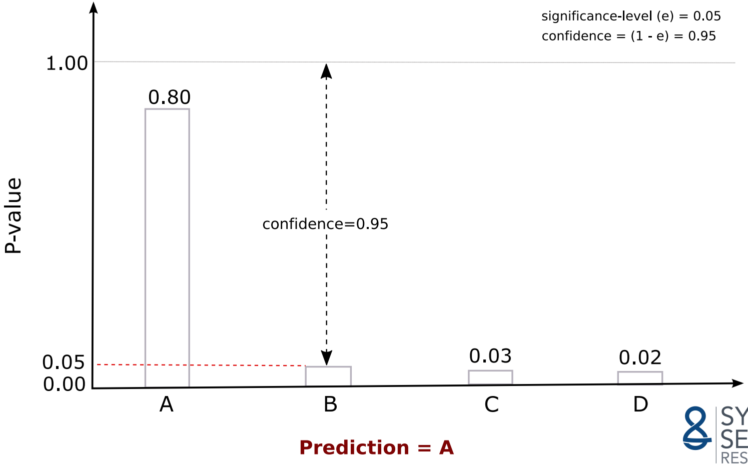
- Nonconformity Measure (NCM) is a geometric measure of how well a sample is far from a class.
  - → For SVM, the NCM $\mathscr{N}_D^z$ of a sample z w.r.t. class D is sum distances from all hyperplanes bounding the class D.

$$\mathscr{N}_D^z = \sum_i d(z, \mathscr{H}_i)$$

- P-value is a statistical measure of how well a sample fits in a class.
  - → P-value $\mathscr{P}_D^z$ represents the proportion of samples in D that are more different than z w.r.t. D.

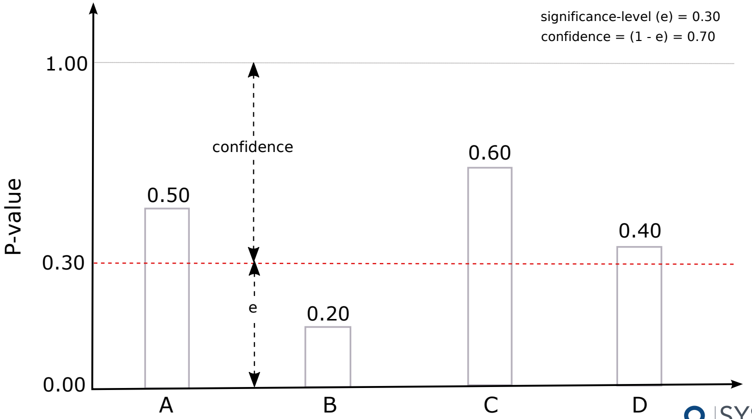
$$\mathscr{P}_D^z = \frac{|\{j = 1, \ldots, n : \mathscr{N}_D^j \geq \mathscr{N}_D^z\}|}{n}$$

Given a new object s, conformal predictor picks the class with the highest p-value and return a singular prediction.



OBTAINING PREDICTION SETS

Given a new object s, we can set a significance-level e for p-values and obtain a prediction set Γ^e includes labels whose p-value is greater than e for the sample.



Prediction Set = {A, C, D}

WHEN TO USE CONFORMAL PREDICTION?

- CP is an expensive algorithm
  - → For each sample, we need to derive a p-value for each class
  - → Computation complexity of $O(nc)$ where n is number of samples and c is the number of classes

WHEN TO USE CONFORMAL PREDICTION?

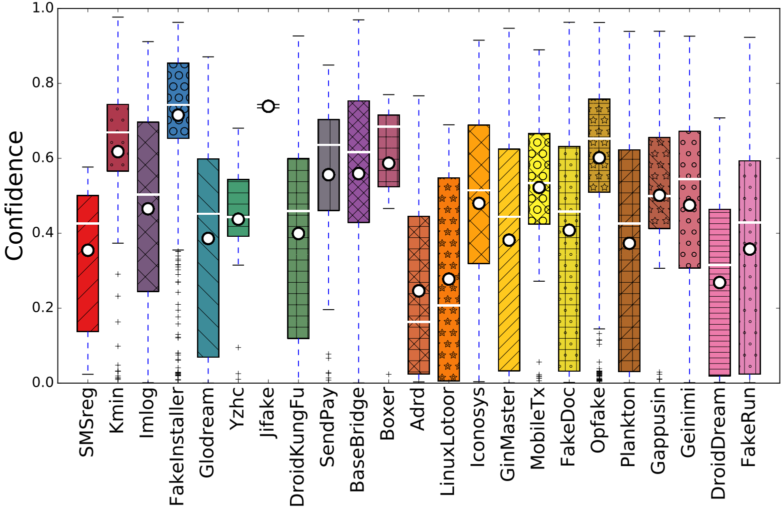
- CP is an expensive algorithm
  - → For each sample, we need to derive a p-value for each class
  - → Computation complexity of $O(nc)$ where n is number of samples and c is the number of classes

Conformal Evaluation[1]

- Provide statistical evaluation of the quality of a ML algorithm
  - → Quality threshold to understand when should be trusting SVM
  - → Statistical evidences of the choices of SVM
  - → Selectively invoke CP to alleviate runtime performance

[1] Jordaney, R., Wang Z., Papini D., Nouretdinov I., Cavallaro L. ``Misleading Metrics: On Evaluating Machine Learning for Malware with Confidence.'' TR 2016-1, Royal Holloway, University of London, 2016.

CONFIDENCE OF CORRECT SVM DECISIONS

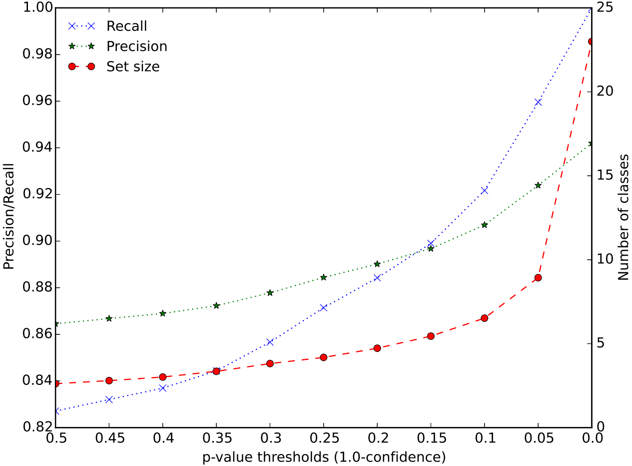


SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

ACCURACY VS. PREDICTION SET SIZE

RQ2.2 Can we deal with sparse behaviors?



- Accuracy improves with the prediction set size

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

COMING UP

- DroidScribe is a 1-gram model
  - → We extend it with larger context sizes and Markov chain construction
  - → We focus also on binary classification
- Preliminary experiments
  - → Marvin dataset (∼9,000 malicious and benign samples)
  - → **130** 1-grams (unique system calls), **5,162** 2-grams and **62,677** 3-grams
- Note: Markov chains from m N-grams, transition matrices of size m^2: huge feature space
- Possible way of addressing it:
  1. Feature selection
  2. PCA
  3. Composite behaviours
  4. Smoothing (WIP)
  5. Log bilinear model (WIP)

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

Feature Selection

Reduce the feature space by selecting the ``highest scoring" features

F-statistic: How significantly the feature contributes to variance in the output
P-value: The probability of achieving a better score by removing the feature

- Preliminary tests: top 1,000 features yields an F1-score of ∼0.965

SYSTEMS SECURITY RESEARCH LAB
ROYAL HOLLOWAY UNIVERSITY OF LONDON

Feature Selection

Reduce the feature space by selecting the ``highest scoring" features

F-statistic: How significantly the feature contributes to variance in the output
P-value: The probability of achieving a better score by removing the feature

- Preliminary tests: top 1,000 features yields an F1-score of $\sim$0.965

PCA Reduction

Tested combination of top selected features and #of principle components

- Optimal values: 1,000 top scoring features and 75 principal components
- Interestingly (perhaps unsurprising) performance still better with 1,000 top scoring features and **no PCA**

		Number of Principle Components																														
	0	10000	9000	8000	7000	6000	5000	4000	3000	2000	1000	900	800	700	600	500	400	300	200	100	75	50	25	10	9	8	7	6	5	4	3	2	1
	10000	0.858	0.862	0.861	0.860	0.864	0.872	0.880	0.892	0.902	0.913	0.914	0.917	0.920	0.917	0.918	0.919	0.920	0.921	0.919	0.915	0.913	0.904	0.895	0.882	0.882	0.870	0.860	0.836	0.817	0.801	0.721	0.600
	9000		0.859	0.863	0.860	0.863	0.866	0.878	0.892	0.909	0.916	0.918	0.919	0.919	0.926	0.921	0.921	0.924	0.921	0.918	0.916	0.910	0.904	0.893	0.883	0.881	0.874	0.866	0.844	0.825	0.810	0.741	0.601
	8000			0.852	0.856	0.856	0.867	0.883	0.892	0.904	0.914	0.918	0.916	0.918	0.921	0.923	0.922	0.922	0.925	0.922	0.918	0.917	0.907	0.891	0.889	0.882	0.878	0.865	0.842	0.827	0.807	0.748	0.593
	7000				0.850	0.850	0.866	0.880	0.890	0.896	0.912	0.914	0.916	0.917	0.921	0.925	0.924	0.929	0.926	0.926	0.925	0.920	0.908	0.892	0.886	0.882	0.877	0.877	0.868	0.859	0.806	0.751	0.598
	6000					0.858	0.864	0.876	0.885	0.899	0.910	0.913	0.914	0.916	0.918	0.919	0.922	0.924	0.922	0.924	0.925	0.920	0.912	0.893	0.891	0.888	0.883	0.877	0.874	0.864	0.812	0.764	0.594
	5000						0.875	0.882	0.892	0.904	0.912	0.915	0.916	0.920	0.923	0.924	0.927	0.927	0.928	0.928	0.929	0.923	0.908	0.892	0.893	0.889	0.888	0.880	0.878	0.866	0.812	0.769	0.597
	4000							0.889	0.897	0.906	0.915	0.915	0.917	0.921	0.918	0.922	0.927	0.929	0.931	0.929	0.927	0.922	0.914	0.898	0.896	0.895	0.891	0.888	0.883	0.871	0.821	0.788	0.605
	3000								0.891	0.902	0.914	0.912	0.915	0.919	0.918	0.922	0.923	0.926	0.930	0.930	0.927	0.923	0.912	0.900	0.900	0.898	0.893	0.883	0.882	0.871	0.821	0.778	0.604
	2000									0.904	0.915	0.917	0.917	0.920	0.920	0.921	0.924	0.927	0.930	0.933	0.933	0.928	0.919	0.894	0.892	0.889	0.891	0.887	0.880	0.873	0.825	0.793	0.607
	1000										0.916	0.915	0.920	0.919	0.919	0.920	0.923	0.924	0.928	0.930	0.933	0.931	0.927	0.910	0.908	0.902	0.902	0.899	0.888	0.858	0.846	0.806	0.601
	900											0.921	0.919	0.920	0.923	0.924	0.927	0.924	0.930	0.933	0.930	0.933	0.926	0.911	0.908	0.907	0.904	0.899	0.888	0.867	0.847	0.799	0.606
	800												0.920	0.920	0.921	0.923	0.924	0.923	0.925	0.927	0.927	0.927	0.926	0.906	0.901	0.902	0.897	0.893	0.884	0.861	0.845	0.803	0.606
	700													0.923	0.923	0.926	0.923	0.926	0.926	0.928	0.928	0.930	0.927	0.914	0.912	0.908	0.904	0.895	0.885	0.861	0.844	0.798	0.602
	600														0.920	0.923	0.921	0.922	0.925	0.924	0.924	0.925	0.923	0.913	0.911	0.908	0.908	0.900	0.890	0.865	0.840	0.801	0.604
K Best Selected Features Prior To PCA Reduction	500															0.923	0.924	0.924	0.925	0.924	0.925	0.928	0.923	0.916	0.916	0.913	0.903	0.902	0.887	0.874	0.842	0.796	0.596
	400																0.921	0.923	0.923	0.924	0.924	0.925	0.923	0.913	0.915	0.913	0.903	0.897	0.881	0.859	0.823	0.785	0.598
	300																	0.921	0.919	0.920	0.921	0.920	0.918	0.912	0.909	0.906	0.892	0.879	0.871	0.848	0.814	0.767	0.614
	200																		0.914	0.919	0.916	0.918	0.919	0.911	0.908	0.908	0.909	0.902	0.856	0.849	0.806	0.748	0.623
	100																			0.911	0.910	0.911	0.903	0.900	0.901	0.898	0.897	0.896	0.889	0.878	0.867	0.817	0.630
	75																				0.906	0.904	0.900	0.895	0.896	0.890	0.889	0.886	0.886	0.882	0.858	0.785	0.638
	50																					0.900	0.894	0.893	0.897	0.894	0.893	0.890	0.893	0.883	0.879	0.841	0.655
	25																						0.892	0.879	0.881	0.880	0.877	0.875	0.862	0.857	0.860	0.838	0.701
	10																							0.852	0.853	0.851	0.853	0.852	0.851	0.853	0.852	0.852	0.832
	9																								0.833	0.833	0.833	0.833	0.833	0.833	0.832	0.833	0.832
	8																									0.829	0.829	0.829	0.829	0.829	0.829	0.829	0.829
	7																										0.830	0.829	0.830	0.829	0.829	0.829	0.829
	6																											0.830	0.830	0.830	0.829	0.830	0.830
	5																												0.830	0.830	0.830	0.830	0.830
	4																													0.829	0.829	0.829	0.829
	3																														0.825	0.826	0.825
	2																															0.800	0.800
	1																																0.799

Figure: Heatmap of F1 scores after PCA reduction applied.

Preliminary Experiments

Early tests conducted on ∼9,000 samples with a rough 1:1 ratio

- N-gram frequencies as features across traces with only syscalls and syscalls with Binder reconstruction
- Random forests classifiers (128 estimators)
- Hold out validation 64-33% averaged over 10 runs

Preliminary Experiments

Early tests conducted on ∼9,000 samples with a rough 1:1 ratio

- N-gram frequencies as features across traces with only syscalls and syscalls with Binder reconstruction
- Random forests classifiers (128 estimators)
- Hold out validation 64-33% averaged over 10 runs

1. F1, precision and recall scores of around 0.95-0.96 with 1,000 top features
2. Scores over 0.9 with 25 features and over 0.8 with only 1 feature
3. Syscalls + Binder produces slightly higher scores and with fewer features
4. Binder methods also prominent in feature importance
5. While 2-grams outperforms 1-grams, 3-grams shows no improvement (effect of junk syscalls?).

COMING UP (CONT.)

	Granularity		SYSCALLS			SYS_BINDER	
	Ngram size	1	2	3	1	2	3
	Classifier	RF	RF	RF	RF	RF	RF
K-best features	10000			0.961		0.963	0.963
	9000			0.960		0.962	0.962
	8000			0.960		0.964	0.963
	7000			0.960		0.964	0.962
	6000			0.960		0.965	0.962
	5000			0.959		0.964	0.964
	4000		0.960	0.959		0.965	0.963
	3000		0.960	0.959		0.965	0.962
	2000		0.961	0.958		0.965	0.963
	1000		0.959	0.953		0.965	0.960
	750		0.958	0.952		0.965	0.960
	500		0.955	0.949		0.963	0.958
	250		0.951	0.939	0.953	0.959	0.953
	100	0.940	0.933	0.932	0.947	0.953	0.945
	75	0.934	0.929	0.927	0.947	0.951	0.938
	50	0.931	0.918	0.922	0.940	0.942	0.925
	25	0.924	0.904	0.908	0.927	0.923	0.911
	10	0.894	0.842	0.882	0.844	0.853	0.876
	5	0.847	0.819	0.861	0.758	0.832	0.857
	1	0.716	0.810	0.810	0.714	0.805	0.805

Figure: Heatmap of F1-scores from preliminary tests.

COMING UP (CONT.): DATASETS CONUNDRUM

- Datasets should be ``history-aware"[6]
- 1:1 benign to malicious ratio easy to benchmark and compare against others
- Real settings likely with **unbalanced** datasets
  - → Skewing towards one class or another is possible
  - → Generally performance degrades on an imbalanced dataset (precision sensitive)
  - → AUC ROC ill-suited; better AUC precision-recall
    - → FPR is the same, but the proportion of false positives to true positives is much worse with a higher number negative objects
    - → ROC curve stable given class imbalance so conceals the degradation in precision
    - → The precision-recall curve does change visually and reveals these hidden effects[7]

---

[6] Are Your Training Datasets Yet Relevant? - An Investigation into the Importance of Timeline in Machine Learning-Based Malware Detection Allix, Kevin; Bissyande, Tegawende Francois D Assise; Klein, Jacques; Le Traon, Yves in ESSoS 2015

[7] ``The Relationship Between Precision-Recall and ROC Curves.'' Jesse Davis and Mark Goadrich. ICML, 2006

RQ3—Concept Drift

Statistical Evaluation of ML Classifiers

Usually, a 2-phase process:

1. Training: build a model M, given labeled objects
2. Testing: given M, predict the labels of unknown objects

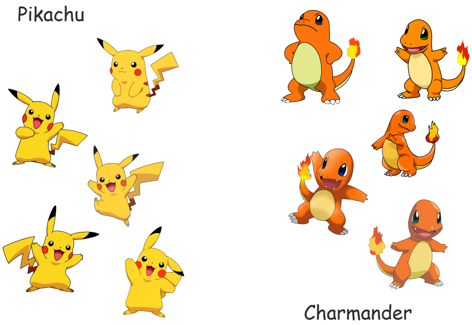
Objects are described as vectors of features

MACHINE LEARNING CLASSIFICATION

Usually, a 2-phase process:

1. Training: build a model M, given labeled objects
2. Testing: given M, predict the labels of unknown objects

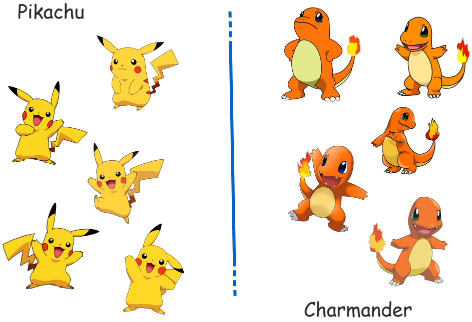
Objects are described as vectors of features



Usually, a 2-phase process:

1. Training: build a model M, given labeled objects
2. Testing: given M, predict the labels of unknown objects

Objects are described as vectors of features



MACHINE LEARNING CLASSIFICATION PROBLEM: CONCEPT DRIFT

- Concept drift is the **change in the statistical properties** of an object in unforeseen ways
- Drifted objects will likely be wrongly classified

Hitmonlee
(new pokémon family)

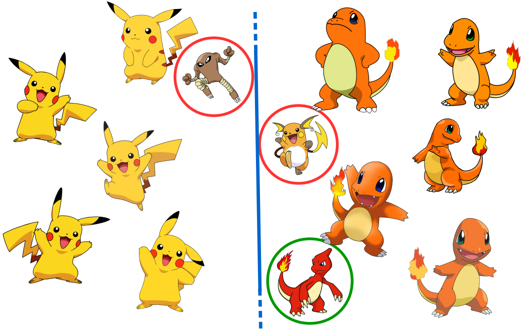
Raichu
(evolution of Pikachu)

Charmeleon
(evolution of Charmander)

SYSTEMS SECURITY RESEARCH LAB

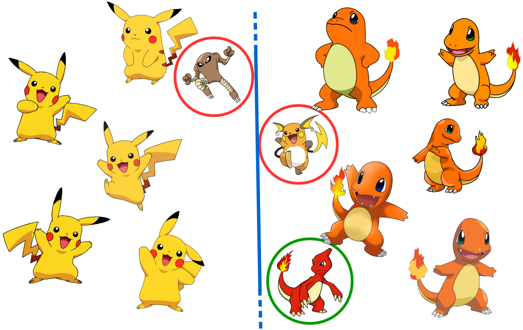
ROYAL HOLLOWAY UNIVERSITY OF LONDON

- Concept drift is the **change in the statistical properties** of an object in unforeseen ways
- Drifted objects will likely be wrongly classified



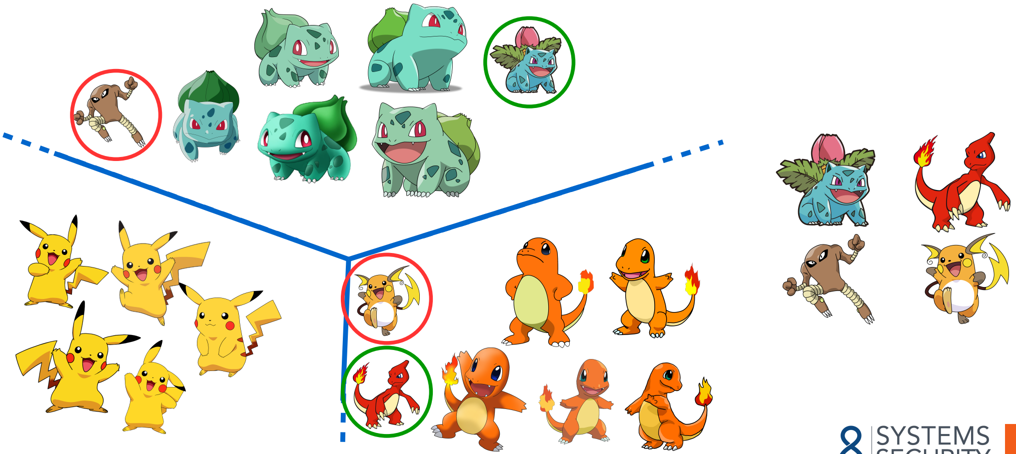
MACHINE LEARNING CLASSIFICATION PROBLEM: CONCEPT DRIFT

- Concept drift is the **change in the statistical properties** of an object in unforeseen ways
- Drifted objects will likely be wrongly classified



Of course, the problem exists in multiclass classification settings...

- Multiclass classification is a generalization of the binary case



CONCEPT DRIFT

- In non-stationary contexts classifiers will suffer from concept drift due to:
  - → malware evolution 
  - → new malware families 
- Need a way to **assess the predictions** of classifiers
  - → Ideally classifier-agnostic assessments
- Need to identify objects that fit a model and those drifting away

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

CONCEPT DRIFT

- In non-stationary contexts classifiers will suffer from concept drift due to:
  - → malware evolution 
  - → new malware families 
- Need a way to **assess the predictions** of classifiers
  - → Ideally classifier-agnostic assessments
- Need to identify objects that fit a model and those drifting away

Our Contributions

- Conformal Evaluator: statistical evaluation of ML classifiers
- Per-class quality threshold to identify reliable and unreliable predictions

CONFORMAL EVALUATOR

- Assesses decisions made by a classifier
  - $\rightarrow$ Mark each decision as reliable or unreliable
- Builds and makes use of p-value as assessment criteria
- Computes per-class thresholds to divide reliable decisions from unreliable ones

CONFORMAL EVALUATOR: P-VALUE?

- Used to measure ``how well'' a sample fits into a single class
- Conformal Evaluator computes a p-value for each class, for each test element

Definition

$$
\begin{aligned}
\alpha_t &= \text{Non-conformity score for test element t} \\
\forall i \in \mathscr{K}, \alpha_i &= \text{Non-conformity score for train element i} \\
\text{p-value} &= \frac{|\{i : \alpha_i \geq \alpha_t\}|}{|\mathscr{K}|} \\
\mathscr{K} &= \text{Total number of element}
\end{aligned}
$$

P-value

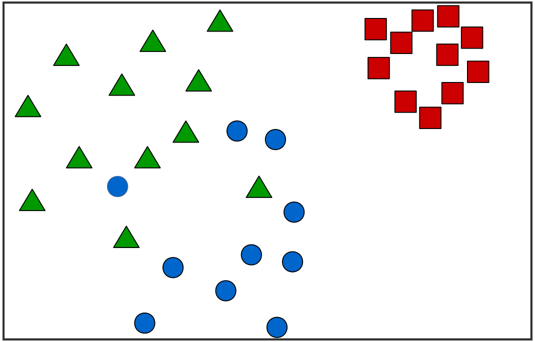
Ratio between the number of training elements that are more dissimilar than the element under test

SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

CONFORMAL EVALUATOR: P-VALUE EXAMPLE

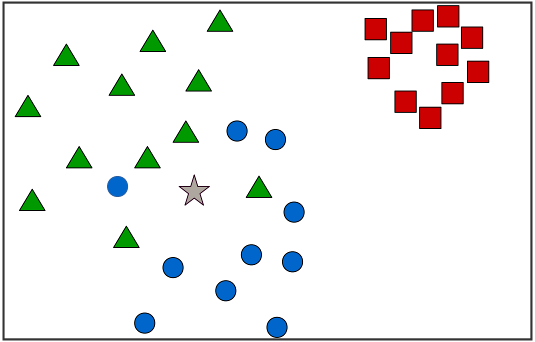
ML classifier:
distance from centroid



1. Setting: 3-class classification

CONFORMAL EVALUATOR: P-VALUE EXAMPLE

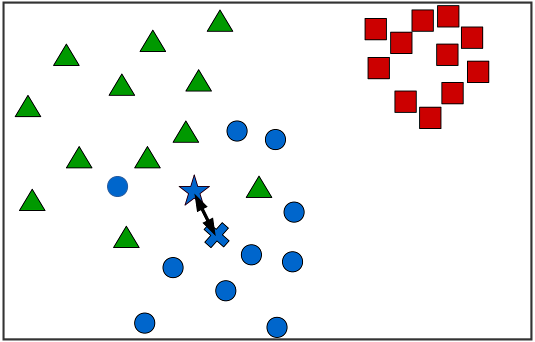
ML classifier:
distance from centroid



1. Setting: 3-class classification
2. Test object

CONFORMAL EVALUATOR: P-VALUE EXAMPLE

ML classifier:
distance from centroid

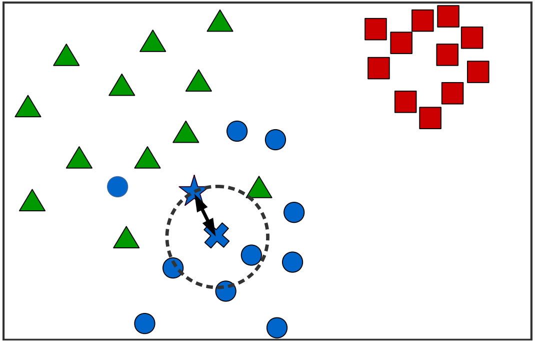


1. Setting: 3-class classification
2. Test object
   3.1 Compute distance to blue class

SYSTEMS SECURITY RESEARCH LAB
ROYAL HOLLOWAY UNIVERSITY OF LONDON

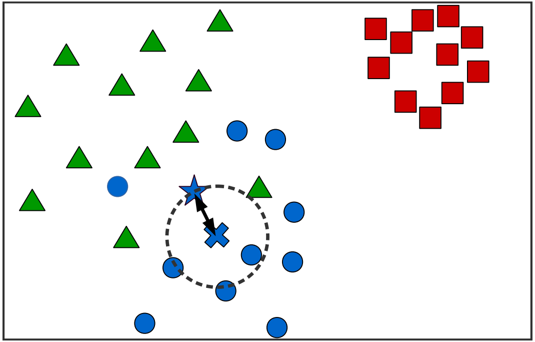
CONFORMAL EVALUATOR: P-VALUE EXAMPLE

ML classifier:
distance from centroid



1. Setting: 3-class classification
2. Test object
   3.1 Compute distance to blue class
   3.2 How many objects are more dissimilar than the one under test?

ML classifier:
distance from centroid

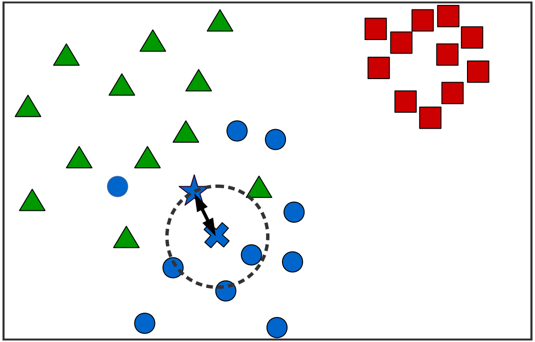


1. Setting: 3-class classification
2. Test object
   3.1 Compute distance to blue class
   3.2 How many objects are more dissimilar than the one under test?
   3.3 9

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

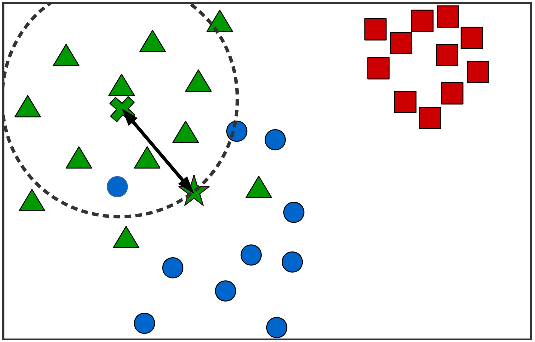
ML classifier:
distance from centroid



1. Setting: 3-class classification
2. Test object
   3.1 Compute distance to blue class
   3.2 How many objects are more dissimilar than the one under test?
   3.3 9
   3.4 P-value$_{\star} = \frac{9}{10}$

CONFORMAL EVALUATOR: P-VALUE EXAMPLE

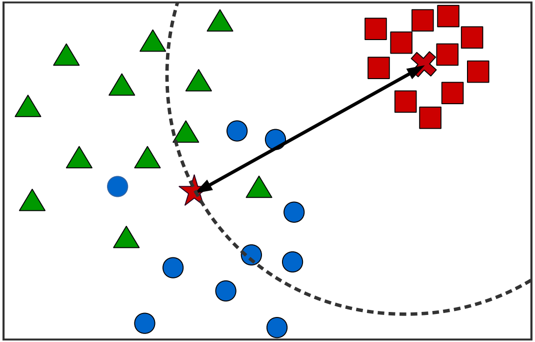
Machine learning classifier:
distance from centroid



1. Initial situation: three classes
2. Test object
   4.1 Calculate distance to green class
   4.2 How many objects are more dissimilar than the one under test?
   4.3 4
   4.4 P-value$_{\star} = \frac{4}{12}$

CONFORMAL EVALUATOR: P-VALUE EXAMPLE

Machine learning classifier:
distance from centroid



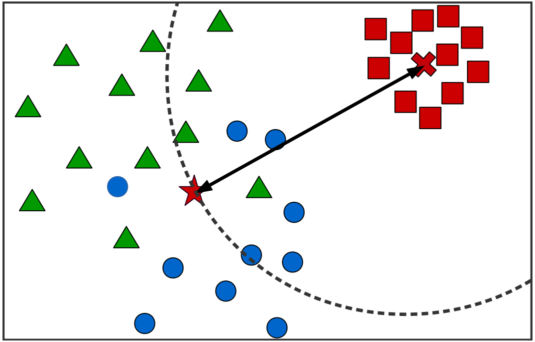
1. Initial situation: three classes
2. Test object
   - 5.1 Calculate distance to red class
   - 5.2 How many objects are more dissimilar than the one under test?
   - 5.3 0
   - 5.4 P-value$_{\star} = \frac{0}{11}$

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

CONFORMAL EVALUATOR: P-VALUE EXAMPLE

Machine learning classifier:
distance from centroid

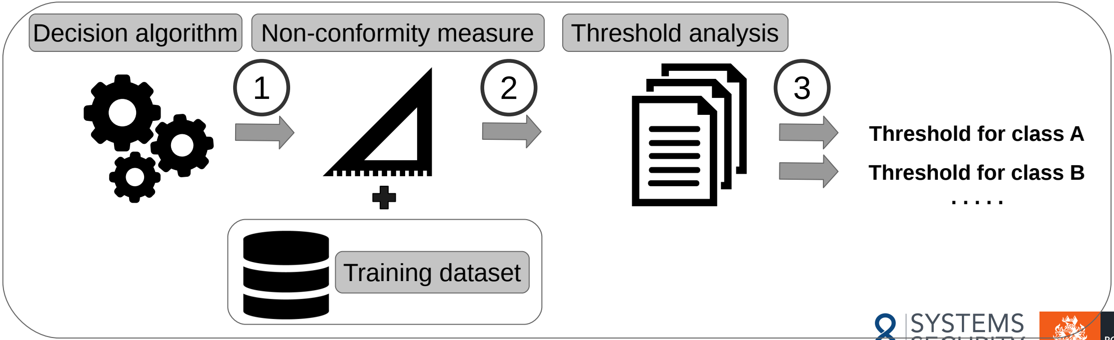


1. Initial situation: three classes
2. Test object
   - 5.1 Calculate distance to red class
   - 5.2 How many objects are more dissimilar than the one under test?
   - 5.3 0
   - 5.4 P-value$_{\star} = \frac{0}{11}$

Let's see how p-values are used within Conformal Evaluator

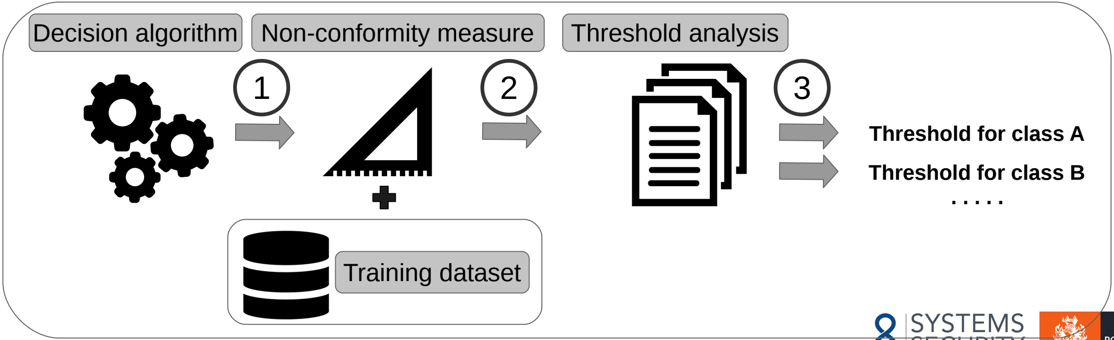
CONFORMAL EVALUATOR: HOW DOES IT WORK?

1. Extracts the non-conformity measure (NCM) from the decision making algorithm
   - → NCM provides non-conformity scores for p-value computations
   - → Example: distance from hyperplane, Random Forest probability (adapted to satisfy the non-conformity requirement)



CONFORMAL EVALUATOR: HOW DOES IT WORK?

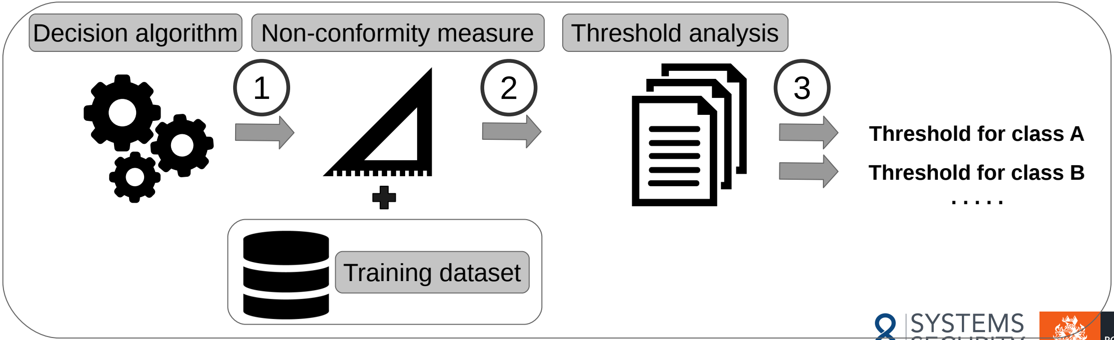
1. Extracts the non-conformity measure (NCM) from the decision making algorithm
2. Builds p-values for all training samples in a cross-validation fashion



SYSTEMS SECURITY RESEARCH LAB
ROYAL HOLLOWAY UNIVERSITY OF LONDON

CONFORMAL EVALUATOR: HOW DOES IT WORK?

1. Extracts the non-conformity measure (NCM) from the decision making algorithm
2. Builds p-values for all training samples in a cross-validation fashion
3. Computes per-class threshold to divide reliable predictions from unreliable ones



SYSTEMS SECURITY RESEARCH LAB
ROYAL HOLLOWAY UNIVERSITY OF LONDON

Customizable constraints:

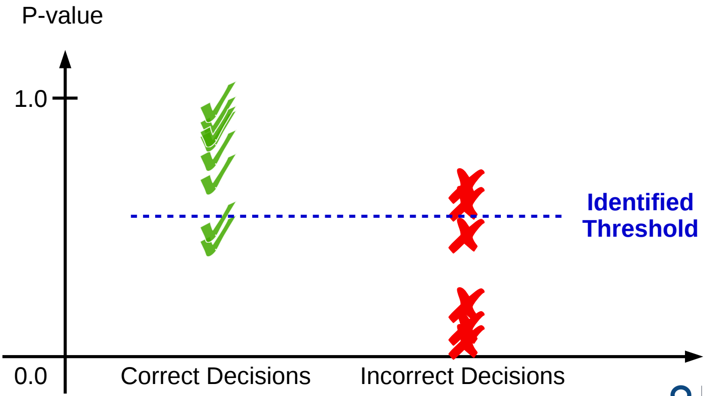
- Desired performance (of the predictions marked as reliable)
  → E.g.: high-level performance will raise the threshold
- Number of unreliable prediction tolerated
  → E.g.: low number of unreliable prediction will lower the threshold

Assumptions & Hypothesis

- Performance of non-drifted elements are similar to the one declared by the algorithm
- Predictions with high confidence will have higher p-values

CONFORMAL EVALUATOR: IDENTIFYING PER-CLASS THRESHOLDS

- We use the p-values and prediction labels from training samples
- From the thresholds that satisfy the constraints we chose the one that maximize one or the other



EXPERIMENTAL RESULTS: CASE STUDIES

- Binary case study: Android malware detection algorithm
  - → Reimplemented Drebin[8] algorithm with similar results
    (0.95-0.92 precision-recall on malicious apps and 0.99-0.99 precision-recall on benign apps)
  - → Static features of Android apps, linear SVM (used as NCM)
  - → Concept drift scenario: malware evolution
- Multiclass case study: Microsoft malware classification algorithm
  - → Solution to Microsoft Kaggle competition[9], ranked among the top ones
  - → Static features from Windows PE binaries, Random Forest (used as NCM)
  - → Concept drift scenario: family discovery

[8] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, and Konrad Rieck. Drebin: Effective and Explainable Detection of Android Malware in Your Pocket. In 21st Annual Network and Distributed System Security Symposium (NDSS), San Diego, California, USA, February 23-26, 2014.

[9] KAGGLE INC. Microsoft Malware Classification Challenge (BIG 2015). https://www.kaggle.com/c/malware-classification, 2015.

EXPERIMENTAL RESULTS: CASE STUDIES

- Binary case study: Android malware detection algorithm
  - → Reimplemented Drebin[8] algorithm with similar results (0.95-0.92 precision-recall on malicious apps and 0.99-0.99 precision-recall on benign apps)
  - → Static features of Android apps, linear SVM (used as NCM)
  - → Concept drift scenario: malware evolution

[8] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, and Konrad Rieck. Drebin: Effective and Explainable Detection of Android Malware in Your Pocket. In 21st Annual Network and Distributed System Security Symposium (NDSS), San Diego, California, USA, February 23-26, 2014.

[9] KAGGLE INC. Microsoft Malware Classification Challenge (BIG 2015). https://www.kaggle.com/c/malware-classification, 2015.

EXPERIMENTAL RESULTS: BINARY CLASSIFICATION (MALWARE EVOLUTION)

- Drebin dataset: samples collected from 2010 to 2012
- Marvin dataset[10]: malware apps collected from 2010 to 2014 (no duplicates)
  - → We expect some object to drift from objects in the Drebin dataset

Drebin Dataset

Type	Samples
Benign	123,435
Malware	5,560

Marvin Dataset

Type	Samples
Benign	9,592
Malware	9,179

[10] Martina Lindorfer, Matthias Neugschwandtner, and Christian Platzer. MARVIN: Efficient and Comprehensive Mobile App Classification through Static And Dynamic Analysis. In 39th IEEE Annual Computer Software and Applications Conference (COMPSAC), Taichung, Taiwan, July 1-5, 2015

EXPERIMENTAL RESULTS: BINARY CLASSIFICATION (MALWARE EVOLUTION)

Experiment: Drift Confirmation

- Training dataset: Drebin dataset
- Testing dataset: 4,500 benign and 4,500 malicious random samples from Marvin dataset

	Prediction label		
Original label	Benign	Malicious	Recall
Benign	4,498	2	1
Malicious	2,890	1,610	0.36
Precision	0.61	1	

SYSTEMS SECURITY RESEARCH LAB

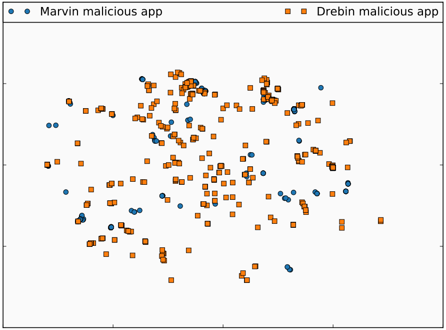
ROYAL HOLLOWAY UNIVERSITY OF LONDON

EXPERIMENTAL RESULTS: BINARY CLASSIFICATION (MALWARE EVOLUTION)

Experiment: Drift Confirmation

- Training dataset: Drebin dataset
- Testing dataset: 4,500 benign and 4,500 malicious random samples from Marvin dataset

	Prediction label		
Original label	Benign	Malicious	Recall
Benign	4,498	2	1
Malicious	2,890	1,610	0.36
Precision	0.61	1	



SYSTEMS SECURITY RESEARCH LAB

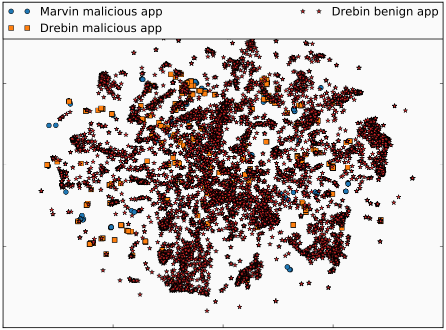
ROYAL HOLLOWAY UNIVERSITY OF LONDON

Experiment: Drift Confirmation

- Training dataset: Drebin dataset
- Testing dataset: 4,500 benign and 4,500 malicious random samples from Marvin dataset

Prediction label

Original label	Benign	Malicious	Recall
Benign	4,498	2	1
Malicious	2,890	1,610	0.36
Precision	0.61	1	



SYSTEMS SECURITY RESEARCH LAB
ROYAL HOLLOWAY UNIVERSITY OF LONDON

EXPERIMENTAL RESULTS: BINARY CLASSIFICATION (MALWARE EVOLUTION)

Experiment: Threshold Identification

- Training dataset: Drebin dataset
- Testing dataset: 4,500 benign and 4,500 malicious random samples from Marvin dataset
- Make use of Conformal Evaluator's prediction assessment algorithm
  - → Constraints: F1-score of 0.99 and 0.76 of elements marked as reliable

	Prediction label		
Original label	Benign	Malicious	Recall
Benign	4,257	2	1
Malicious	504	1,610	0.76
Precision	0.89	1	

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

EXPERIMENTAL RESULTS: BINARY CLASSIFICATION (MALWARE EVOLUTION)

Experiment: Retraining

- Training dataset: Drebin dataset + samples marked as unreliable from previous experiment
- Testing dataset: 4,500 benign and 4,500 malicious random samples of Marvin dataset (no sample overlap from previous experiment)

	Assigned label		
Sample	Benign	Malicious	Recall
Benign	4,413	87	0.98
Malicious	255	4,245	0.94
Precision	0.96	0.98	

Experiment: Threshold Comparison

- Compare probability- and p-value-based thresholds
  - → Central tendency and dispersion points of true positive distribution
- Training dataset: Drebin dataset
- Testing dataset: 4,500 benign and 4,500 malicious apps from Marvin dataset (random sampling)

	TPR (reliable predictions)		TPR (unreliable predictions)		FPR (reliable predictions)		FPR (unreliable predictions)	
	p-value	probability	p-value	probability	p-value	probability	p-value	probability
1st quartile	0.9045	0.6654	0.0000	0.3176	0.0007	0.0	0.0000	0.0013
Median	0.8737	0.8061	0.3080	0.3300	0.0000	0.0	0.0008	0.0008
Mean	0.8737	0.4352	0.3080	0.3433	0.0000	0.0	0.0008	0.0018
3rd quartile	0.8723	0.6327	0.3411	0.3548	0.0000	0.0	0.0005	0.0005

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

EXPERIMENTAL RESULTS: BINARY CLASSIFICATION (MALWARE EVOLUTION)

Experiment: Threshold Comparison

- Compare probability- and p-value-based thresholds
  → Central tendency and dispersion points of true positive distribution
- Training dataset: Drebin dataset
- Testing dataset: 4,500 benign and 4,500 malicious apps from Marvin dataset (random sampling)

	TPR (reliable predictions)		TPR (unreliable predictions)		FPR (reliable predictions)		FPR (unreliable predictions)	
	p-value	probability	p-value	probability	p-value	probability	p-value	probability
1st quartile	0.9045	0.6654	0.0000	0.3176	0.0007	0.0	0.0000	0.0013
Median	0.8737	0.8061	0.3080	0.3300	0.0000	0.0	0.0008	0.0008
Mean	0.8737	0.4352	0.3080	0.3433	0.0000	0.0	0.0008	0.0018
3rd quartile	0.8723	0.6327	0.3411	0.3548	0.0000	0.0	0.0005	0.0005

SYSTEMS SECURITY RESEARCH LAB
ROYAL HOLLOWAY UNIVERSITY OF LONDON

EXPERIMENTAL RESULTS: BINARY CLASSIFICATION (MALWARE EVOLUTION)

Experiment: Threshold Comparison

- Compare probability- and p-value-based thresholds
  → Central tendency and dispersion points of true positive distribution
- Training dataset: Drebin dataset
- Testing dataset: 4,500 benign and 4,500 malicious apps from Marvin dataset (random sampling)

	TPR (reliable predictions)		TPR (unreliable predictions)		FPR (reliable predictions)		FPR (unreliable predictions)	
	p-value	probability	p-value	probability	p-value	probability	p-value	probability
1st quartile	0.9045	0.6654	0.0000	0.3176	0.0007	0.0	0.0000	0.0013
Median	0.8737	0.8061	0.3080	0.3300	0.0000	0.0	0.0008	0.0008
Mean	0.8737	0.4352	0.3080	0.3433	0.0000	0.0	0.0008	0.0018
3rd quartile	0.8723	0.6327	0.3411	0.3548	0.0000	0.0	0.0005	0.0005

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

TAKE AWAYS

Conformal Evaluator (CE)[1]

Statistical evaluation to assess predictions of ML classifiers and identify concept drift

[1] (Transcend: Detecting Concept Drift in Malware Classification Models. USENIX Sec 2017)

TAKE AWAYS

Conformal Evaluator (CE)[1]

Statistical evaluation to assess predictions of ML classifiers and identify concept drift

[1] (Transcend: Detecting Concept Drift in Malware Classification Models. USENIX Sec 2017)

Algorithm Agnostic: Uses non-conformity measure (NCM) from the ML classifier
Statistical Support: Builds p-values from NCM to statistically-support predictions
Quality Thresholds: Builds thresholds from p-values to identify unreliable predictions

TAKE AWAYS

Conformal Evaluator (CE)[1]

Statistical evaluation to assess predictions of ML classifiers and identify concept drift

[1] (Transcend: Detecting Concept Drift in Malware Classification Models. USENIX Sec 2017)

Algorithm Agnostic: Uses non-conformity measure (NCM) from the ML classifier
Statistical Support: Builds p-values from NCM to statistically-support predictions
Quality Thresholds: Builds thresholds from p-values to identify unreliable predictions

- We evaluate the proposed solution on different ML classifiers and case studies
  - → Android malware apps in binary classification settings
  - → Windows PE binaries in multi-class classification settings
- Information on CE's python code and dataset availability at:

https://s2lab.isg.rhul.ac.uk/projects/ce

SYSTEMS SECURITY RESEARCH LAB
ROYAL HOLLOWAY UNIVERSITY OF LONDON

- CopperDroid: automatic reconstruction of apps behaviors[11]
  - → System calls to abstract OS- and Android-specific behaviors
  - → Resilient to changes to the runtime and Android versions

[11] http://s2lab.isg.rhul.ac.uk/papers/files/ndss2015.pdf
[12] http://s2lab.isg.rhul.ac.uk/papers/files/most2016.pdf
[13] http://s2lab.isg.rhul.ac.uk/papers/files/aisec2016.pdf and http://s2lab.isg.rhul.ac.uk/papers/files/usenixsec2017.pdf

CONCLUSION

- CopperDroid: automatic reconstruction of apps behaviors[11]
  - → System calls to abstract OS- and Android-specific behaviors
  - → Resilient to changes to the runtime and Android versions
- Classification with such semantics: "It... Could... Work!"[12]
  - → Selective set-based classification (CE/CP)
  - → (WIP: binary classification and different feature engineering)

[11] http://s2lab.isg.rhul.ac.uk/papers/files/ndss2015.pdf
[12] http://s2lab.isg.rhul.ac.uk/papers/files/most2016.pdf
[13] http://s2lab.isg.rhul.ac.uk/papers/files/aisec2016.pdf and http://s2lab.isg.rhul.ac.uk/papers/files/usenixsec2017.pdf

CONCLUSION

- CopperDroid: automatic reconstruction of apps behaviors[11]
  - → System calls to abstract OS- and Android-specific behaviors
  - → Resilient to changes to the runtime and Android versions
- Classification with such semantics: "It... Could... Work!"[12]
  - → Selective set-based classification (CE/CP)
  - → (WIP: binary classification and different feature engineering)
- Statistical evaluation of ML seems promising[13]
  - → Identify concept drift and and when to trust a prediction
    - → TPR from **37.5%** to **92.7%** in realistic settings
    - → Identifies previously-unknown classes or malicious samples

[11] http://s2lab.isg.rhul.ac.uk/papers/files/ndss2015.pdf
[12] http://s2lab.isg.rhul.ac.uk/papers/files/most2016.pdf
[13] http://s2lab.isg.rhul.ac.uk/papers/files/aisec2016.pdf and http://s2lab.isg.rhul.ac.uk/papers/files/usenixsec2017.pdf

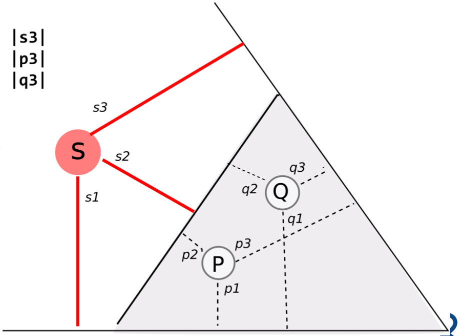
CP: OVERVIEW AND EXAMPLE

- P-value is the probability of truth for the hypothesis that a sample belongs to a class

```
ncm_S = |s1| + |s2| + |s3|
ncm_P = |p1| + |p2| + |p3|
ncm_Q = |q1| + |q2| + |q3|

ncm_S > ncm_P > ncm_Q

pval_S = 0/3
pval_P = 1/3
pval_Q = 2/3
```



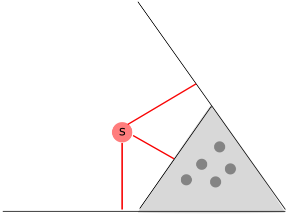
- Nonconformity Measure (NCM) is a geometric measure of how well a sample is far from a class.
  - → For SVM, the NCM $\mathscr{N}_D^z$ of a sample z w.r.t. class D is sum distances from all hyperplanes bounding the class D.

$$\mathscr{N}_D^z = \sum_i d(z, \mathscr{H}_i)$$

- P-value is a statistical measure of how well a sample fits in a class.
  - → P-value $\mathscr{P}_D^z$ represents the proportion of samples in D that more different than z w.r.t. D.

$$\mathscr{P}_D^z = \frac{|\{j = 1, \ldots, n : \mathscr{N}_D^j \geq \mathscr{N}_D^z\}|}{n}$$

- Standard classification algorithms calculate probability of a sample belonging to a class
- For the case of SVM, this is based on Euclidean distance (Platt's scaling)



Using Probabilites

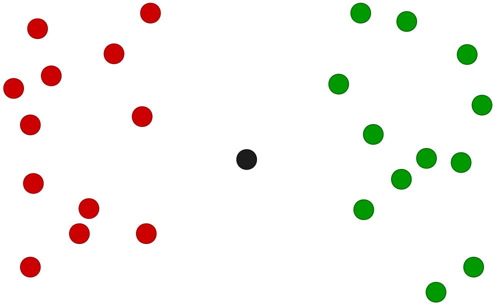
- Platt's scaling is based on logistic regression
- Logistic regression is sensitive to outliers which introduces inaccuracies
- Probabilities to sum up to one which introduces skewing

BINARY CLASSIFICATION CASE STUDY: COMPARISON WITH PROBABILITY

	TPR of kept elements		FPR of kept elements		TPR of discarded elements		FPR of discarded elements		MALICIOUS kept elements		BENIGN kept elements	
	p-value	probability	p-value	probability	p-value	probability	p-value	probability	p-value	probability	p-value	probability
1st quartile	0.9045	0.6654	0.0007	0.0	0.0000	0.3176	0.0000	0.0013	0.3956	0.1156	0.6480	0.6673
Median	0.8737	0.8061	0.0000	0.0	0.3080	0.3300	0.0008	0.0008	0.0880	0.0584	0.4136	0.4304
Mean	0.8737	0.4352	0.0000	0.0	0.3080	0.3433	0.0008	0.0018	0.0880	0.1578	0.4136	0.7513
3rd quartile	0.8723	0.6327	0.0000	0.0	0.3411	0.3548	0.0005	0.0005	0.0313	0.0109	0.1573	0.1629

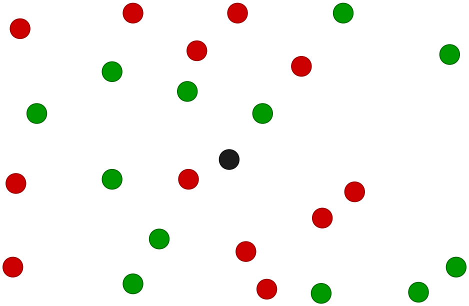
Table 4: Thresholds comparison between p-value and probability. The results show, together with the performance of the sample marked as unreliable, a clear advantage of the p-value metric compared to the probability one.

P-VALUE VS PROBABILITY: SITUATION 1



	P-value	Probability
Red	0.0	0.5
Green	0.0	0.5

P-VALUE VS PROBABILITY: SITUATION 2



	P-value	Probability
Red	0.5	0.5
Green	0.5	0.5

MULTICLASS CLASSIFICATION (NEW FAMILY DISCOVERY)

- Dataset: Microsoft Malware Classification Challenge (2015)

Microsoft Malware Classification Challenge Dataset

Malware	Samples	Malware	Samples
Ramnit	1 541	Obfuscator.ACY	1 228
Lollipop	2 478	Gatak	1 013
Kelihos_ver3	2 942	Kelihos_ver1	398
Vundo	475	Tracur	751

SYSTEMS SECURITY RESEARCH LAB

ROYAL HOLLOWAY UNIVERSITY OF LONDON

MULTICLASS CLASSIFICATION (NEW FAMILY DISCOVERY)

Experiment: Family Discovery

- Training families: Ramnit, Lollipop, Kelihos_ver3, Vundo, Obfuscator.ACY, Gatak, Kelihos_ver1
- Testing family: Tracur

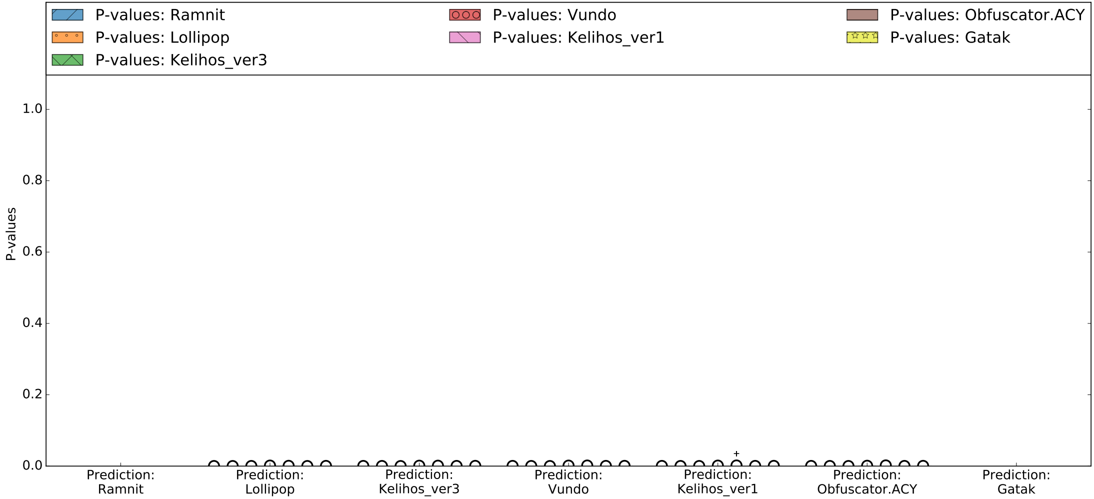
Classification results:

Lollipop	Kelihos_ver3	Vundo	Kelihos_ver1	Obfuscator.ACY
5	6	358	140	242

SYSTEMS SECURITY RESEARCH LAB
ROYAL HOLLOWAY UNIVERSITY OF LONDON

MULTICLASS CLASSIFICATION (NEW FAMILY DISCOVERY)

P-value distribution for samples of Tracur family; as expected, the values are all close to zero.



MULTICLASS CLASSIFICATION (NEW FAMILY DISCOVERY)

Probability distribution for samples of Tracur family; bounded to sum to one, the values are different than zero.