PlatPal: Detecting Malicious Documents with Platform Diversity

Meng Xu and Taesoo Kim

Georgia Institute of Technology

Malicious Documents On the Rise

APT28 Targets Hospitality Sector, Presents Threat to Travelers

August 11, 2017 | by Lindsay Smith, Ben Read | Threat Research

FireEye has moderate confidence that a campaign targeting the hospitality sector is attributed to Russian actor APT28. We believe this activity, which dates back to at least July 2017, was intended to target travelers to hotels throughout Europe and the Middle East. The actor has used several notable techniques in these incidents such as sniffing passwords from Wi-Fi traffic, poisoning the NetBIOS Name Service, and spreading laterally via the EternalBlue exploit.

APT28 Uses Malicious Document to Target Hospitality Industry

Microsoft PowerPoint exploit used to bypass antivirus and spread malware

It's the first time this exploit has been used to target PowerPoint users - and it's being used to distribute powerful Trojan malware, say researchers.

By Danny Palmer | August 14, 2017 -- 16:49 GMT (17:49 BST) | Topic: Security

Adobe Components Exploited

137 CVEs in 2015

227 CVEs in 2016

Element parser

JavaScript engine

Font manager

System dependencies

Maldoc Formula

Flexibility of doc spec

+

A large attack surface

+

Less caution from users

=

More opportunities to profit

Battle against Maldoc - A Survey

Category	Focus	Work	Year	Detection
Static				
Dynamic				

Battle against Maldoc - A Survey

Category	Focus	Work	Year	Detection
Static	JavaScript	PJScan	2011	Lexical analysis
	JavaScript	Vatamanu et al.	2012	Token clustering
	JavaScript	Lux0r	2014	API reference classification
	JavaScript	MPScan	2013	Shellcode and opcode sig
Dynamic				

Battle against Maldoc - A Survey

Category	Focus	Work	Year	Detection
Static	JavaScript	PJScan	2011	Lexical analysis
	JavaScript	Vatamanu et al.	2012	Token clustering
	JavaScript	Lux0r	2014	API reference classification
	JavaScript	MPScan	2013	Shellcode and opcode sig
	Metadata	PDF Malware Slayer	2012	Linearized object path
	Metadata	Srndic et al.	2013	Hierarchical structure
	Metadata	PDFrate	2012	Content meta-features
	Both	Maiorca et al.	2016	Many heuristics combined
Dynamic				

Battle against Maldoc - A Survey

Category	Focus	Work	Year	Detection
Static	JavaScript	PJScan	2011	Lexical analysis
	JavaScript	Vatamanu et al.	2012	Token clustering
	JavaScript	Lux0r	2014	API reference classification
	JavaScript	MPScan	2013	Shellcode and opcode sig
	Metadata	PDF Malware Slayer	2012	Linearized object path
	Metadata	Srndic et al.	2013	Hierarchical structure
	Metadata	PDFrate	2012	Content meta-features
	Both	Maiorca et al.	2016	Many heuristics combined
Dynamic	JavaScript	MDScan	2011	Shellcode and opcode sig
	JavaScript	PDF Scrutinizer	2012	Known attack patterns
	JavaScript	ShellOS	2011	Memory access patterns
	JavaScript	Liu et al.	2014	Common attack behaviors

Battle against Maldoc - A Survey

Category	Focus	Work	Year	Detection
Static	JavaScript	PJScan	2011	Lexical analysis
	JavaScript	Vatamanu et al.	2012	Token clustering
	JavaScript	Lux0r	2014	API reference classification
	JavaScript	MPScan	2013	Shellcode and opcode sig
	Metadata	PDF Malware Slayer	2012	Linearized object path
	Metadata	Srndic et al.	2013	Hierarchical structure
	Metadata	PDFrate	2012	Content meta-features
	Both	Maiorca et al.	2016	Many heuristics combined
Dynamic	JavaScript	MDScan	2011	Shellcode and opcode sig
	JavaScript	PDF Scrutinizer	2012	Known attack patterns
	JavaScript	ShellOS	2011	Memory access patterns
	JavaScript	Liu et al.	2014	Common attack behaviors
	Memory	CWXDetector	2012	Violation of invariants

Reliance on External PDF Parser

Category	Focus	Work	Year	Detection	External Parser ?
Static	JavaScript	PJScan	2011	Lexical analysis	Yes
	JavaScript	Vatamanu et al.	2012	Token clustering	Yes
	JavaScript	Lux0r	2014	API reference classification	Yes
	JavaScript	MPScan	2013	Shellcode and opcode sig	No
	Metadata	PDF Malware Slayer	2012	Linearized object path	Yes
	Metadata	Srndic et al.	2013	Hierarchical structure	Yes
	Metadata	PDFrate	2012	Content meta-features	Yes
	Both	Maiorca et al.	2016	Many heuristics combined	Yes
Dynamic	JavaScript	MDScan	2011	Shellcode and opcode sig	Yes
	JavaScript	PDF Scrutinizer	2012	Known attack patterns	Yes
	JavaScript	ShellOS	2011	Memory access patterns	Yes
	JavaScript	Liu et al.	2014	Common attack behaviors	No
	Memory	CWXDetector	2012	Violation of invariants	No

Reliance on External PDF Parser

Category	Focus	Work	Year	Detection	External Parser ?
Static	JavaScript	PJScan	2011	Lexical analysis	Yes
	JavaScript	Vatamanu et al.	2012	Token clustering	Yes
	JavaScript	Lux0r	2014	API reference classification	Yes
	JavaScript	MPScan	2013	Shellcode and opcode sig	No
	Metadata	PDF Malware Slayer	2012	Linearized object path	Yes
	Metadata	Srndic et al.	2013	Hierarchical structure	Yes
	Metadata	PDFrate	2012	Content features	Yes
	Both	Maiorca et al.	2016	Many heuristics combined	Yes
Dynamic	JavaScript	MDScan	2011	Shellcode and opcode sig	Yes
	JavaScript	PDF Scrutinizer	2012	Known attack patterns	Yes
	JavaScript	ShellOS	2011	Memory access patterns	Yes
	JavaScript	Liu et al.	2014	Common attack behaviors	No
	Memory	CWXDetector	2012	Violation of invariants	No

Parser-confusion attacks

(Carmony et al., NDSS'16)

Reliance on Machine Learning

Category	Focus	Work	Year	Detection	Machine Learning ?
Static	JavaScript	PJScan	2011	Lexical analysis	Yes
	JavaScript	Vatamanu et al.	2012	Token clustering	Yes
	JavaScript	Lux0r	2014	API reference classification	Yes
	JavaScript	MPScan	2013	Shellcode and opcode sig	No
	Metadata	PDF Malware Slayer	2012	Linearized object path	Yes
	Metadata	Srndic et al.	2013	Hierarchical structure	Yes
	Metadata	PDFrate	2012	Content meta-features	Yes
	Both	Maiorca et al.	2016	Many heuristics combined	Yes
Dynamic	JavaScript	MDScan	2011	Shellcode and opcode sig	No
	JavaScript	PDF Scrutinizer	2012	Known attack patterns	No
	JavaScript	ShellOS	2011	Memory access patterns	No
	JavaScript	Liu et al.	2014	Common attack behaviors	No
	Memory	CWXDetector	2012	Violation of invariants	No

Reliance on Machine Learning

Category	Focus	Work	Year	Detection	Machine Learning ?
Static	JavaScript	PJScan	2011	Lexical analysis	Yes
	JavaScript	Vatamanu et al.	2012	Token clustering	Yes
	JavaScript	Lux0r	2014	API reference classification	Yes
	JavaScript	MPScan	2013	Shellcode and opcode sig	No
	Metadata	PDF Malware Slayer	2012	Linearized object path	Yes
	Metadata	Srndic et al.	2013	Object structural path	Yes
	Metadata	PDFrate	2012	Structural meta-features	Yes
	Both	Maiorca et al.	2016	Many heuristics combined	Yes
Dynamic	JavaScript	MDScan	2011	Shellcode and opcode sig	No
	JavaScript	PDF Scrutinizer	2012	Known attack patterns	No
	JavaScript	ShellOS	2011	Memory access patterns	No
	JavaScript	Liu et al.	2014	Common attack behaviors	No
	Memory	CWXDetector	2012	Violation of invariants	No

Automatic classifier evasions

(Xu et al., NDSS'16)

Reliance on Known Attacks

Category	Focus	Work	Year	Detection	Known Attacks ?
Static	JavaScript	PJScan	2011	Lexical analysis	Yes
	JavaScript	Vatamanu et al.	2012	Token clustering	Yes
	JavaScript	Lux0r	2014	API reference classification	Yes
	JavaScript	MPScan	2013	Shellcode and opcode sig	Yes
	Metadata	PDF Malware Slayer	2012	Linearized object path	Yes
	Metadata	Srndic et al.	2013	Hierarchical structure	Yes
	Metadata	PDFrate	2012	Content meta-features	Yes
	Both	Maiorca et al.	2016	Many heuristics combined	Yes
Dynamic	JavaScript	MDScan	2011	Shellcode and opcode sig	Yes
	JavaScript	PDF Scrutinizer	2012	Known attack patterns	Yes
	JavaScript	ShellOS	2011	Memory access patterns	Yes
	JavaScript	Liu et al.	2014	Common attack behaviors	Yes
	Memory	CWXDetector	2012	Violation of invariants	No

Reliance on Known Attacks

Category	Focus	Work	Year	Detection	Known Attacks ?
Static	JavaScript	PJScan	2011	Lexical analysis	Yes
	JavaScript	Vatamanu et al.	2012	Token clustering	Yes
	JavaScript	Lux0r	2014	API reference classification	Yes
	JavaScript	MPScan	2013	Shellcode and opcode sig	Yes
	Metadata	PDF Malware Slayer	2012	Linearized object path	Yes
	Metadata	Srndic et al.	2013	Hierarchical structure	Yes
	Metadata	PDFrate	2012	Content meta-features	Yes
	Both	Maiorca et al.	2016	Many heuristics combined	Yes
Dynamic	JavaScript	MDScan	2011	Shellcode and opcode sig	Yes
	JavaScript	PDF Scrutinizer	2012	Known attack patterns	Yes
	JavaScript	ShellOS	2011	Memory access patterns	Yes
	JavaScript	Liu et al.	2014	Common attack behaviors	Yes
	Memory	CWXDetector	2012	Violation of invariants	No

How about zero-day attacks ?

Reliance on Detectable Discrepancy

(between benign and malicious docs)

Category	Focus	Work	Year	Detection	Discrepancy ?
Static	JavaScript	PJScan	2011	Lexical analysis	Yes
	JavaScript	Vatamanu et al.	2012	Token clustering	Yes
	JavaScript	Lux0r	2014	API reference classification	Yes
	JavaScript	MPScan	2013	Shellcode and opcode sig	No
	Metadata	PDF Malware Slayer	2012	Linearized object path	Yes
	Metadata	Srndic et al.	2013	Hierarchical structure	Yes
	Metadata	PDFrate	2012	Content meta-features	Yes
	Both	Maiorca et al.	2016	Many heuristics combined	Yes
Dynamic	JavaScript	MDScan	2011	Shellcode and opcode sig	No
	JavaScript	PDF Scrutinizer	2012	Known attack patterns	No
	JavaScript	ShellOS	2011	Memory access patterns	Yes
	JavaScript	Liu et al.	2014	Common attack behaviors	Yes
	Memory	CWXDetector	2012	Violation of invariants	No

Reliance on Detectable Discrepancy

(between benign and malicious docs)

Category	Focus	Work	Year	Detection	Discrepancy ?
Static	JavaScript	PJScan	2011	Lexical analysis	Yes
	JavaScript	Vatamanu et al.	2012	Token clustering	Yes
	JavaScript	Lux0r	2014	API reference classification	Yes
	JavaScript	MPScan	2013	Shellcode and opcode sig	No
	Metadata	PDF Malware Slayer	2012	Linearized object path	Yes
	Metadata	PDFRate	2012	Hierarchical structure	Yes
	Metadata	PDF Classifier	2012	Hierarchical structure	Yes
	Both	Maiorca et al.	2016	Many heuristics combined	Yes
Dynamic	JavaScript	MDScan	2011	Shellcode and opcode sig	No
	JavaScript	PDF Scrutinizer	2012	Known attack patterns	No
	JavaScript	ShellOS	2011	Memory access patterns	Yes
	JavaScript	Liu et al.	2014	Common attack behaviors	Yes
	Memory	CWXDetector	2012	Violation of invariants	No

Mimicry and reverse mimicry attacks

(Srndic et al., Oakland'14 and Maiorca et al, AsiaCCS'13)

Highlights of the Survey

Prior works rely on

- External PDF parsers ⟶ *Parser-confusion attacks*
- Machine learning ⟶ *Automatic classifier evasion*
- Known attack signatures ⟶ *Zero-day attacks*
- Detectable discrepancy ⟶ *Mimicry and reverse mimicry*

Motivations for PlatPal

Prior works rely on

- External PDF parsers
- Machine learning
- Known attack signatures
- Detectable discrepancy

What PlatPal aims to achieve

Motivations for PlatPal

Prior works rely on

- ~~External PDF parsers~~
- Machine learning
- Known attack signatures
- Detectable discrepancy

What PlatPal aims to achieve

- Using Adobe's parser

Motivations for PlatPal

Prior works rely on

- ~~External PDF parsers~~
- ~~Machine learning~~
- Known attack signatures
- Detectable discrepancy

What PlatPal aims to achieve

- Using Adobe's parser
- Using only simple heuristics

Motivations for PlatPal

Prior works rely on

- ~~External PDF parsers~~
- ~~Machine learning~~
- ~~Known attack signatures~~
- Detectable discrepancy

What PlatPal aims to achieve

- Using Adobe's parser
- Using only simple heuristics
- Capable to detect zero-days

Motivations for PlatPal

Prior works rely on

- ~~External PDF parsers~~
- ~~Machine learning~~
- ~~Known attack signatures~~
- ~~Detectable discrepancy~~

What PlatPal aims to achieve

- Using Adobe's parser
- Using only simple heuristics
- Capable to detect zero-days
- Do not assume discrepancy

Motivations for PlatPal

Prior works rely on

- ~~External PDF parsers~~
- ~~Machine learning~~
- ~~Known attack signatures~~
- ~~Detectable discrepancy~~

What PlatPal aims to achieve

- Using Adobe's parser
- Using only simple heuristics
- Capable to detect zero-days
- Do not assume discrepancy
- Complementary to prior works

A Motivating Example

- A CVE-2013-2729 PoC against Adobe Reader 10.1.4

SHA-1: 74543610d9908698cb0b4bfcc73fc007bfeb6d84


Adobe Reader X
PDF
sample-cve-2013-2729-10.1.4


Macintosh HD
PDF
sample-
cve-201...0.1.4.pdf

Platform Diversity as A Heuristic

When the same document is opened across different platforms:

- A benign document “behaves” the same
- A malicious document “behaves” differently

Similar Ideas

- Two variants placed in disjoint memory partitions [*N-Variant Systems*]

- Two variants with stacks growing in different directions [*Orchestra*]

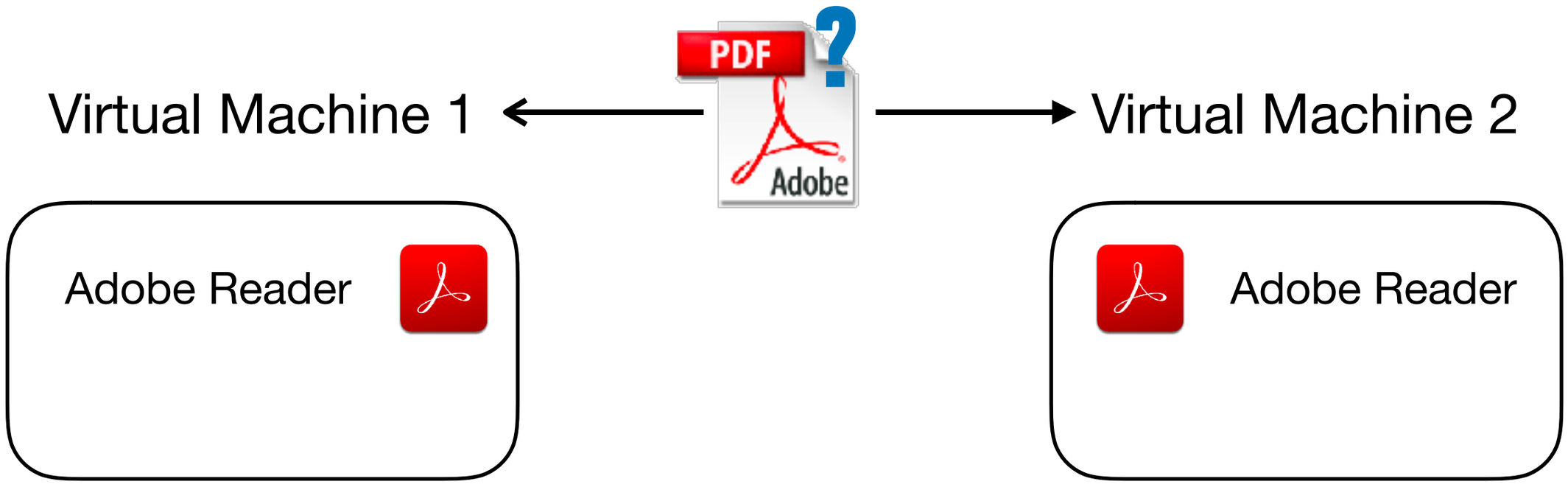
- Multiple variants with randomized heap object locations [*DieHard*]

- Multiple versions of the same program [*Varan, Mx*]

Questions for PlatPal

- What is a “behavior” ?
- What is a divergence ?
- How to trace them ?
- How to compare them ?

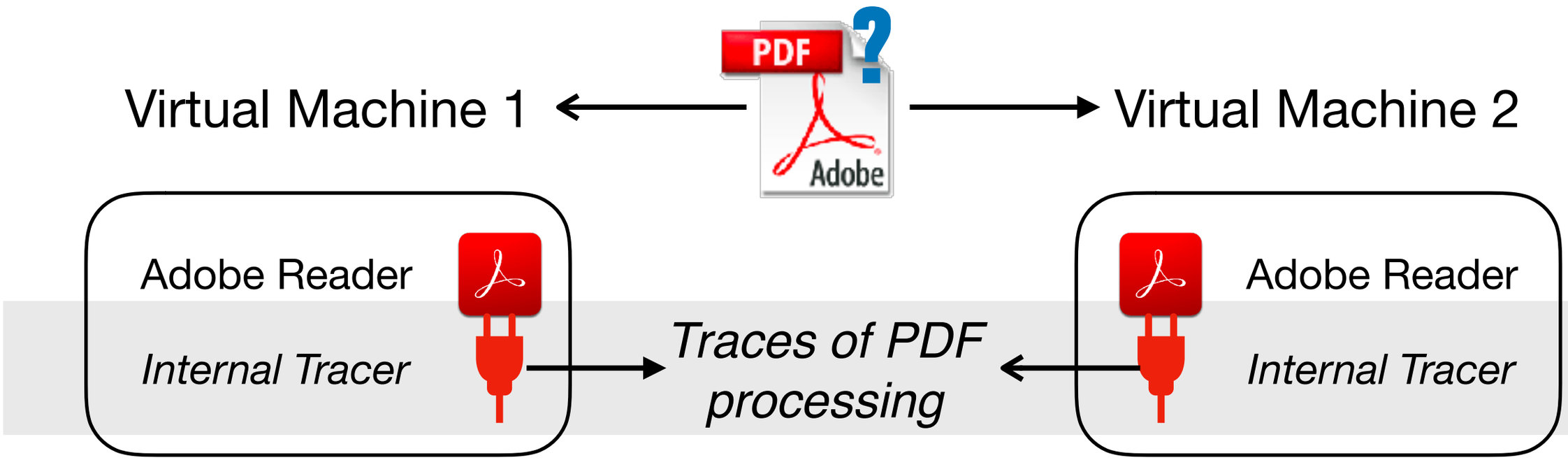
PlatPal Basic Setup



Windows Host

MacOS Host

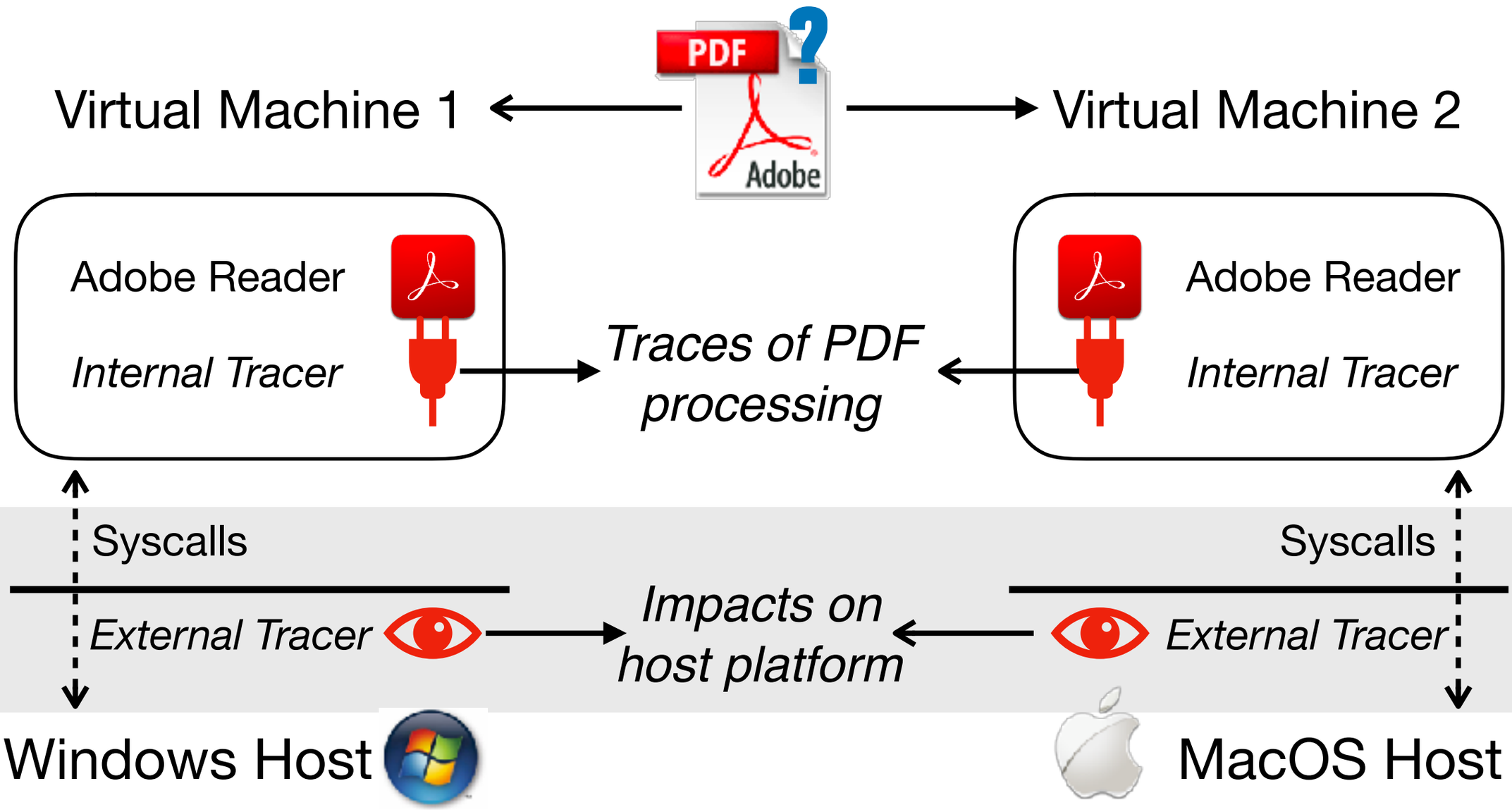
PlatPal Dual-Level Tracing



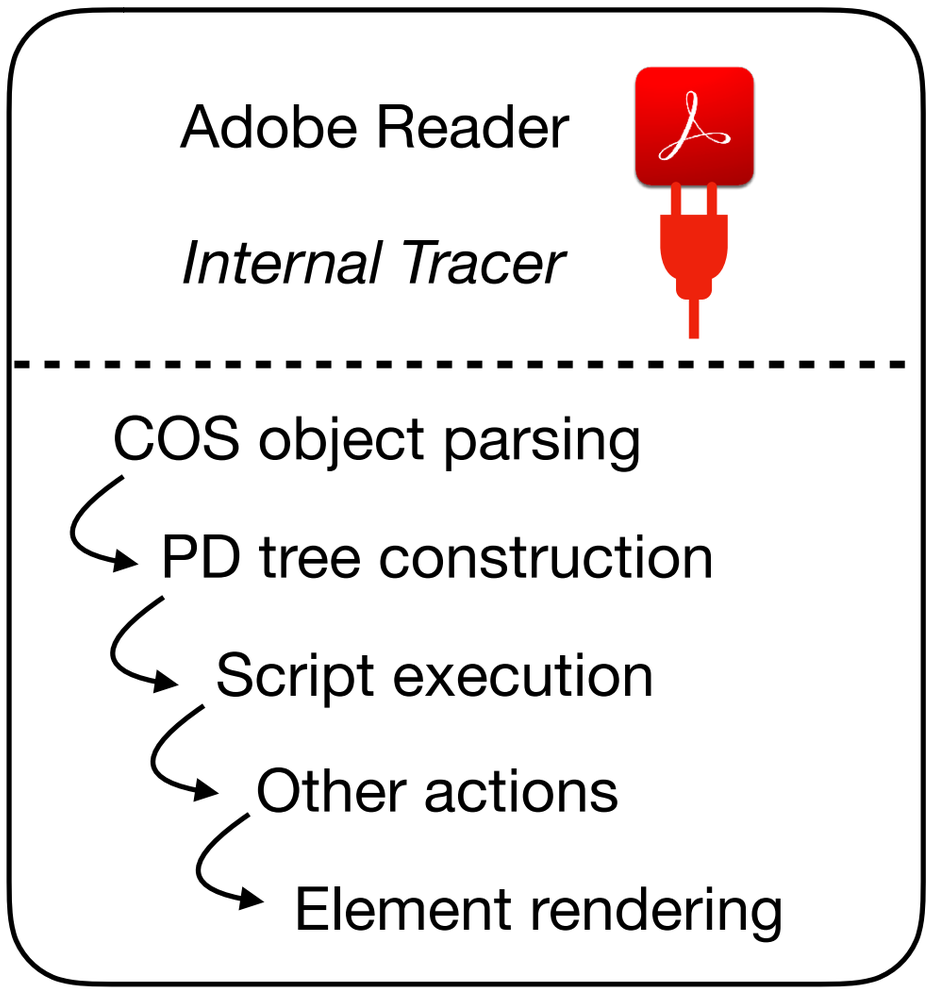
Windows Host

MacOS Host

PlatPal Dual-Level Tracing

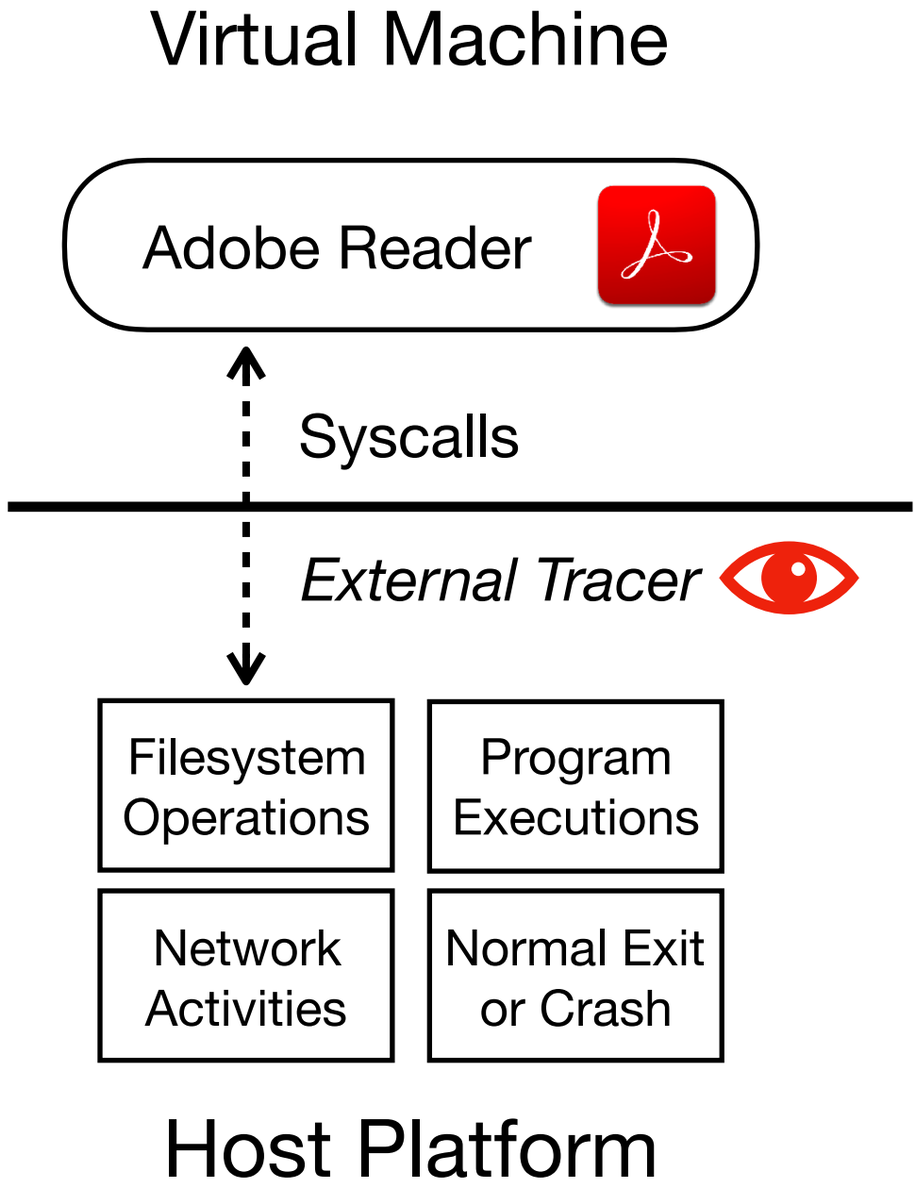


PlatPal Internal Tracer



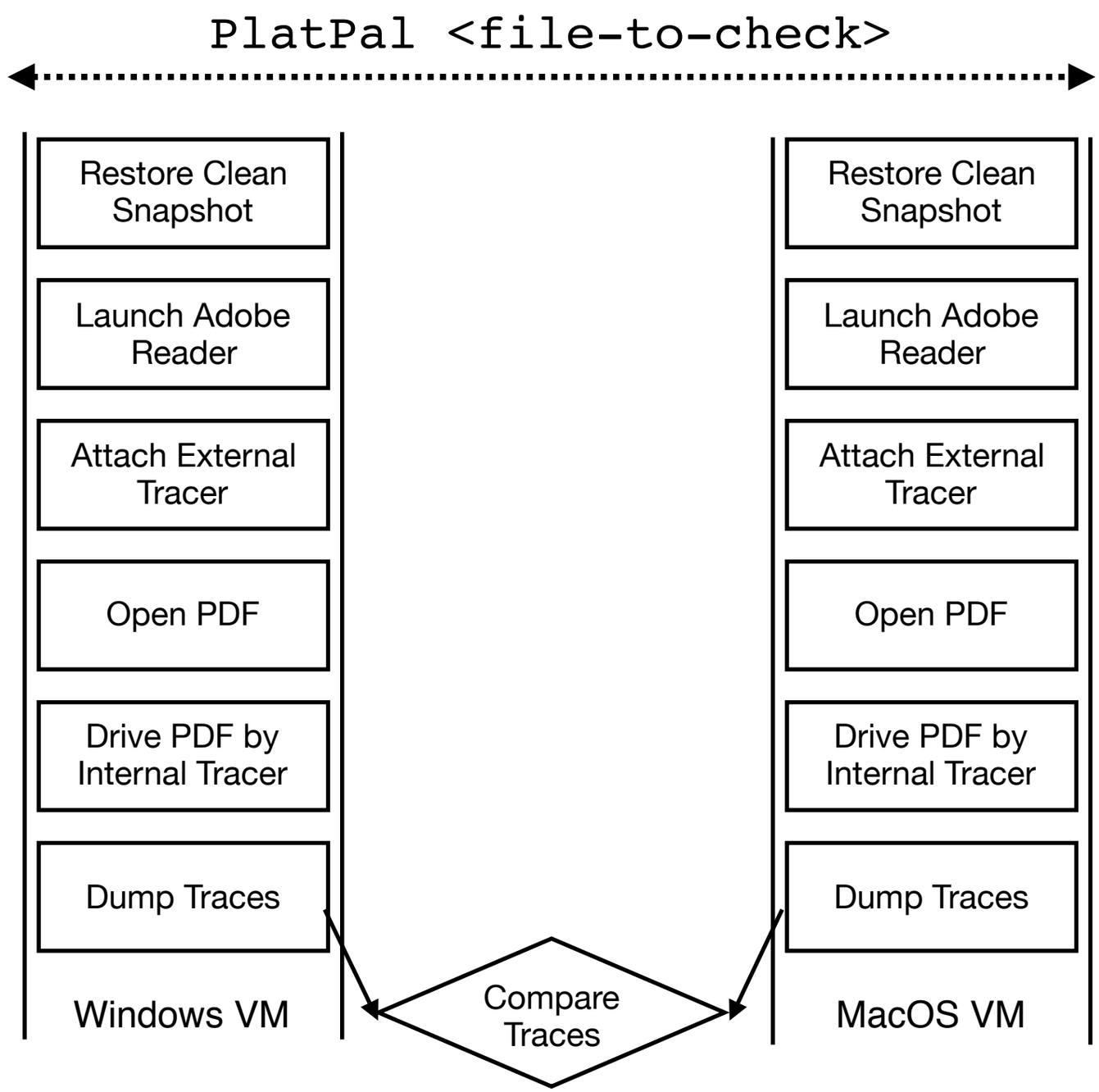
- Implemented as an Adobe Reader plugin.
- Hooks critical functions and callbacks during the PDF processing lifecycle.
- Very fast and stable across Adobe Reader versions.

PlatPal External Tracer



- Implemented based on *NtTrace* (for Windows) and *Dtrace* (for MacOS).
- Resembles high-level system impacts in the same manner as Cuckoo guest agent.
- Starts tracing only after the document is loaded into Adobe Reader.

PlatPal Automated Workflow



Evaluate PlatPal

- Robustness against benign samples
  - A benign document “behaves” the same ?
- Effectiveness against malicious samples
  - A malicious document “behaves” differently ?
- Speed and resource usages

Robustness

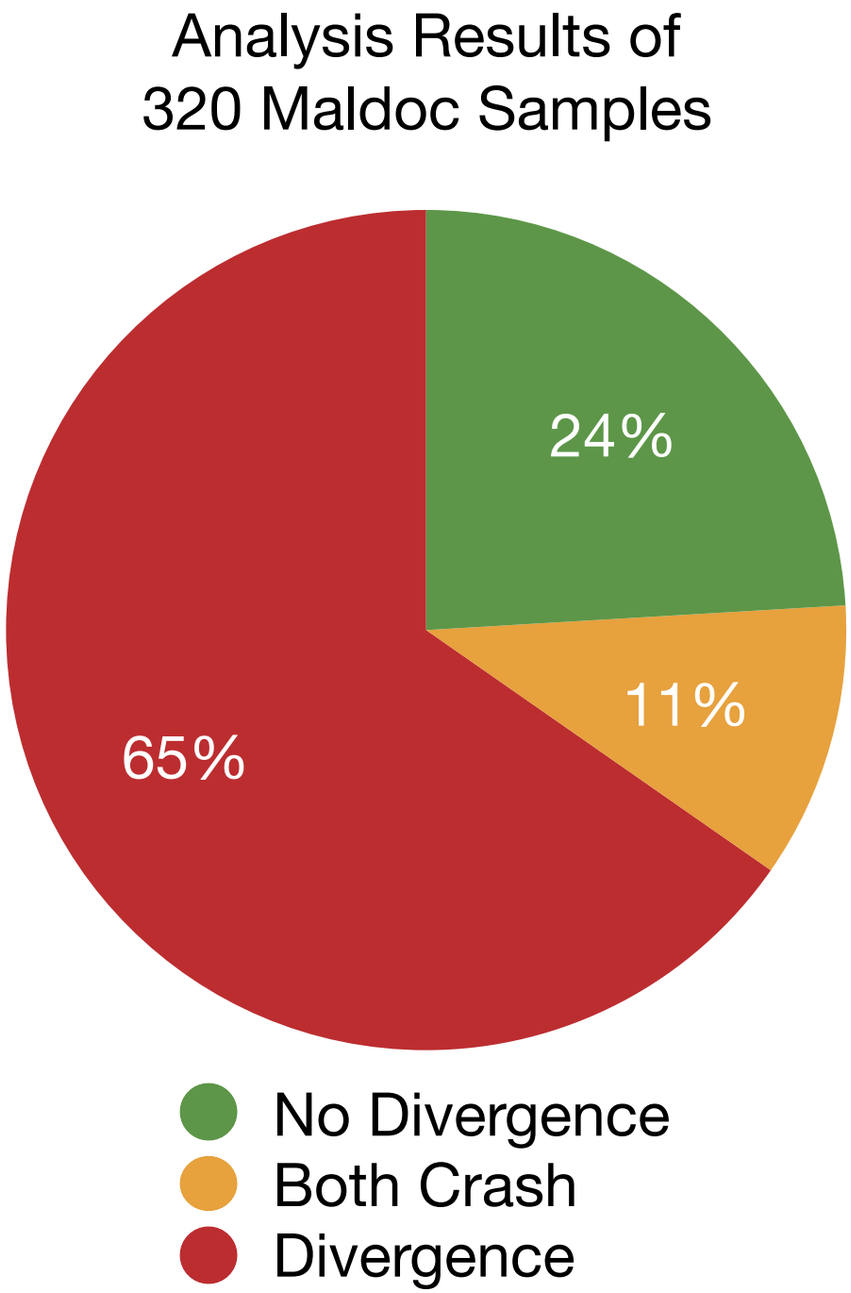
- 1000 samples from Google search.
- 30 samples that use advanced features in PDF standards from PDF learning sites.

Sample Type	Number of Samples	Divergence Detected ? (i.e., False Positive)
Plain PDF	966	No
Embedded fonts	34	No
JavaScript code	32	No
AcroForm	17	No
3D objects	2	No

Effectiveness

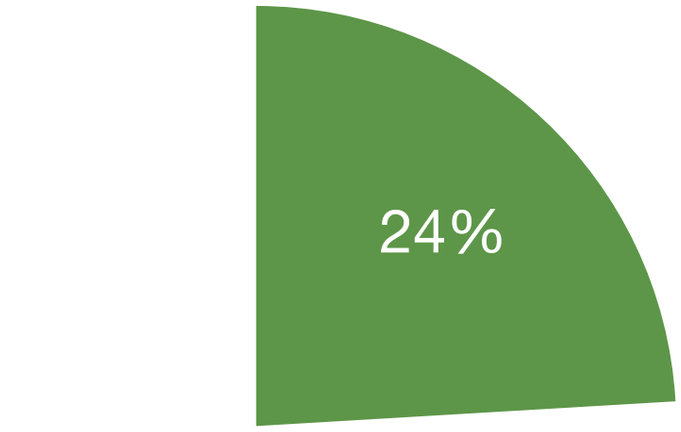
- 320 malicious samples from VirusTotal with CVE labels.
- Restricted to analyze CVEs published after 2013.
- Use the most recent version of Adobe Reader when the CVE is published.

Effectiveness



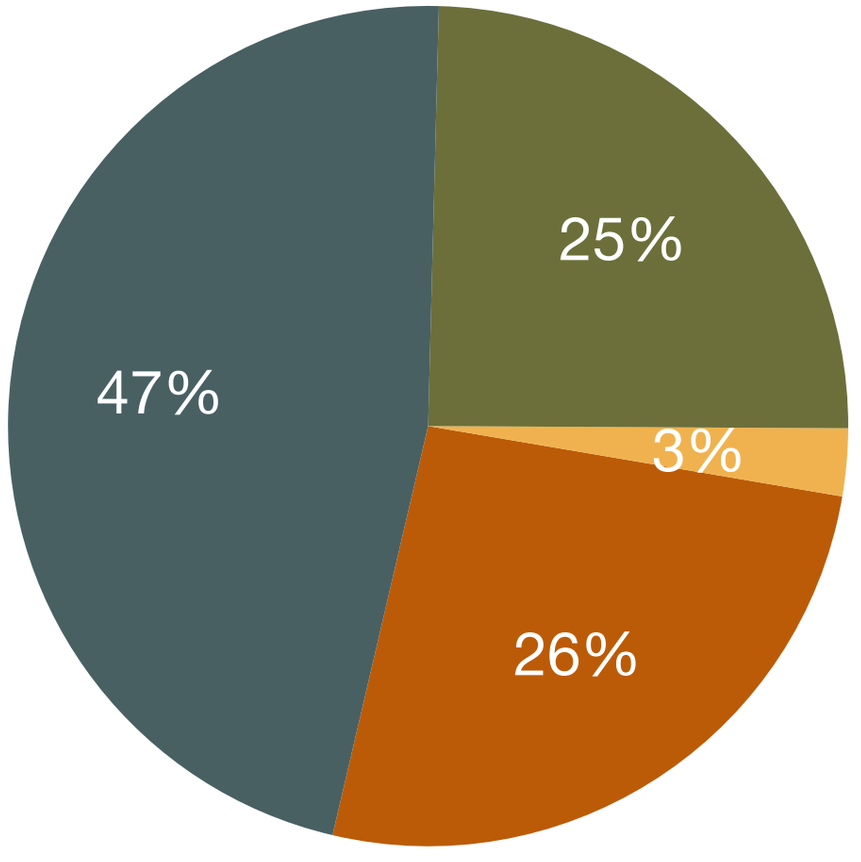
Effectiveness

Analysis Results of
320 Maldoc Samples



No Divergence

Breakdown of 77
potentially false positives



Targets old versions
Mis-classified by AV vendor
No malicious activity trigerred
Unknown

Time and Resource Usages

Average Analysis Time Breakdown
(unit. Seconds)

Item	Windows	MacOS
Snapshot restore	9.7	12.6
Document parsing	0.5	0.6
Script execution	10.5	5.1
Element rendering	7.3	6.2
Total	**23.7**	**22.1**

Resource Usages

- 2GB memory per running virtual machine.
- 60GB disk space for Windows and MacOS snapshots that each corresponds to one of the 6 Adobe Readers versions.

Evaluation Highlights

- Confirms our fundamental assumption in general:
  - benign document “behaves” the same
  - malicious document “behaves” differently
- PlatPal is subject to the pitfalls of dynamic analysis
  - i.e., prepare the environment to lure the malicious behaviors
- Incurs reasonable analysis time to make PlatPal practical

Further Analysis

- What could be the root causes of these divergences?

Diversified Factors across Platforms

Category	Factor	Windows	MacOS
Shellcode Creation			
Memory Management			
Platform Features			

Diversified Factors across Platforms

Category	Factor	Windows	MacOS
Shellcode Creation	Syscall semantics	Both the syscall number and the register set used to hold syscall arguments are different	
	Calling convention	*rcx, rdx, r8* for first 3 args	*rdi, rsi, rdx* for first 3 args
	Library dependencies	e.g., *LoadLibraryA*	e.g. *dlopen*
Memory Management			
Platform Features			

Diversified Factors across Platforms

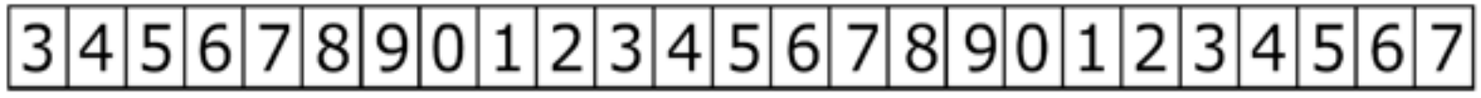
Category	Factor	Windows	MacOS
Shellcode Creation	Syscall semantics	Both the syscall number and the register set used to hold syscall arguments are different	
	Calling convention	*rcx, rdx, r8* for first 3 args	*rdi, rsi, rdx* for first 3 args
	Library dependencies	e.g., *LoadLibraryA*	e.g. *dlopen*
Memory Management	Memory layout	Offset from attack point (e.g., overflowed buffer) to target address (e.g., vtable entries) are different	
	Heap management	Segment heap	Magazine malloc
Platform Features			

Diversified Factors across Platforms

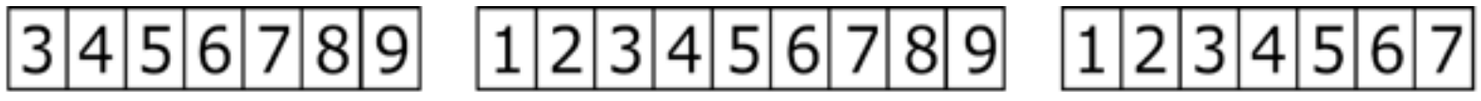
Category	Factor	Windows	MacOS
Shellcode Creation	Syscall semantics	Both the syscall number and the register set used to hold syscall arguments are different	
	Calling convention	*rcx, rdx, r8* for first 3 args	*rdi, rsi, rdx* for first 3 args
	Library dependencies	e.g., *LoadLibraryA*	e.g. *dlopen*
Memory Management	Memory layout	Offset from attack point (e.g., overflowed buffer) to target address (e.g., vtable entries) are different	
	Heap management	Segment heap	Magazine malloc
Platform Features	Executable format	COM, PE, NE	Mach-O
	Filesystem semantics	\ as separator, prefixed drive letter C:\	/ as separator, no prefixed drive letter
	Config and info hub	registry	proc
	Expected programs	MS Office, IE, etc	Safari, etc

Back to The Motivating Example

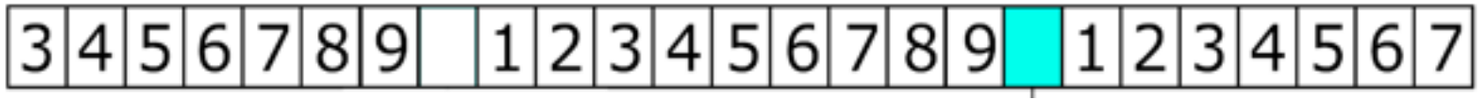
1. Allocate 1000 300-bytes chunks



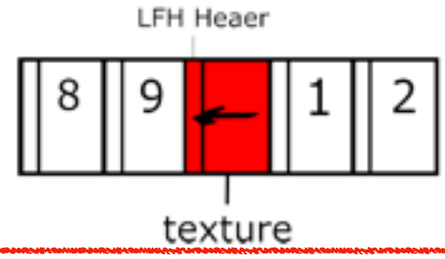
2. Free 1 in every 10



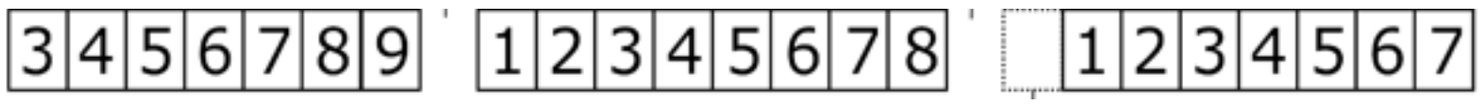
3. Load a 300-byte malicious BMP image



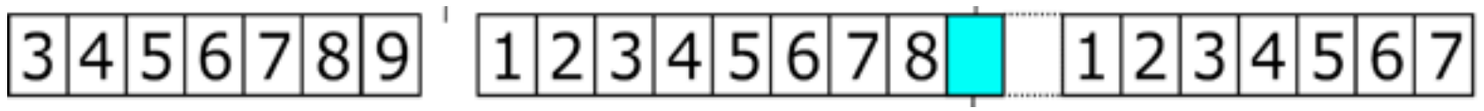
4. Corrupt heap metadata due to a buffer overflow



5. Free BMP image, but what is actually freed is slot 9



6. A *vtable* of 300-byte is allocated on slot 9, which is attacker controlled

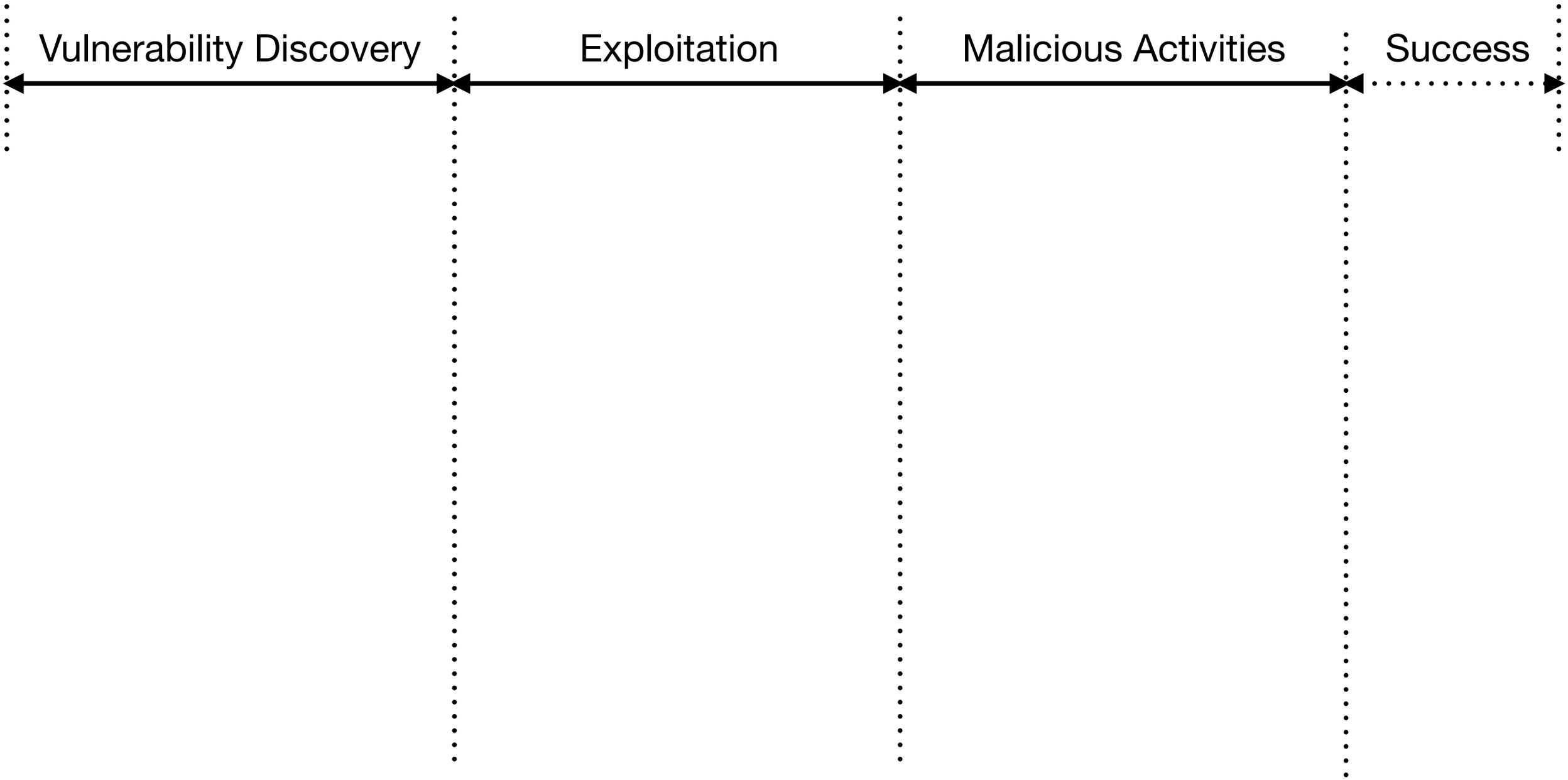


Another Case Study

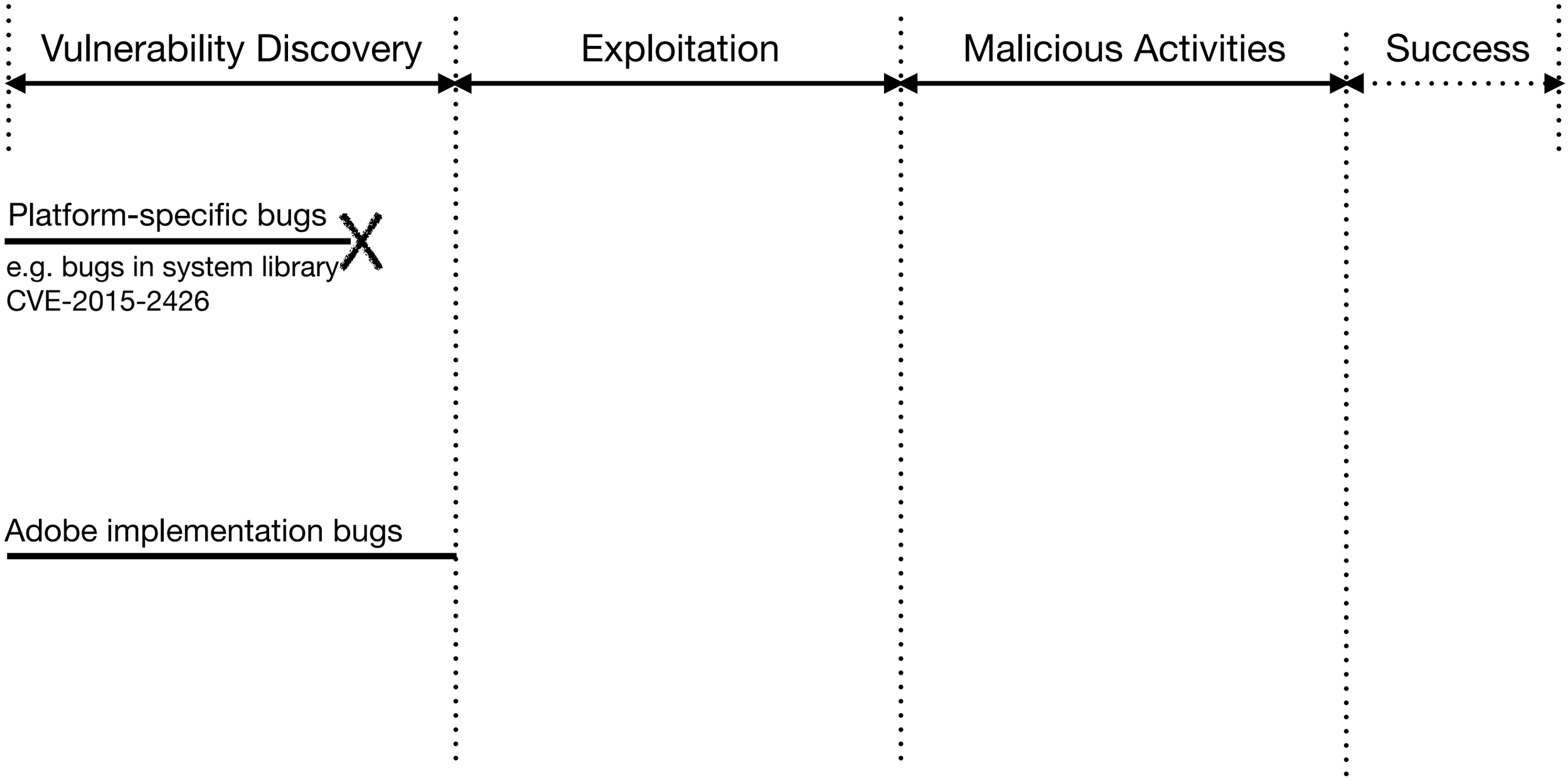
```
1  var t = {};
2  t.__defineSetter__('doc', app.beginPriv);
3  t.__defineSetter__('user', app.trustedFunction);
4  t.__defineSetter__('settings', function() { throw 1; });
5  t.__proto__ = app;
6  try {
7    DynamicAnnotStore.call(t, null, f);
8  } catch(e) {}
9
10 f();
11 function f() {
12   app.beginPriv();
13   var file = '/c/notes/passwords.txt';
14   var secret = util.stringFromStream(
15       util.readFileIntoStream(file, 0)
16       );
17   app.alert(secret);
18   app.endPriv();
19 }
```

CVE-2014-0521 PoC Example

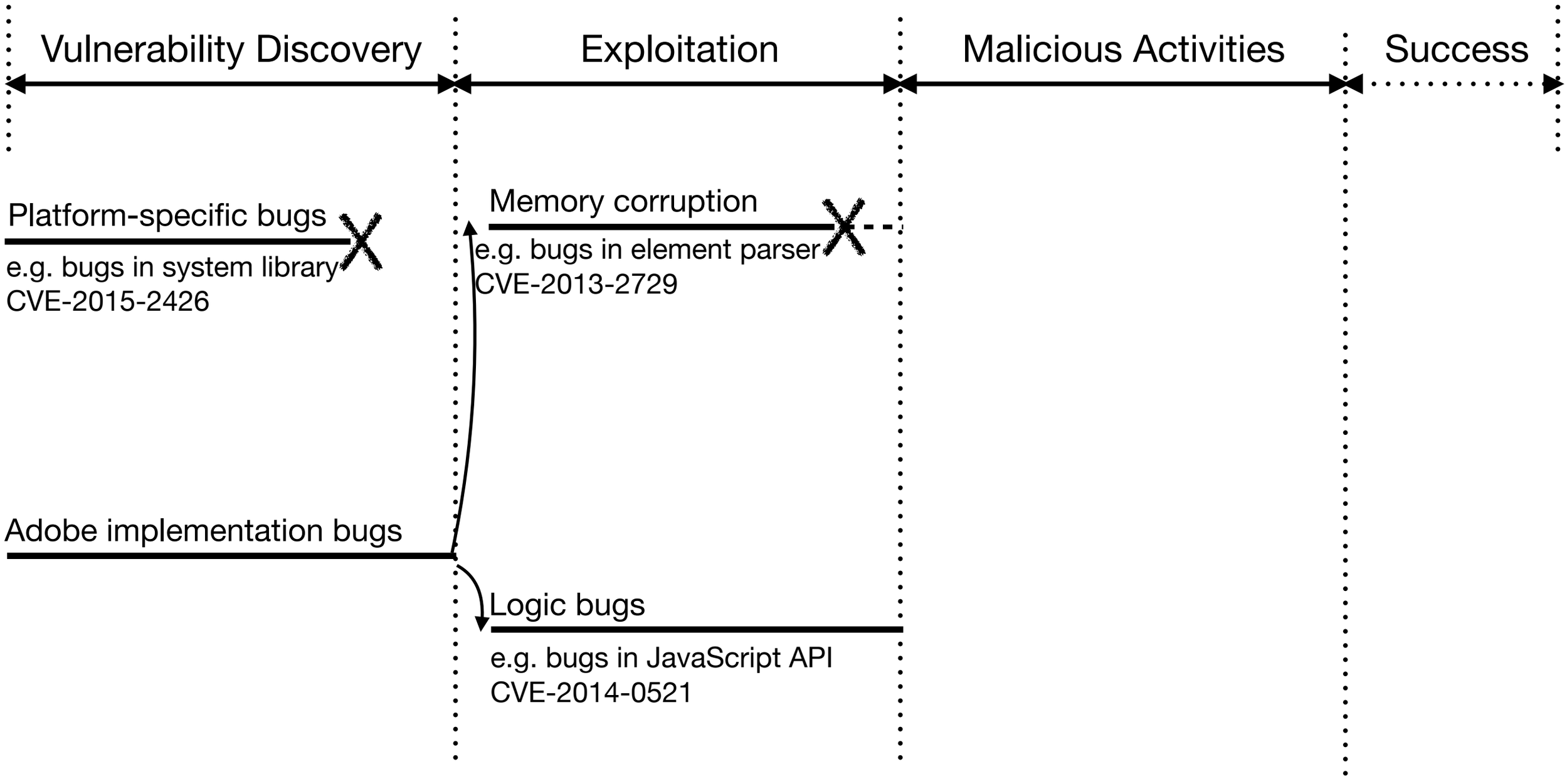
Apply Diversity to Stop Attacks



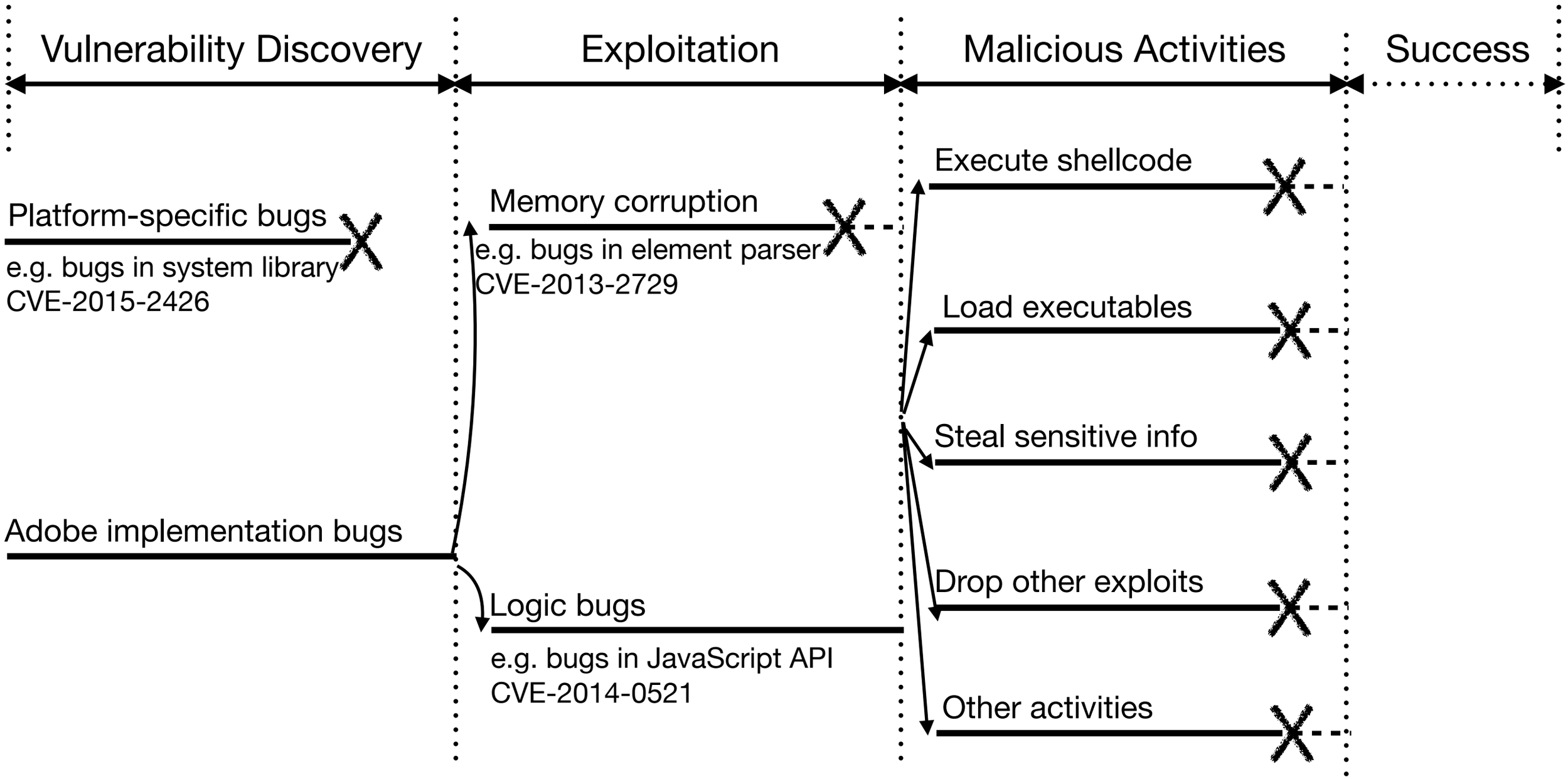
Apply Diversity to Stop Attacks



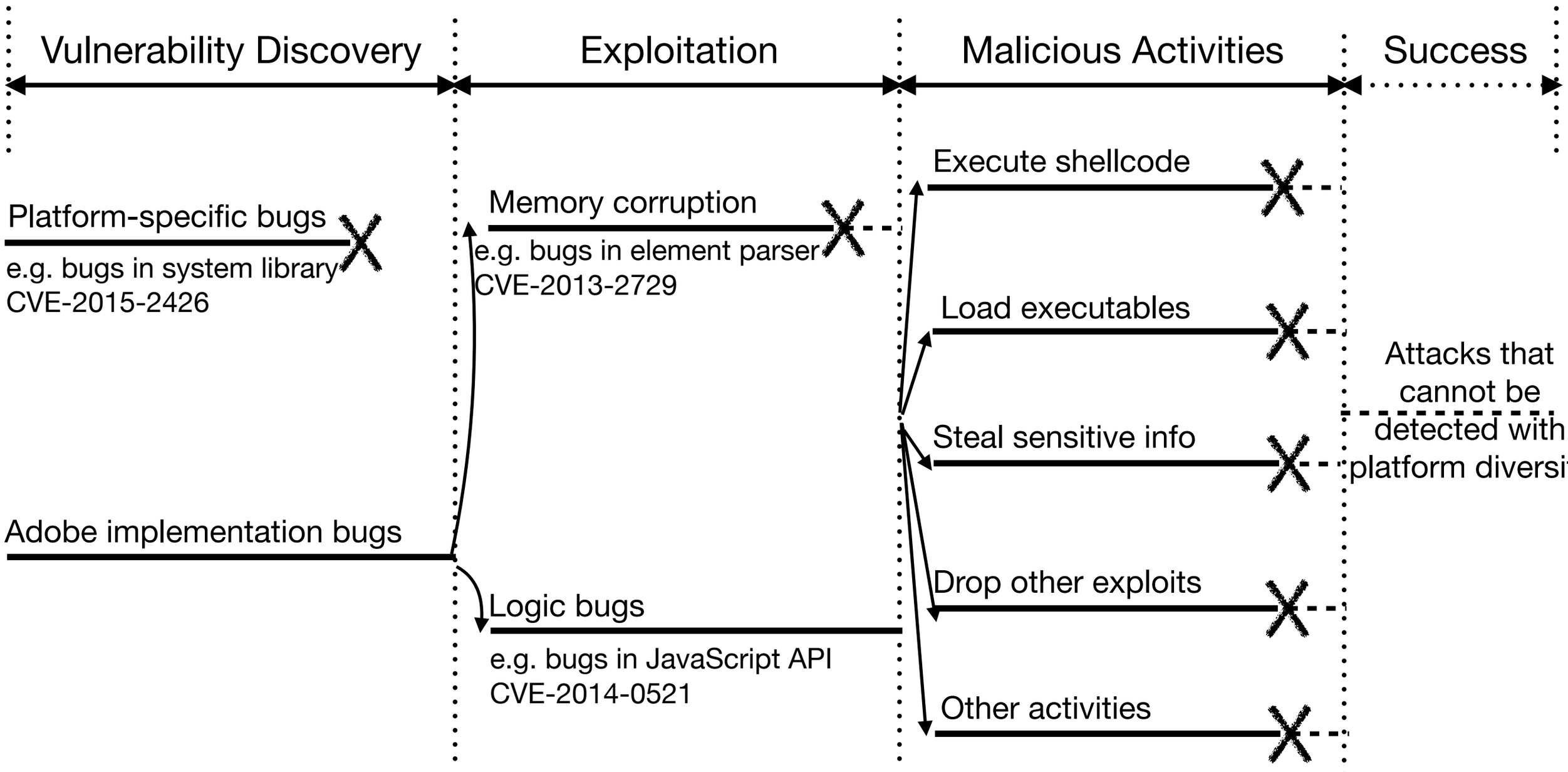
Apply Diversity to Stop Attacks



Apply Diversity to Stop Attacks



Apply Diversity to Stop Attacks



Bypass PlatPal ?

*An attacker has to **<u>simultaneously</u>** compromise all platforms in order to bypass PlatPal.*

Platform-agnostic Attacks

- Heap feng-shui
  - Predict the address of next allocation and de-allocation.
- Heap spray and NOP-sled
  - Alleviate attackers from using precise memory address.
- Polyglot shellcode trampoline
  - Find operations that are meaningful on one platform and NOP on the other.

Limitations of PlatPal

- User-interaction driven attacks
- Social engineering attacks
  - e.g., fake password prompt
- Other none-determinism to cause divergences
  - e.g., JavaScript *gettime* or RNG functions

Potential Deployment of PlatPal

- Not suitable for on-device analysis.
- Best suited for cloud storage providers which can scan for maldocs among existing files or new uploads.
- Also fits the model of online malware scanning services like VirusTotal.
- As a complementary scheme, PlatPal can be integrated with prior works to provide better prediction accuracy.

Conclusion

- It is feasible to harvest platform diversity for malicious document detection.
- PlatPal raises no false alarms in benign samples and detects a variety of behavioral discrepancies in malicious samples.
- PlatPal is scalable with various ways to deploy and integrate.

https://github.com/sslab-gatech/platpal

(Source code will be released soon)