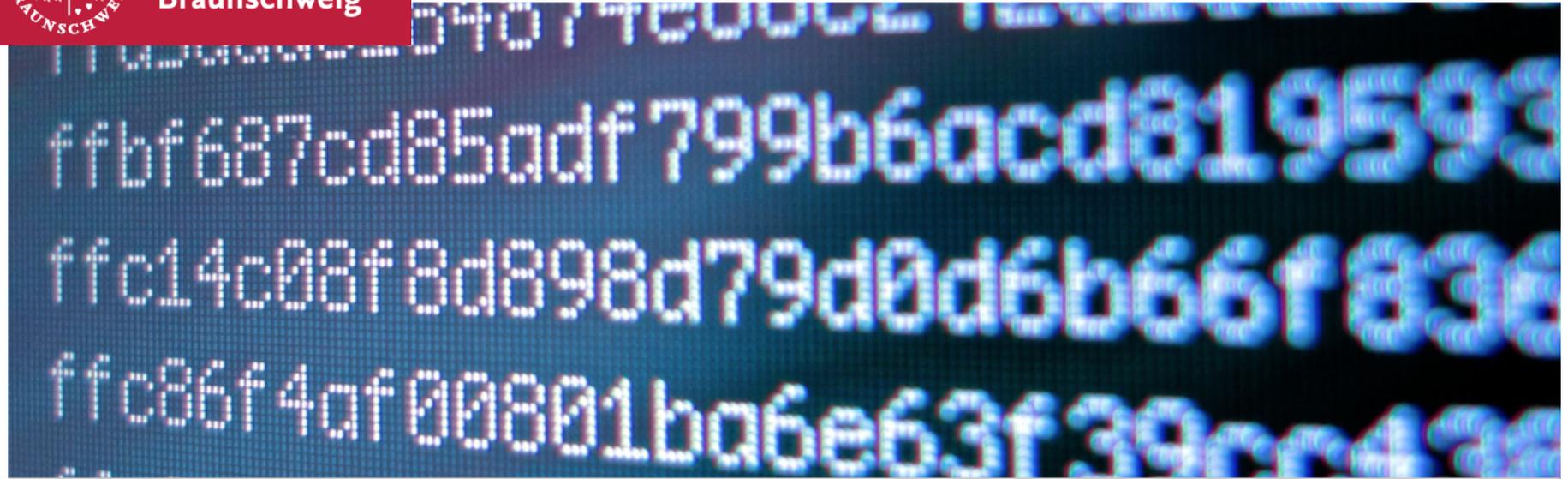




Technische
Universität
Braunschweig

Institute of
System Security



Comprehensive Analysis and Detection of Flash-based Malware

Christian Wressnegger, Fabian Yamaguchi,
Daniel Arp, and Konrad Rieck



DIMVA 2016
Donostia-San Sebastián

Malware

- **Malicious software (Malware)**
 - Lasting problem of computer security
 - Omnipresence of Trojans, Bots, Adware, ...
 - Increase of targeted attacks using Malware
- **Flash-based malware**
 - Malware targeting the Adobe Flash platform
 - Drive-by-Downloads, malicious redirects, exploits, ...



Adobe Flash

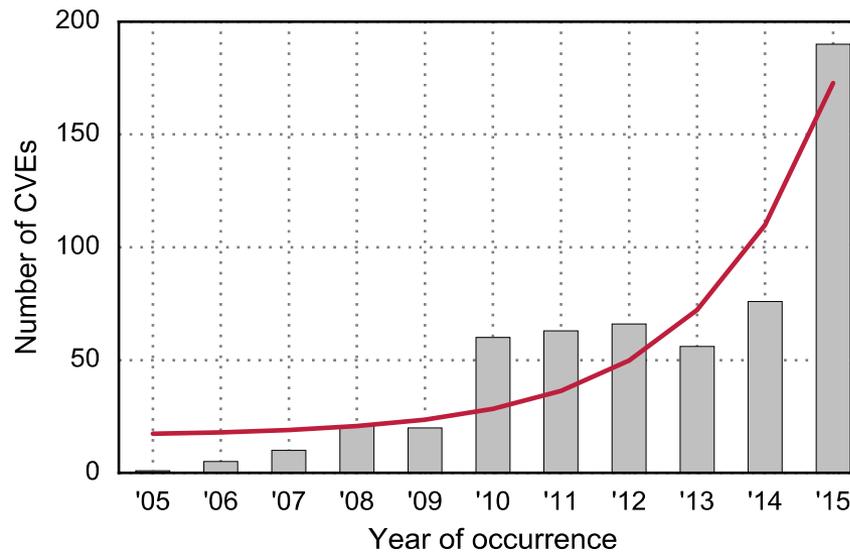
- **Flash is dead!**
 - Deployed on 500 million devices across different platforms
 - Used on 25% of the top 1,000 Alexa web sites
- **Dynamic and multimedia content on web pages**
 - *Advertisement, video streaming, gaming, ...*
 - 20 years of deployment
 - Powerful scripting language: **ActionScript**



Adobe Flash Vulnerabilities

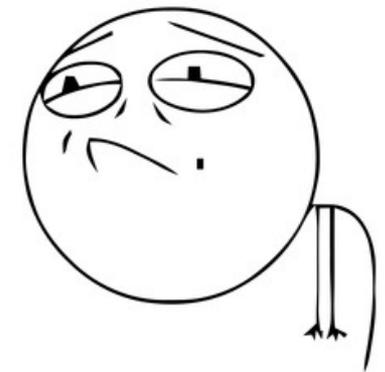
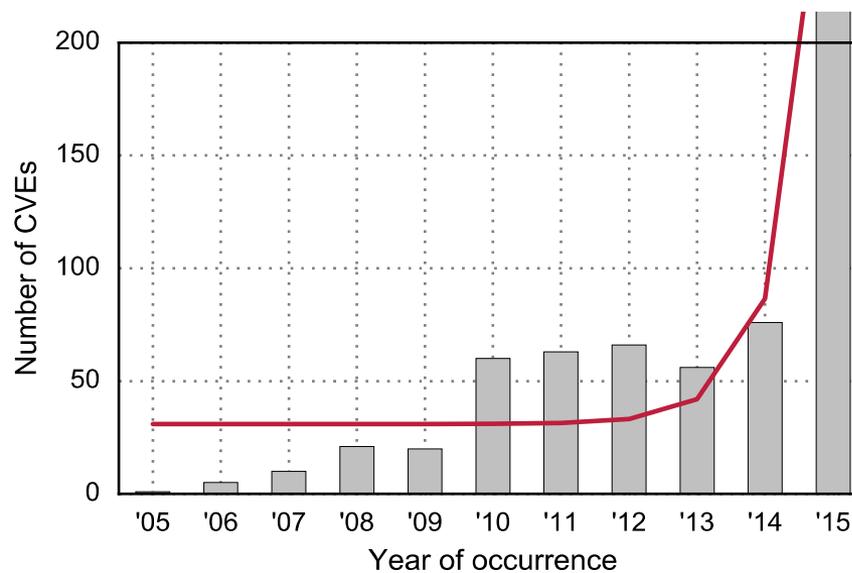
- **Increasing number of CVEs**
 - About 550 different vulnerabilities in total
 - **Until 2015:** 167 new vulnerabilities (80% code execution)

Disclaimer! Effective August 2015



Adobe Flash Vulnerabilities

- **Increasing number of CVEs**
 - About **840** different vulnerabilities in total
 - ~~Until 2015: 167 new vulnerabilities (80% code execution)~~
314 new vulnerabilities (85% code execution)



Attack Vectors and Scenarios

1. Structural Exploits against the Flash Player

- Vulnerabilities in the file format parser

2. Malicious ActionScript code

- Launching or preparing exploits (*Obfuscation, heap-spraying, ...*)

3. Environment fingerprinting

- Selecting targets based on interpreter or OS information

Concrete attacks may fall into more than one of these categories



Obfuscation

- **Staged execution**
 - Dynamic code-loading in form of another animation
loadMovie (ActionScript 2), Loader object (ActionScript 3)
 - Layered encryption/ polymorphism
Runtime-packers (secureSWF, DoSWF)
- **Source-code Obfuscation**
 - Variable substitution, string assembly, dead code, etc.
- **Probing the execution environment**
 - Triggering a malware's payload on specific systems only





- **Comprehensive analysis of Flash animations**

Support for all versions of ActionScript and Adobe Flash platforms

- Structural Analysis (*static*)
- Guided code-execution (*dynamic*)

- **Learning-based detection of Flash-based malware**

- Detects **90–95%** of malicious Flash files at **0.1%** and **1.0%** FPs
 - Significantly outperforms related approaches
 - Best learning-based detector for Flash-based Malware
- No need for manually constructed detection rules



Structural Analysis

- **Flash animations are composed out of “tags”**
 - Containers to store code, animation specs and data (*audio, video, images, fonts, etc.*)
 - Future versions may extend on the number of tags
 - Possibly occurring nested (*DefineShape, ...*)
- **Offering a huge attack surface**
 - Many exploits rely on a specific (sequences of) tag
 - Memory corruption exploits such as **CVE-2007-0071**



Structure Reports

- Exemplary report for a LadyBoyle sample using CVE-2015-323

```
69 FileAttributes
77 Metadata
  9 SetBackgroundColor
  2 DefineShape
    39 DefineSprite
    26 PlaceObject2
86 DefineSceneAndFrameLabelData
43 FrameLabel
87 DefineBinaryData // Payload
87 DefineBinaryData // Payload
82 DoABC // ActionScript 3
76 SymbolClass
  1 ShowFrame
```

- **More compact:** 69 77 9 2 [39 26] 86 43 87 87 82 76 1

md5: cac794adea27aa54f2e5ac3151050845



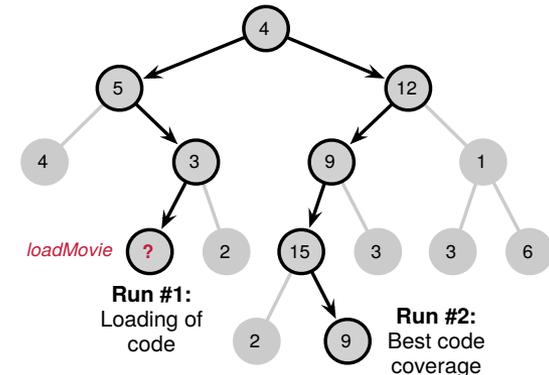
Analyzing Code

- **Dynamic code analysis**
 - Single execution “as-is” is not sufficient
 - Covering all execution paths is not feasible
 - **Heuristics needed!**
- **Previous approaches, e.g.,**
 - Determine which paths to execute based on external input
(“Exploring Multiple Execution Paths for Malware Analysis“, Moser et al.)
 - Multi-execution of branches along the intended path
(“Rozzle: De-cloaking Internet Malware“, Kolbitsch et al.)



Guided Code-Execution

- **Gordon:** Guide the interpreter towards indicative code regions
 - Branches that contains indicative functions
(*loadMovie, loadBytes, ByteArray, ...*)
 - Paths with many instructions
- **Two-step procedure**
 - Determine Control-flow statically
 - Use CFG to guide the analyzer
 - Multiple runs possible
 - Force Execution at environment sensitive conditions



Execution Reports

- **Excerpt of a report for a sample using CVE-2015-323**

```
R1 973:  pushString    "fla"  
R1 975:  pushString    "sh.uti"  
R1 977:  add          "fla" + "sh.uti"  
R1 978:  pushString    "ls.Byt"  
R1 980:  add          "flash.uti" + "ls.Byt"  
R1 981:  pushString    "eArray"  
R1 983:  add          "flash.utils.ByteArray" + "eArray"  
R1 984:  callProperty [ns:flash.utils] getDefinitionByName 1  
R1 >    Looking for definition of  
R1 >          [ns:flash.utils] ByteArray  
R1 >    Getting definition for  
R1 >          [ns:flash.utils] ByteArray  
R1 987:  getLex: [ns:] Class
```

- **For automatic processing reports meta data is omitted**

md5: 4f293f0bda8f851525f28466882125b7



Learning-based Detection

- **Preprocessing of reports**
 - **Structure reports:** cf. compact representation
 - **Execution reports:** Instruction names and parameters only
 - Parameters are replaced with their respective type
- **n -gram models of structure and execution reports**
 - Observe occurrences of **token** n -grams
 - No need for manually constructed detection rules
(instruction counts, call frequencies, etc.)
- **Classification using Support Vector Machines (SVMs)**



Evaluation

- **Datasets**
 - 26,600 Flash Animations collected over 12 weeks
 - 1,923 malicious and 24,671 benign samples
- **How well are we able to detect Flash-based malware?**
 - Comparison to the state-of-the-art methods
- **What's all the fuss about two different analyses?**
 - Wouldn't be one of them enough?



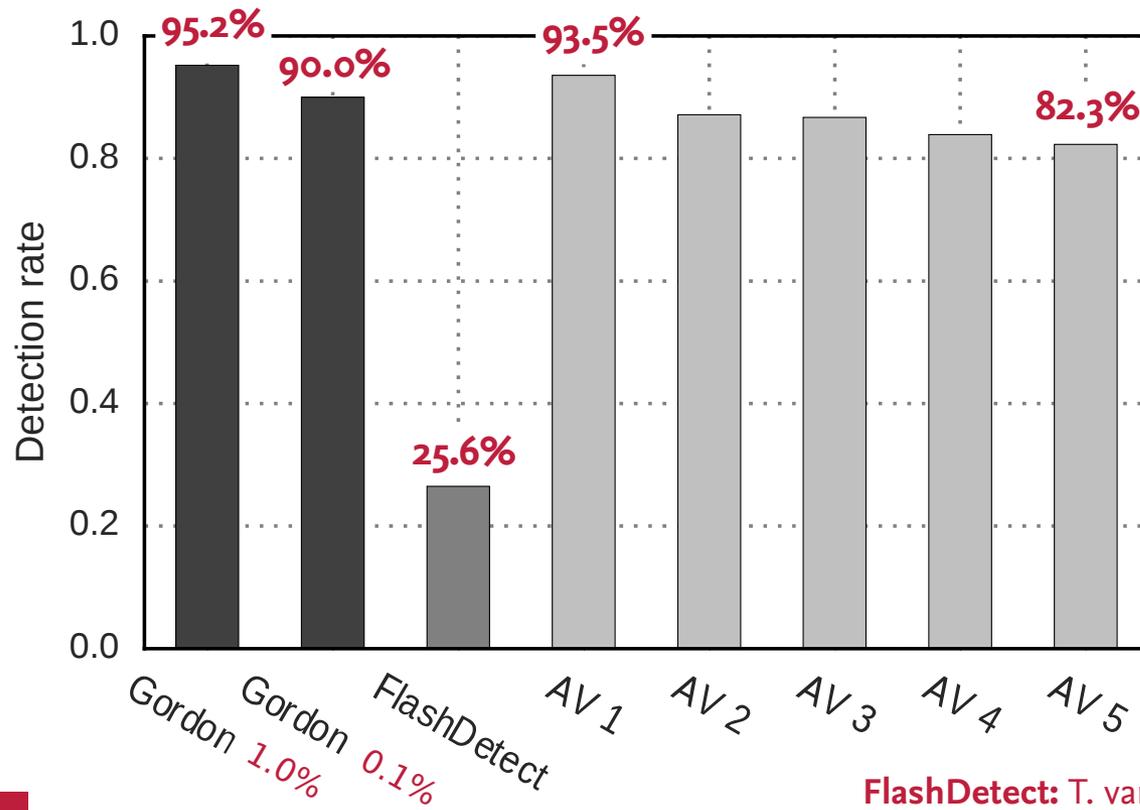
Experimental Setting

- **Temporal split of the data**
 - Weeks 1-6 for **training**, weeks 7-9 for **validation**, and the remainder, weeks 10-12 for **testing**
 - **All test data has been collected after training**
- **Related approaches**
 - FlashDetect (*T. van Overveldt et al, RAID 2012*)
 - Adjusted to 1% false-positives
 - Not supported version have been excluded (version 8 and below)
 - Virus scanners listed at VirusTotal



Comparative Evaluation

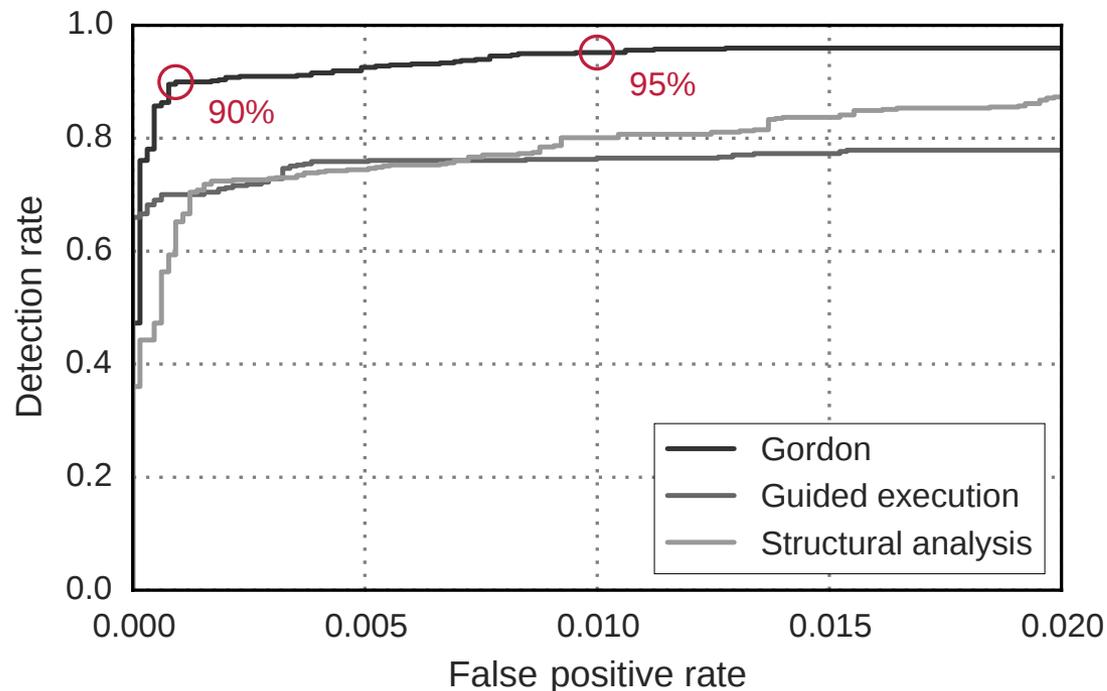
- **Gordon is on a par with tradition approaches**
 - No manual effort needed, though



FlashDetect: T. van Overveldt et al, RAID 2012

Combined Detection Performance

- **Gordon benefits from two orthogonal analyses**
 - Individual representations only detect 60–65% at 0.1% FPs



Summary

- **Comprehensive Analysis of Flash-based malware**
 - Structural analysis
 - Guided code-execution
 - Directed analysis of **indicative code regions**
- **Effective Detection of a large variety of Flash-based malware**
 - High detection rate: **90–95% of malicious samples**
 - Low false-positive rates
 - Best learning-based detector for Flash-based Malware
 - Can be used to bootstrap traditional methods

