

Static Analysis and Dynamic Instrumentation for Intelligent Exploit Analysis:

Abstract:

With the rise in number of state sponsored APT attacks targeting government and private companies, there lies an improved requirement in automated exploit analysis and filtering document file formats.

There are a huge no of security devices out there that promises to do most of the detection. Our talk would be on the current drawbacks of these systems and how our automated system handles these drawbacks. The aim of the talk would be to explain the intelligence that we have added on to our tool, so that common users |government employees could learn how they could utilize these techniques in detecting these sort of APT attacks targeting them.

We been working on an exploit analysis system named "Sandy" a free tool developed under Indian Honeynet project. And in my talk I would pass on to the users the various techniques I have learned from my past 8 months of adventures we had with exploit analysis, that involves but not limited to exploit obfuscation, exploit reliability, automated analysis bypass, APT attribution, multi targeting and everything that makes APT attacks scary.

About Sandy:

Sandy stands for Static and Dynamic analysis. Sandy is capable of doing both static and dynamic analysis of Malicious Office, Jar, Flash, HTML files at the moment. Our aim is to build a sandbox specialized in processing file format exploits.

The main aim of sandy is to extract the embedded executable, dropped documents and url controllers from these file formats and provide attribution to the Attack groups and there technology. Sandy initial analysis it performs multiple static analysis, that included detecting simple XOR, ROL, ROR encryption, Packer detection, Signature scan, Shellcode Detection, Meta Data analysis, Entropy and Cryptanalysis, File version detection and finally provides the extracted analysis data after processing for download to the end user. Once the static analysis is finished the data generated is passed on to our dynamic analysis box for improved efficiency. All current systems out there blindly pass exploit samples to a dynamic sandbox. But sandy uses the static analysis data to do an intelligent dynamic analysis, there by making the system unique.

Automated Static Analysis of Exploit Code:

Our advantage is the capability of doing cryptanalysis to find malicious objects in file formats. We depend mostly on file entropy based detection for finding malicious files. And if the initial statics analysis fails, the metadata and file version extracted is used to submit the data to Sandy's dynamic analysis

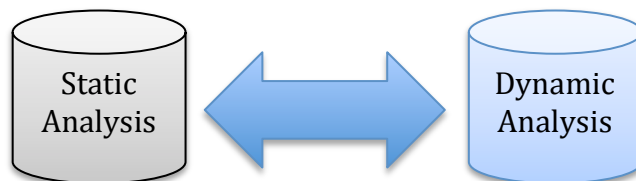
sandbox. The many similar tools out there are only capable of handling specific file formats, but our support for both static and dynamic analysis on [Office, PDF, Jar, Flash, HTML] as well as both static and dynamic analysis makes sandy unique.

Dynamic Instrumentation for Intelligent Exploit Analysis:

For example in this section we are going to highlight few techniques we use to mass analyze Java exploits.

How Sandy Handles Java Exploits:

Once the static analysis is done on Java files, which includes, decompiling, string analysis, version detection, and function sequence analysis, code review. We gather valuable information from the static analysis and move on to dynamic phase, which includes dynamic instrumentation for a managed Java runtime environment. Our technique allows us to intercept Java applications allowing us to modify and understand instructions at runtime. This is very much important when obfuscation of Java exploits is widely deployed for hiding binary, controller information's. Later the dynamic analyzed data is passed on to a static analysis engine. And these hybrid analyses make exploit analysis unique. We would be explaining our dynamic instrumentation on Java exploit in our talk.



What users would benefit out of the Talk.

This talk would explain the various challenges involved in exploit analysis, and would guide you to handle and detect APT attacks. It will also guide you to the tools and techniques for handling exploit analysis and code instrumentation. The talk is aimed at government employees, corporates, Antivirus companies or any one who could be a potential victim to APT attacks or any one who want to learn exploit analysis and detection.

Why is this talk different?

Exploit analysis is one thing every security | research company is interested in, and the amount of exploit companies collected daily is a huge no, so there is a necessity for an automated system specialized in exploit analysis for improved efficiency. And there are very little talks that highlight one such tool or technique.

Presentation Time Required (20, 30, 50 Minutes): 35-45 minutes

- Is there any demonstration? : Yes
- Are you releasing any new tool? : Yes
- Are you releasing any new exploit? No
- Have you presented the paper before on any other security / technology conference(s)? : Not in any public events.

Author:

Rahul Sasi (fb1h2s) is working as a Security Researcher working for a Global Research firm. He has authored multiple security tools, advisories and articles. He has been invited to speak at various security conferences like HITB [KL], BlackHat [US Arsenal], Cocon (2011, 2012), Nullcon (2011, 2012,2013), HITB (AMS 2012,2013), BlackHat (EU 2012), EKoparty (Argentina),CanSecwest(Canada 2013). His work could be found at www.Garage4Hackers.com.

Sandy in Action: [Ver 1.0 Pre release version]

Front End:



Currently Supports [Jar,Doc,PDF,Flash,RTF] Files

Sandy the SandBox

Drop Exploits in this Box

Select Multiple Documents



Processed files:

Scanned Files

MDS	Filename	File Size	File Type
a6952e9072aa11d448d6de411018dab0	1c956c5d2e718ac11b3b744784c101271c275a068ec5382919bfff9a98b555bad	117248	application/octet-stream
39fab366217b20973a1431a9f42d008d	4e7b51eda4d48acfc2babfdd9f4dd9f6a55cbfe99aa3eccc3983dd8cdf519416	133120	application/octet-stream
f0ec397575700a4beaa9366a6adc183	9ac152e0dc74e273a5cc5acf717e8a02bab3fab12abc0420fba29a869372b94	112544	application/octet-stream
f97e80ed1a2c7c6d6ac46291720c4bcc	43f83a204bccfeabe079cd70517f42d4ce0c903969486ce86fc148eb1b897705	141896	application/octet-stream
33de32ddaba2ee746a6cfd140a991900	9d2e8bbdd88dedd2be45692bc87f913d0b67e7cdf8d2996fd50b617cbfe086b	207650	application/octet-stream
997a64fe5c8e55689482db7d1567f0f3	d26855802a17ff7d15ba5e0e85dfe0fc675cd57e86b74f5d3c9aa279ada1881	460288	application/octet-stream
69b5d014d5e06c2887ce270c48bc82ad	e8ae02ade2ba83bb5652a0bfea07e2ab3546b8afd070333fb2111778aeb6a54	216455	application/octet-stream
127e530882674aa9a0e85a80ce8c8b8b6	6a83738bbdef1c43095f276d53b65d75172968716dcaba3a3c75dd57c6f778a9	82232	application/octet-stream
fa01407724ea8b29a9cb91dcab8913c3	6ca86689f54543c9f282c4748ad7eba9e2b913f0f2a33474d1100e0930e6f65c	100935	application/octet-stream
39fab366217b20973a1431a9f42d008d	4e7b51eda4d48acfc2babfdd9f4dd9f6a55cbfe99aa3eccc3983dd8cdf519416	133120	application/octet-stream
1962ba9f28326c977f416a9ea29c9684	5f841f1ec31cfd6d09f7007019f5cbf19d6e6ff829e3a0111b204cbe01c4b217	148711	application/octet-stream
f0ec397575700a4beaa9366a6adc183	9ac152e0dc74e273a5cc5acf717e8a02bab3fab12abc0420fba29a869372b94	112544	application/octet-stream
39fab366217b20973a1431a9f42d008d	4e7b51eda4d48acfc2babfdd9f4dd9f6a55cbfe99aa3eccc3983dd8cdf519416	133120	application/octet-stream
39fab366217b20973a1431a9f42d008d	4e7b51eda4d48acfc2babfdd9f4dd9f6a55cbfe99aa3eccc3983dd8cdf519416	133120	application/octet-stream
f0ec397575700a4beaa9366a6adc183	9ac152e0dc74e273a5cc5acf717e8a02bab3fab12abc0420fba29a869372b94	112544	application/octet-stream
afe5c6af476b55ca0a0aeb8383da6e	d1b60893c66776875e185f5d6222ab69fc8294fa94022b61a07170ecb9bda8b2	148199	application/octet-stream
3a11482a7a661a452731ca9ffed6b02e	a932bbe85bd3618e893c696dd2ac7a82781828d9aabcde955497dfcc4caf10e4	211968	application/octet-stream
h8Re9e4974d8d320nah1h3ah4d0h7c	Rc2erc4d41a33h209e195a6r6rcera7695a76d73077906h0275711z:c92rRdaRRc	58528	application/octet-stream

Extracted Information:

```

Comment: DocSumInfo_19: False*
Comment: DocSumInfo_22: False*
Comment: Template: 'Normal'
Comment: RevisionNumber: 2'
Comment: TotalEditingTime: 0:01:00'
Comment: NumWords: 7'
Comment: NumCharacters: 41'
Comment: Security: 0'
Comment: Encrypted: 'True'
MIME type: application/msword'
Endianness: Little endian']

```

File Static Analysis

Signatures Analysis

```

!Malicious Signatures found
!Malicious Signatures found

```

Extracted Binary Information

```

Download Dropped Files
Embedded Flash File embedded.swf
Download the Binary
UPX Packed Embedded Executable embeded.exe

```

Sandbox Analysis Information

**Pending Job

Sandy Processing Jar files:

URLS FOUND

http://www.sms911.ru
http://www.sms911.ru/
http://depositmobi.com/
http://depositmobi.com/
http://depositmobi.com/conditions.html
http://www.mobi911.ru
http://www.mobi911.ru/
http://www.mobi911.ru/

















http://www.mobi911.ru/

FILE VIEWER

Md5	Filename	Filetype
e9910f6aceff7daf77efed3698a618b2	Main.class	compiled Java class data, version 47.0 (Java 1.3)
babeaaaa72490d81e711090e264d21e	a.class	compiled Java class data, version 47.0 (Java 1.3)

2bb2240a8fb4e1947d11b3e652d696f	21.temp	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
2bb2240a8fb4e1947d11b3e652d696f	22.temp	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
2bb2240a8fb4e1947d11b3e652d696f	23.temp	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
2bb2240a8fb4e1947d11b3e652d696f	24.temp	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows

Decompiled Class files

Name	Size	Date modified	Download View
 a.java	2.67 kB	06 06 2013 05:40:17	 
 b.java	2.49 kB	06 06 2013 05:40:17	 
 c.java	3.58 kB	06 06 2013 05:40:17	 
 d.java	861 B	06 06 2013 05:40:17	 
 e.java	6.38 kB	06 06 2013 05:40:17	 
 f.java	879 B	06 06 2013 05:40:17	