

Weapons of Targeted Attack

Modern Document Exploit Techniques

Ming-chieh Pan <naninb@gmail.com>
Sung-ting Tsai <ttsecurity@gmail.com>

Blackhat USA 2011

Abstract

The most common and effective way is using document exploit in the targeted attack. Due to the political issue, we have had opportunities to observe APT (advanced persistent threat) attacks in Taiwan since 2004. Therefore we have studied and researched malicious document for a long period of time.

Recently, we found APT attacks (e.g. RSA) used the same technique as we disclosed last year, e.g. embedding flash exploit in an excel document. In order to protect users against malicious document and targeted attacks, we would like to discuss the past, present, and future of document exploit from technical perspective, and predict possible techniques could be used in a malicious document in the future by demonstrating "proof of concept" exploits.

The presentation will cover four major types of document attacks:

- Advanced fuzzing techniques.
- Techniques to against exploit mitigation technologies (DEP/ASLR).
- Techniques to bypass sandbox and policy control.
- Techniques to defeat behavior based protection, such as host IPS.

Contents

Abstract.....	1
1. Introduction.....	3
1.1. Background.....	3
1.2. Targeted Attack and Document Exploit	3
1.3. Cat and Mouse Game	3
1.4. Contents of the Paper	3
2. Recent Document Exploit Attacks	4
2.1. Hybrid Document Exploit	4
2.2. Incomplete Protection	4
2.3. Advanced Memory Attack Techniques	5
2.4. Vendor Responses.....	5
2.5. Our Finding in Real Attacks	5
3. Future Document Exploit Attacks.....	5
3.1. Advanced Fuzzing Techniques	5
3.2. Techniques to Against Exploit Mitigation Technologies	6
3.3. Techniques to Bypass Sandbox / Policy / Access control.....	8
3.4. Techniques to defeat behavior based protection and automatic analyzing sandbox.....	9
4. Conclusion	10
Reference	12

1. Introduction

1.1. Background

APT (Advanced Persistent Threat) has become very popular in 2011. Actually we have already known this kind of attack since 2004. Due to the political issue, Government units and large enterprises in Taiwan has been targeted for many years. They have kept receiving purpose-made e-mails and malwares (exploits), never stopped. Thus we have chances to observe the attack trend and we also spent a lot of time on document exploit research.

Nowadays, not only in Taiwan, this kind of silent threat are attacking whole world, e.g. Google Aurora attack and recent RSA attack. Unlike normal cyber-criminals, they are hacking for the information, not for profit. And unfortunately, most of security software couldn't protect effectively.

We are going to discuss document exploit from technical perspective, introduce attack techniques that might be used in future. We wish application and security vendors could be aware of the attack and have new approaches to protect people.

1.2. Targeted Attack and Document Exploit

Attacker sends an e-mail with specific content and document exploit (antivirus couldn't detect) to his targets. After open the document, attacker could take control the victim's system. It is the most common way and not easy to be aware of. The malicious document usually includes malicious web page (attacking browsers), office document, PDF, and Flash.

Document exploit is actually the weapon of targeted attack.

1.3. Cat and Mouse Game

Exploit attack and defense is like a cat and mouse game. Vendors keep patching application and inventing new technologies to prevent attack, however attackers always can find ways to defeat those protections. So if we could be ahead of attackers by guessing their next tricks, we might have better protections for people.

1.4. Contents of the Paper

In this paper we will discuss document exploit from technical perspective. Recent document exploit techniques will be introduced in chapter 2.

Chapter 3 will cover four major types of new document attacks, including our latest findings:

- **Advanced fuzzing techniques:** our flash AVM fuzzing technique will be introduced.
- **Techniques to against exploit mitigation technologies:** our new JIT spraying techniques will be introduced.
- **Techniques to bypass sandbox and policy control:** a flash vulnerability will be introduced as an example.
- **Techniques to defeat behavior based protection:** new approaches to write a document exploit. This will make security vendors headache.

2. Recent Document Exploit Attacks

2.1. Hybrid Document Exploit

If you have installed all Microsoft office patches, and there is no 0-day vulnerability and exploit. Will it be 100% safe to open a word or excel document? The answer is no. Modern document application is very complicated. Most of them could embed document objects of other applications. For example, the Excel could embed an Adobe flash object. In this case, even your Excel is up to date, it is still not 100% safe when you open an Excel document which includes a flash object and your flash application is vulnerable.

Most of people know browser could include a lot of document objects, such as PDF, flash, and other multimedia files. So they are cautious when they open web page. However, when they open a document in the e-mail, they would not be aware of the danger.

This kind of attack is very popular recently. For example, a flash vulnerability could be repacked as a malicious web page, a PDF exploit, or even an office document exploit.

2.2. Incomplete Protection

Application vendors delivered new technologies to make their application safer. Especially the exploit mitigation techniques could do really good jobs to avoid execution of exploits, e.g. DEP and ASLR.

However, it is very difficult to do protections completely. Because application is very complicated as well as the environment of operating system, it is not possible to update every component, every tool to adopt the protection technologies. And you don't need to think that you could ask all users to install updates or manually enable protections.

For example, even you have adopted DEP and ASLR, there are always some researchers could find some modules are not protected by ASLR, and they could use the module to do ROP (return-oriented programming) and make effective exploits.

2.3.Advanced Memory Attack Techniques

Researchers are also finding some new approaches to bypass DEP and ASLR. **Flash JIT spraying** techniques has been introduced in BHDC2010. Flash JIT could bypass DEP, and the spraying technique could defeat ASLR. This technique could exploit the newest Office 2010 and Internet Explorer.

2.4.Vendor Responses

Vendors have been working hard to patch vulnerabilities and adopt new protections in applications. Flash has started to encode/encrypt AVM code area since version 10.1, and the memory area has become non-executable. Also it has better ASLR to arrange its memory sections. These new techniques effectively mitigate JIT spraying exploit.

And Microsoft released Enhanced Mitigation Experience Toolkit 2.0 in BlueHat v10. The EMET tool could provide a lot of memory protections for applications. It could effectively defeat most of exploits with ROP techniques.

2.5.Our Finding in Real Attacks

Recently we found exploit is using the same trick as we disclosed in Syscan 10'. Do you know why attackers don't include a flash exploit in web page or PDF file, and they only use Excel to spread malicious e-mails. The reason is Excel will turn off DEP when a flash object is embedded. It is much easier for attackers to write exploits.

3. Future Document Exploit Attacks

3.1.Advanced Fuzzing Techniques

File format fuzzing is the most common way to discover a vulnerability of document applications. We believe most of document vulnerability discovers (including vendors) are keeping improving their fuzzing tools. We are going to introduce our Flash AVM fuzzing techniques.

Focus on AVM instructions. Take the CVE-2010-1297 as example. Traditional one-byte fuzzing technique modifies each byte of the sample file with 256 values. We found we can focus on the AVM (action script) part, the method_body of code area. And we also found there are only around 170 AVM instructions. So our fuzzing tool could only try the AVM part with 170 values. It reduces the testing range and save a lot of time, and we could still find similar vulnerabilities.

We use the approach to fuzz the CVE-2010-1297, and we also discovered APSB11-12 before it was disclosed. (By inserting a Setlocal_1 (0xd5) in code area)



Furthermore, we use this technique to accidentally find that the JIT spraying technique could still work.

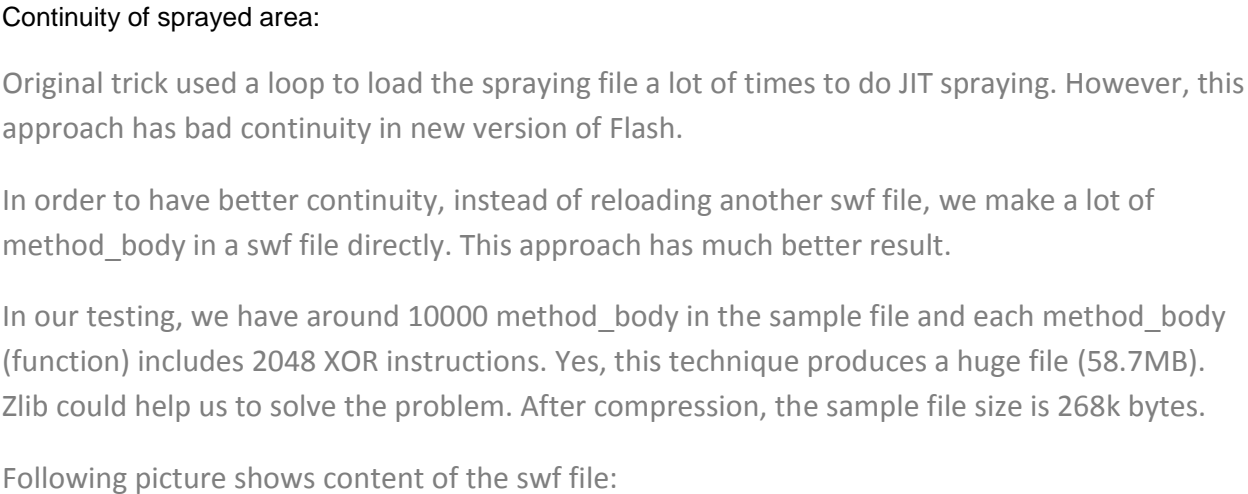
3.2. Techniques to Against Exploit Mitigation Technologies

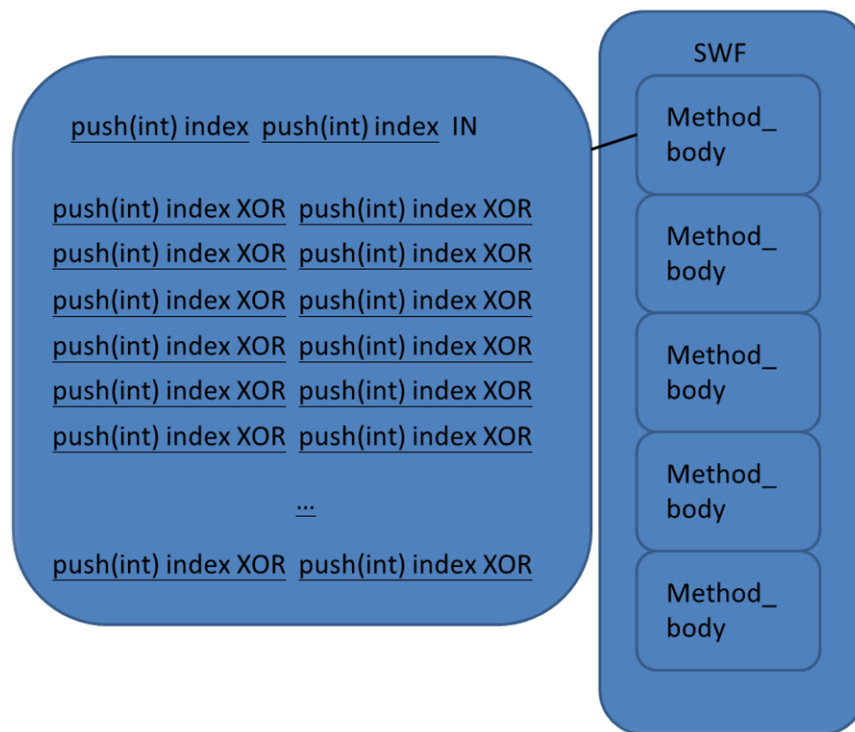
Many researchers are looking for new techniques to bypass DEP and ASLR. We are the same. In this chapter we are going to explain how we bring JIT spraying back, and our JIT spraying improvements.

The magic B4 (IN) instruction:

The original JIT spraying is use '35 90 90 90 3C' to fill up the code area. By our fuzzing technique, we found if we replace the first XOR(AA) with IN(B4), the AVM code area will not be encoded in memory, and memory section will become executable (like before).

Old trick (the XOR trick) could be used again. However, the improved ASLR reduced the success rate. We need some other techniques.





Use OR:

Instead of XOR instruction, we found a better solution. We use OR(A9) instead of XOR(AA) to spray the memory. The content in memory will be '0D 0D 0D 0D 0C', instead of '35 90 90 90 3C'. This technique makes it easier to jump into our sprayed area when trigger the vulnerability.

It works everywhere.

Protection	New JIT Spraying with Flash Player 10.3.181.34 (Released 6/28/2011)
Office2000 ~Office 2010 (DEP AlwaysOn, ASLR)	works
Internet Explorer (DEP AlwaysOn, ASLR)	works
Adobe PDF (DEP AlwaysOn, ASLR)	works
EMET v2.1 (Enabled all functions)	works

3.3. Techniques to Bypass Sandbox / Policy / Access control

Except for memory exploitation, the attack to design of security policy and resource access control will be another topic for document exploit researchers.

In order to provide secure execution environment for clients and users, application usually has complicated policy and permission control to isolate access to each resource. There might be some logic design flaws in applications.

We take a policy design flaw we found recently in Flash as the example. There are 4 types of properties in Flash `Security.SandboxType`: `Security.REMOTE`, `Security.LOCAL_WITH_FILE`, `Security.LOCAL_WITH_NETWORK`, and `Security.LOCAL_TRUSTED`. The basic idea is if you can access network, you can't access local resource, vice versa. The flaw is in its 'url protocol' design.

We embed a Flash object in an Office document. This flash object is allowed to access local files, and not allowed to access internet. However there is a problem when handling the 'mms' protocol. When the flash object opens a mms link, IE will be launched, and then media player will be launched as well. The media player will connect to the link.

Using this flaw, we could retrieve user information, and use mms protocol to send information to internet. For example, we might steal user's cookie, user's saved password, etc. And we could use this technique to probe user environment.

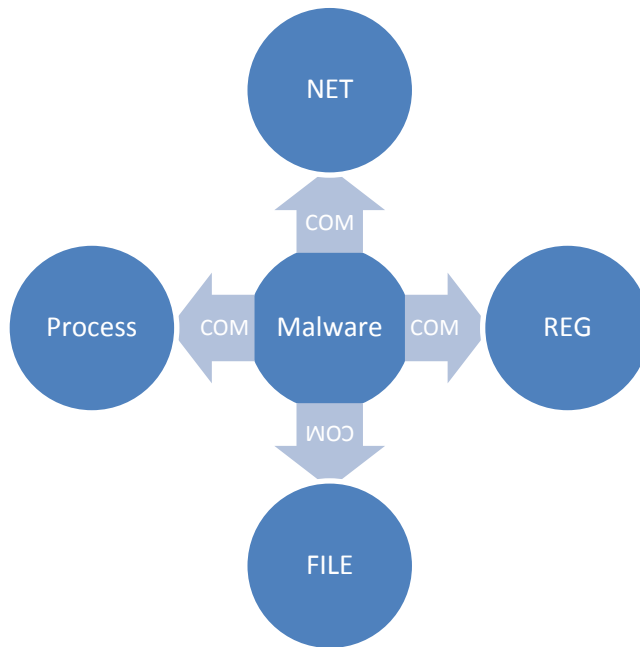
3.4. Techniques to defeat behavior based protection and automatic analyzing sandbox

In case of exploit is launched, traditional signature based malware protection is useless, because the exploit or malware is usually 'customized'. Users can only rely on behavior based protection. For example, the HIPS could block your connection to Internet, block drop files to system folders, block access to sensitive registries. Therefore defeating HIPS will become exploit writer's next major task.

The HIPS usually does hook to observe malicious behaviors (No matter in ring0 or ring3). Once it detects a suspicious behavior, it would check 'who' is doing this by identifying the process. If the process is not in its legitimate (white) process list, it could block the action.

Try to imagine, if legitimate process could do things for us, the HIPS would become useless. Doing injection to those processes? No, the injection could be blocked.

We noticed that Microsoft has already provided complete solutions – the WMI and many useful COM objects. By leveraging the technologies, system process could do everything for us, including connecting to Internet, access files/registries, and even installing a MSI file.



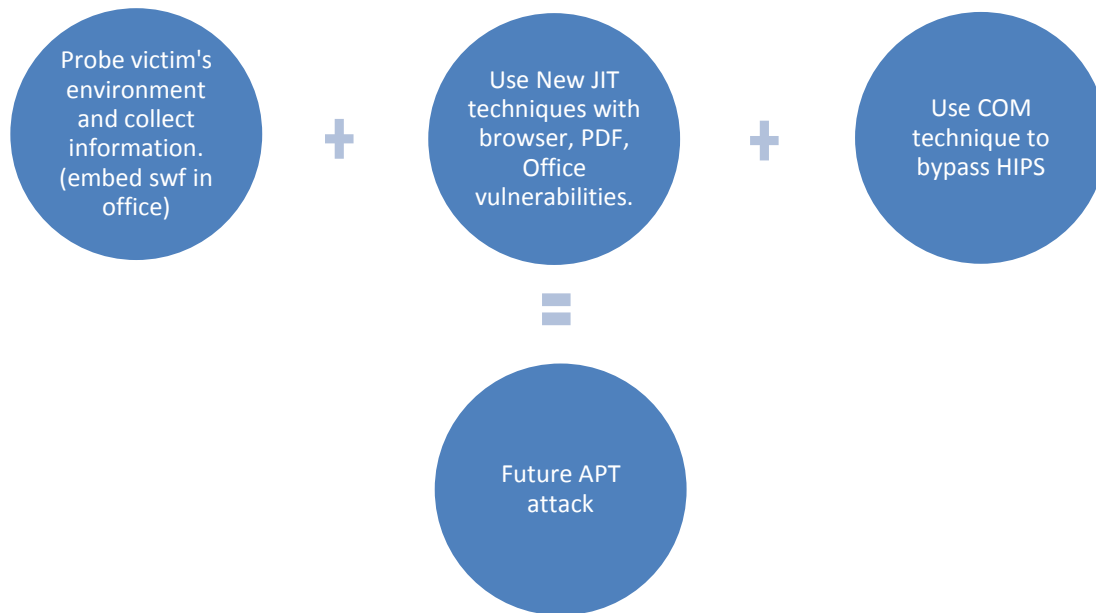
Not only defeating HIPS, the approach could also defeat automation analyzing sandbox system. The malware 'process' actually does nothing directly. The sandbox could record nothing if the sandbox only tracks malware process.

4. Conclusion

We have discussed complete solutions to make a weapon of targeted attack with many new techniques:

- How to find vulnerabilities: AVM fuzzing technique.
- How to defeat exploit mitigation technologies: new JIT spraying.
- How to make an exploit without memory hard work: attack policy flaw.
- How to defeat desktop protection and analyzing system: WMI and COM.

We believe attackers are working hard on these topics. We wish security vendors could address these problems to come out solutions ahead of attackers.



Reference

Office is Still Yummy Ming-chieh Pan and Sungting Tsai, 2010.	http://exploitspace.blogspot.com/2011/06/our-presentation-in-syscan-10-singapore.html
as3compile.exe	http://www.swftools.org/
Adobe Virtual Machine 2 (AVM2)	http://www.adobe.com/devnet/actionscript/articles/avm2overview.pdf
INTERPRETER EXPLOITATION: POINTER INFERENCE AND JIT SPRAYING	http://www.semanticscope.com/research/BHDC2010/BHDC-2010-Paper.pdf
Writing JIT-Spray Shellcode for fun and profit	http://dsecrg.com/files/pub/pdf/Writing%20JIT-Spray%20Shellcode%20for%20fun%20and%20profit.pdf
Enhanced Mitigation Experience Toolkit v2.1	http://www.microsoft.com/download/en/details.aspx?id=1677
swfretool	https://github.com/sporst/SWFREtools
MS11-050 IE mshtml!CObjectElement. Use After Free	http://d0cs4vage.blogspot.com/2011/06/insecticides-dont-kill-bugs-patch.html
Win32_Process Class	http://msdn.microsoft.com/en-us/library/aa394372(v=VS.85).aspx
Bypassing Export address table Address Filter (EAF)	http://skypher.com/index.php/2010/11/17/bypassing-eaf/