Michał Bućko
Senior security specialist
HACK.PL

IE ActiveX-based 0-days basics Demystified

The aim of this paper is not to explain in simple words how to find and exploit an unknown vulnerability in IE. The aim of it is to show the This paper is limited to bugs in ActiveX controls handling only. This document was written for educational purposes only. I am in no way responsible for any irresponsible action taken by the readers.

This is going to be a short article giving the notion of exploitation unknown bugs. Basic knowledge of exploitation and ActiveX is required. Knowledge of code obfuscation is recommended. This paper covers only the basics. I would also like to drop a few lines about my understanding of hacking. If You really want to understand a bit from the Windows OS, this article might be helpful. It is in no way a tutorial how to find a new vulnerability and exploit this.

Firstly, I want to concentrate on the notion of hacking. It is definitely not connected only with security. It has nothing to do with breaking into remote machines. If You want to use this to do something bad, skip it. If You only want to show yourself in the 'hacking world', such thing does not exist. To hack means to crave for knowledge, to admire the beauty of the world, to live and not to look at others. It means much more, If you don't understand it right now, please, don't use this article to act against humanity. Hacker does not want to show himself up in this world, he can stop using computers if he wants, he might have never used one. It is a strange world to learn, to understand it. People come to this world and forget the truth they have in their hearts.

After the important introduction it is time to act. Our task is to find an unknown vulnerability in Internet Explorer. The bugs we are looking for will be based on improper ActiveX controls handling. This article won't teach You everything – take that into consideration. It is hard work.

Lets begin. At first we should take a close look at the available objects, which can be then embedded in the our web site. This means that one can use them this way:

```
<object id="excel"classid="CLSID:0002E510-0000-0000-C000-000000000046">
...
</object>
```

The above code adds an Excel sheet on the exemplary web site. The general syntax is shown below:

```
<object id=target  classid="CLSID:{some_classid}" > </object>
```

But how do we find the vulnerable object?

Well, there are many ways. Some people prepare a fuzzer(or use one!) that goes through all the possible controls (that can be used as embedded objects) and try to crash one by one by the use of the fundamental ways of exploitation. The other people concentrate on the the objective, lets say they want to crash Microsoft Excel Sheet (the aforementioned classid). Concentrating on the objective is, in my opinion, a better way (maybe not faster). One can learn more by trying to exploit an objective as it might a difficult task sometimes. I **highly recommend using your own fuzzers. The software available right now is in no way perfect. Many 0-day vulnerabilities I know that (in some cases) can pass the tests undetected/unnoticed.**

What should I know about the objects?

This is typical:

Class SomeClass
GUID: {some-guid}
Number of Interfaces: how many interfaces?
Default Interface: what is the default interface?
RegKey Safe for Script: F/T
RegKey Safe for Init: F/T
KillBitSet: F/T

The last three factors are significant. Each of them might affect the system's security. Those are the factors I am not going to write about. You should read about those as much as possible with regards to what is going to be said below.

What about crashing?

When I was a child I used to build small castles from Lego blocks. Now it is time to take a close look at the castle. Why is it built on sand? Lets get back to our ActiveX controls. One executes different object's functions with specific parameters (parameter that don't match the required parameter pattern, those parameters shouldn't be rubbish, one should think before doing) as shown below:

```
obj = document.getElementById('target').object;
obj.somefunction("%n%n%n%n%n%n%n%n%n%....");
```

The final part of the exploitation process is described in many papers covering buffer overflow exploitation and shellcoding (knowledge of reverse engineering, debugging, buffer overflows, assembler is required) and I am not going to repeat it one more time. Having our code prepared should we start testing our exploit. We build an exemplary website with our code and check how's it going. It would be also nice if first tests gathered much information about victims and eventual error logs. Error logs are going to help if something (in some cases) goes bad. My experience tells me that one might often overlook a substantial feature that is necessary for the exploit to work correctly. It is quite normal that first exploit applications are going to crash the application remotely without code execution. This must be analysed and the information should be in the logs.

However, being able to exploit the vulnerability remotely is not everything. Our task is to gain the control over many machines. The code must not only exploit the vulnerability but shellcode used should also help us stat undetected in the remote machine for some time. This is mainly achieved by code obfuscation, which makes our code much more difficult to analyze. The specialist (before realizing a patch) must overcome the obstacles connected to the obfuscation. There is a variety of methods that can be used to hide the code.

As the task is to gain control over many machines it is also advised to prepare a special code to be executed and to install a rootkit on the remote machine. Bypassing the AV restrictions should be taken into consideration, new possibilities should be analysed. The AV should in no way be able to detect the attacker.

As we want to exploit the Internet Explorer and hide very deep in the system, should we be cognizant of shellcoding on Windows systems. One must learn about the basics of registry, startup and shutdown procedures and windows services. One should know how to use object viewers, process explorers, file monitors and kernel debuggers. One should also learn about boot configuring in Windows(boot.ini, ntdetect,..), Ntoskrnl, Registry Hive and device drivers. Hiding in the system requires knowledge of execution layers (executive and kernel), creating and deleting processes and threads, memory management and interprocess communication. It is very important to learn more about remote thread injection and virtual memory management. At the time You are versed in the aforementioned Your knowledge will become a very powerful weapon. But, please, comprehension and Understanding (not only computer understanding, understanding is achieved by living, by asking questions and by 'admiring the grass and the tree') is your privilege and you should not use it against other people.

And now just a few things about ActiveX from an attacker point of view. There are many AX controlls, many of them don't present much of a risk. ActiveX are very similar to executable files in Windows, however, can be executed remotely. ActiveX controls are mobile. It must be understood that ActiveX vulnerabilities not only affect Internet Explorer but different software using it. Currently, I have a few unreported 0-day vulnerabilities in AX controls. Those vulnerabilities can be quite easily exploited and affect much software, not only Internet Explorer. I would say that creating a dangerous worm is not a problem for a real blackhat specialist. This is why new solutions are required.

Obfuscation is also worth mentioning. The attacker might try to hide his/her real intentions and make security analysts work harder by obfuscating the code. The code should be as difficult to understand as it is possible. There are many worms that simply collects the code from different places, the code is quite difficult to analyze (not at first sight!). Code obfuscation in future worms can be even made difficult by the advanced use of botnets (mobile and migrating code, analogy to WSN).

Kind regards,
Michał Bućko