The Architecture of Exploitbuster

–

An Exploit Analysis Appliance Framework

by

Florian Rienhardt (peanut@bitnuts.de)

2013/01/05 (version 0.2)

Abstract

Traditional tools like firewalls, anti-virus tools, behavior-analysis systems were designed only for already known patterns of malicious code and attacks. But today we see personalized attacks against cooperate IT-infrastructures and users. Exploitbuster enables you to automaticly open suspicious objects (PDFs, DOCs, XLSs, PPTs etc.) on pre-configurated forensics computers and analyze the system's behavior while processing such objects. Thus Exploitbuster lets you test suspicious contents to see if potential toxic content is harmful. The following whitepaper describes the underlying architecture of Exploitbuster – An Exploit Analysis Appliance Framework.

Table of contents

Illustrations

1 Introduction

One of the first an main questions could be: Why should I use a framework like Exploitbuster? Well, more and more sophisticated and targeted zero-day attacks rise in our internet and computer driven world. Traditional security defenses, such as anti-virus, IDS or next-generation firewalls are not able to keep up with the amount of new attacks flooding computer systems and networks day after day. The impact to organizations is significant: Denial of Service (DoS), increased help desk calls, network downtime, information and intellectual property loss, etc. That all sums up in lost productivity and lost money.

Especially targeted attacks survive most classical detection systems like firewalls, IDS, AVs and filtering internet gateways. Traditional solutions offer only little protection against such attackers, because new attacks (zero-days) have a really good chance getting not detected and nailed by traditional solutions. Attackers search for security hole in commonly used applications like browsers, multimedia- or portable document viewers to bootstrap the process of infecting your system with a trojan or bot.

Exploitbuster enables you to automaticly open suspicious documents (PDFs, DOCs, XLSs, PPTs etc.) on pre-configurated forensics computers. The forensic computers are each packed with different versions of Windows, common used web browsers, plug-ins and applications to open a suspicious file or URL. Using special analyzing tools in user- and kernel-mode Exploitbuster is able to track a lot of suspicious behavior on the fly by keeping the flase-positives/negatives rate low.

Exploitbuster lets you test suspicious contents to see if potential toxic content is really harmful. If so, you can block or trace back such objects. Exploitbuster gives you a powerful environment to inspect potential malware, zero-day attacks embedded through URLs or well known document file formats with little knowledge about exploits in general and in-depth inspection needed.

2 The Model of Exploitbuster

The design of Exploitbuster was kept as simple as possible to decrease the level of complexity. In general the main framework consists of a broker and workers.

The broker manages requests and starts all the analyzing jobs. The worker instances just open an object in their pre-configurated execution environment, log execution information from kernel- and user-mode and then reply the report to the broker.

In the following two section a basic introduction and description will be given on the broker and the worker instances.

2.1 The Broker

The following illustration gives a general overview:

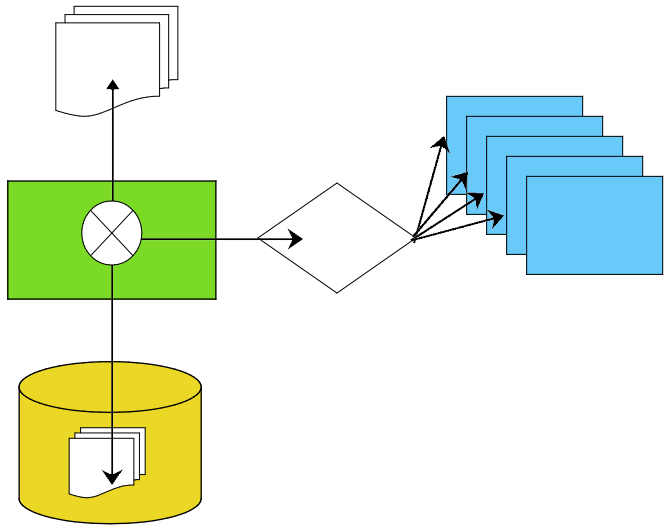

Illustration 1: General overview of the architecture

As described above the broker (green) manages the whole analyzing framework. The broker

- receives objects to analyze (white documents),
- shares these objects across the destination analyzing environments (blue instances),
- waits for the analyzing reports and
- provides and shares reports.

The broker communicates through HTTP, meaning that all requests are provided as POST requests and results will be served through simple GET requests. Both requesting types are encoded with "x-www-form-urlencoded", binary data will be encoded in Base64.

Each request is stored in the framework's global database (yellow). The broker then generates a request specific job containing all relevant information on how the object should be analyzed. The broker then notifies the relevant worker instances (blue) to handle the object in their execution environment.

By using standard HTTP, Exploitbuster can be accessed and managed directly through a web interface:

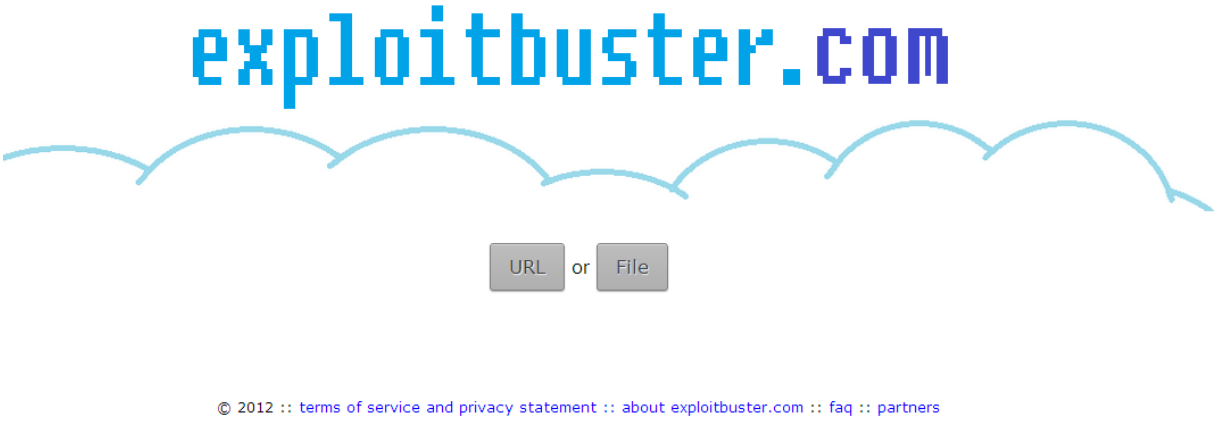

Illustration 2: User interface

On the user's interface there are two possible options to chose from:

i) analyze an URL

ii) analyze a file (or batch of files)

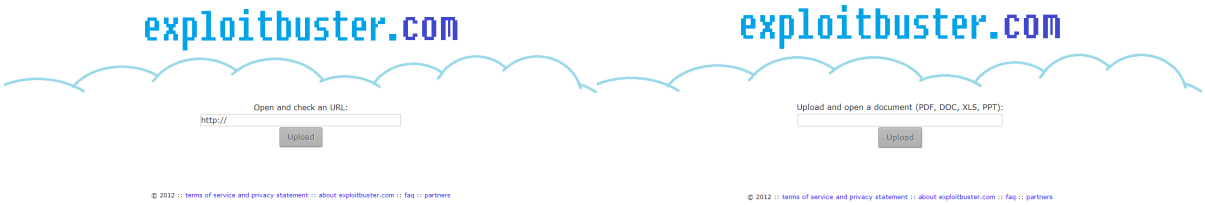

Illustration 3: User interface (open and analyze an URL or file)

Depending on the broker's configuration it handles how to analyze a given input. If you have several worker instances equipped with different operating systems and applications to open and process a given input, the broker manages that an input is distributed and processed by the intended target machines

- A given URL against different versions of Internet Explorer
- A given URL against different versions of Internet Explorer and different versions of Microsoft Windows
- A document like PDF or DOC against different versions of Adobe Reader and Microsoft Office
- etc.

The broker uses a SQL-based database to save an object to analyze. Depending on its mime-type and the configuration the object will be shared across the destination analyzing environments.

As described above, an object will be opened and processed by the corresponding execution environment where Exploitbuster's analyzing tools will monitor and log the machine's activity. Exploitbuster logs general activities like

i) what network connections will be initiated

ii) what registry keys will be altered or newly written

iii) what files will be altered or newly written to the file system

iv) what processes or drivers will be executed

In general, this is no rocket science and logging system activities is old age.

While opening objects there will be a lot of activity from the given list above. Thus it is very tricky to catch suspicious or unintended actions out of a bunch of hundreds or thousands of log entries. It is Exploitbuster's added value to automatically strip out suspicious behavior of such a bunch. If Exploiutbuster rates an objects's run time actions as too obscure, a given object will be marked as potential dangerous/suspicious.

To accomplish that, Exploitbuster automatically checks

i) what executables will be written or run in a worker instance,

ii) what well-known suspicious and bad Win32 API-calls are used,

iii) what network connections are fired up and

iv) what registry keys are written or altered.

This is done in kernel- and real-mode. This makes it harder for an attacker to bypass the analyzing framework e.g. by directly calling system functions that might not be hooked in real-mode. On the other hand Exploitbuster uses special filtering drivers that use randomization tricks to avoid getting located by an attacker trying to disable the framework's kernel-mode part.

To get around a given worker instance, an attacker must bypass more than just one analyzing approach and thus an attacker must use more than one exploit to come around. Clearly spoken: There is not just one user-mode application or hook to bypass, there are several and there are more in kernel-mode. And even with privilege escalation it is hard to bypass the in-depth monitoring.

Because exploits and especially zero-days are often very small in size (meaning their executable footprint) attackers do not have a lot of code that could be used to detect and then avoid getting detected or to disable a detection system. Well, thus it is very likely that Exploitbuster will detect suspicious actions that took place while processing an object.

2.2 Reports

The following snippet shows some part of the textual feedback that a worker instance gives back to the broker:

```
? Executable written **\Windows\Temp\SDIAG_b22f64df-aaa4-4802-a98b-436046be220b\de-
DE\DiagPackage.dll.mui
! Executable written **\Windows\Temp\SDIAG_b22f64df-aaa4-4802-a98b-
436046be220b\DiagPackage.dll
? Executable memory from **\Users\*****\AppData\Roaming\Microsoft\Windows\Start
Menu\Programs\Internet Explorer.lnk
? Executable memory from **\Users\Public\Documents\desktop.ini
? Executable memory from **\Users\Public\Videos\desktop.ini
? Executable memory from **\Windows\Media\Windows User Account Control.wav
? Executable memory from **\Windows\Microsoft.NET\Framework\v4.0.30319\CORPerfMonSymbols.h
? Executable memory from **\Windows\Microsoft.NET\Framework\v4.0.30319\corperfmonsymbols.ini
? Executable memory from **\Windows\Microsoft.NET\Framework\v4.0.30319\mscoree.tlb
? Executable memory from **\Windows\Microsoft.NET\Framework\v4.0.30319\mscorlib.tlb
! Registry key written to **[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
"systemoverview"="c:\\window\\system32\\rundll32.exe c:\\windows\\system32\\xhook.dll"
! Executable run **\downloads\ofw_vs10o.exe
! connection to 127.0.0.1
```

As you can see Exploitbuster is able to detect if a executable file was written to disk (see ? Executable written). If such an executable image is unknown to the target system it will be rated as more suspicious and thus indicates that such a file contain malicious code. Exploitbuster is also able to detect if certain parts of memory are registered and initialized for executable content (see ? Executable memory from). Such memory might be subject of an attack if the caller is unknown or untrusted. As seen in many exploits and other malware it is a very good indication to check the originator of a call to executable code pages, because

- it is likely that the caller itself is a mad behaving application (due to an exploit) or
- it is already the fire-it-up sequence of an malicious application

If you detect code in such an early stage it is easy to block and even trace back malicious code and it is even more probable that you can catch the originating source of an attack.

3 Hard- and Software requirements

Exploitbuster's target audience is the typical Microsoft Windows-based IT environment. Licensed versions of Windows XP, Vista, 7 and 8 (x86 and x64 versions). A VMWare Virtualization solution is recommended and supported but VirtualBox works fine too.

An instance of Exploitbuster can be installed on a standard PC with a x86/x64 Intel/AMD-based CPU, 1 to 4 GBs of RAM and at least 20 GBs of HDD. But I highly recommend to install Exploitbuster on some Appliance-Server with VMWare as a virtualization host where you can run different Exploitbuster-instances in parallel.

Exploitbuster runs with a very, very small binary footprint and can be run on <= €100 machines if one does not like virtualization to detect malware. Thus if you like, you can install and run the framework on a bunch of cheap mini/micro-ITX boxes.

It depends on the use case, namely on your IT infrastructure and potential targets, what versions of Windows you would like to support. E. g. if your infrastructure consists only of Microsoft Windows 7 clients you just need one instance of Exploitbuster. You might run more than one instance of the same pre-configurated system for performance reasons.

4 FAQ

Can I use Exploitbuster as some kind of protections suite (like a desktop firewall or something)?

- No, Exploitbuster is intended to run as a server and its analyzing workers as separate machines, not on one client.
- You might install it on only one client and especially use its drivers as a background service which might provide additional protection, but that is not the intended way using Exploitbuster.

Is Exploitbuster some kind of anti-virus scanner or desktop firewall?

- No, Exploitbuster is an analyzing tool. It natively will not scan files or URLs against well known virus databases or against a collection of commercial anti-virus products. Exploitbuster's aim is to dectect unknown or permutated known attacks hidden in documents or web sites that will not be catched by ordinary anti-virus scanners. But it is no problem to include an anti-virus scanner into the framework and

pre-scan suspicous files before checking them on an Exploitbuster-instance.

I have a suspicious object (file, url). Can you run it through your instance of Exploitbuster?

- For sure, but FIRST we MUST find some kind of mutual agreement regarding the terms of service and regarding your privacy, responsible disclosure etc. Feel free and contact me.

How much is the Exploitbuster Appliance and where can I buy it?

- Well, Exploitbuster is a private project and currently I do not plan to sell it as a product. If there is demand I can assist interested parties on how to install and setup an Exploitbuster Analysis Appliance. Feel free and contact me.

Do you publish any exploits detected by Exploitbuster?

- Hell no! Well, from time to time we do active Internet measurements using Exploitbuster, but we will not publish any exploits to the public detected by Exploitbuster. If needed and possible we will contact the author of a vulnerable object or the operating authority under responsible disclosure. Maybe we will provide some high level statistics soon.

Is the current version of Exploitbuster final, do you update and elevate Exploitbuster?

- No and yes. Exploitbuster is a private project and we are still developing its code and hardware base. Check out the project's web site and stay tuned.

Will Exploitbuster catch all kinds of exploits?

- Well, we hope that Exploitbuster will catch a lot of exploits but to say we will catch all expolits seems to be an illusion. But we work hard to provide such an mirage.