Malicious PDF Documents Explained

•••

Didier Stevens

PDFs

PDF (Portable Document Format) is a widely used file format used to package everything from research papers to restaurant menus.

It’s a pretty complicated file format, so there is a large attack surface.

Some of the features in the PDF format can be used for malicious purposes, and are (somewhat) indistinguishable from legitimate behavior.

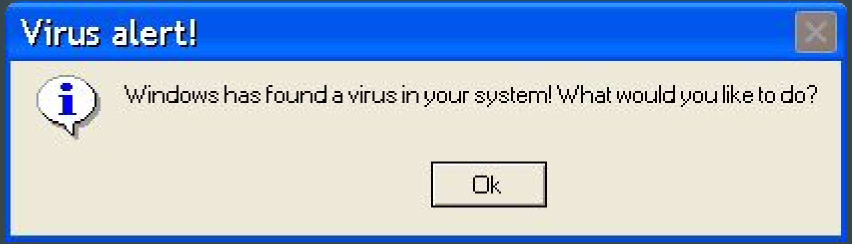
Thankfully, these attacks are limited and generally require the user to blindly click ‘ok’ on a few dialog boxes.

Legitimate code, malicious behavior

Files can be embedded in a PDF, and then opened after a prompt presented to the user is clicked on.

There is nothing in the format that stops you from embedding an executable, but most PDF readers will refuse to extract it.

However, not all forms of executable code are blacklisted, but the user still has to approve the execution. (example: a python script)


Virus alert!

Windows has found a virus in your system! What would you like to do?

Scripting PDFs

The PDF format allows to use Javascript to add input validation to forms, and other similar tasks.

The Javascript interpreter is limited and cannot interact with the host system, but many vulnerabilities have been found in this interpreter.

There was a vulnerability in Adobe Reader’s util.printf Javascript function, which could be tricked into executing whatever input you gave it as native code.

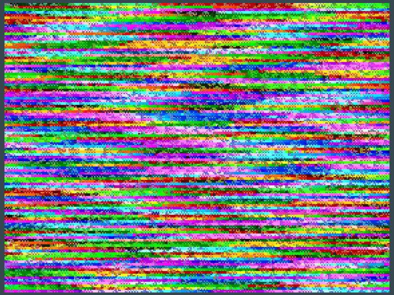
The exploit could be set to run when the document is opened, silently infecting the user’s system.

Malformed Images

PDFs can contain many different kinds of images, so PDF readers have to have a lot of code to deal with all the different formats.

In 2009, a vulnerability was found in the JBIG2Decode decompression algorithm that allowed for arbitrary code execution if Javascript was enabled.

Again, this exploit happened as soon as the document was opened.



Heap Spraying

With many PDF exploits attackers cannot completely control where EIP will go, so they use a technique called Heap Spraying to maximize the probability that their shellcode will be executed.

In Heap Spraying, a large NOP sled with the shellcode appended to it (sometimes up to 1MB in size) is copied over and over into the Javascript heap, so that almost any given address in the heap will point to the shellcode.

This takes a while, so a PDF reader will freeze while the heap spray is happening. This can be an indicator of compromise.

Typical Payloads

Most PDF exploits do not execute their true payload during the exploit - most either download a new payload from the internet or extract a payload from the PDF.

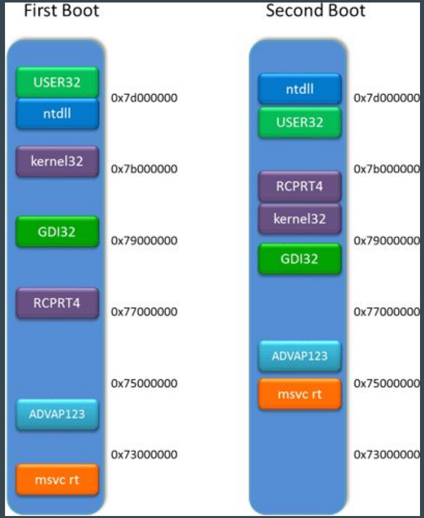
Depending on the exploit, the PDF reader may or may not crash after the exploit.

The actual payload can be anything from a worm to ransomware.

Mitigations

Disabling Javascript won't usually stop the exploit from running, but it will stop the heap spray. The reader will crash but no malicious code will run.

Turning on DEP, ASLR, or using an application sandbox can also stop the exploit from successfully executing malicious code.


First Boot
Second Boot
USER32
ntdll
kernel32
GDI32
RCPRT4
ADVAP123
msvc rt
0x7d000000
0x7b000000
0x79000000
0x77000000
0x75000000
0x73000000
ntdll
USER32
RCPRT4
kernel32
GDI32
ADVAP123
msvc rt