Intercepting Windows Printing by Modifying GDI Subsystem

by Artyom Shishkin,
Positive Technologies
Basically it’s a data source for
  • Monitoring systems
  • DLP solutions
What do we have?

- `FindNextPrinterChangeNotification()`: 
  - Printer name
  - Timestamp
  - Job status
  - Pages count

  Print provider is the source of this info, so I wouldn’t rely on it too much.
Print providers send jobs to a local or a remote machine

A print processor converts the spooled data into a format suitable for a print monitor

The print monitor passes the data to a port monitor

A port monitor is an interface between the usermode and the kernelmode parts of the printing system

What a mess!
A set of Spooler service functions, which serve as wrappers for driver components

At this level, we can only get the spooled data
  * This is a level of raw printing

Try to parse this data
The same set of functions used for Windows graphics

A printer is a device context suitable for GDI drawing functions

- hPrinter = CreateDC('SuperLaserJet', params);
- StartDoc(hPrinter);
- TextOut(hPrinter, ‘Text’);
- ...

Graphical data is Windows graphical data – NT EMF format
Inside GDI

* Found with the help of PEB
* Thanks to Feng Yuan

Process Address Space

GDI shared handle table

GDI cell
- Object kernel address
- Selection count
- Process ID
- Upper handle value
- Object type
- Usermode info
The trick

hOriginalPrinter

hPrintInterceptor

Shared handle table

Cell contents

Magical operation

Cell contents
Profit

- Swap GDI cells to send documents to a fake printer
- It is not always necessary to create your own virtual printer, you can use something like Microsoft XPS Writer
- The intercepted image can be easily forwarded to the original printer
The concept

Application wants to print things

<table>
<thead>
<tr>
<th>DLL: It’s using GDI, I’ll load here using windows hooks and patch some GDI functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateDC()</td>
</tr>
<tr>
<td>StartDoc()</td>
</tr>
<tr>
<td>EndDoc()</td>
</tr>
<tr>
<td>DeleteDC()</td>
</tr>
</tbody>
</table>

I’ll save the original parameters of this printing request

Hey, you’ve decided to print! I’ll swap the GDI cells so that you use the old handle for a new device

Okay, done here, I’ll print your document on the real printer

Let’s clean everything up and make things look like they did before
Sample implementation

- **Settings file**
  - Dll used for function interception
  - GUI-based application that is supposed to print something
  - .xps image

- **Kernelmode magic performer**
- **Application that fulfills the original request**
Thank you!

Any questions?