I am writing a pdf parser and I have been able to read the data into my own custom structure and am also able to write it back as a pdf file I have also implemented functions for making small edits like increasing the page size adding and deleting pages changing fonts etc. Right now my goal is to render the pdf into a texture to display in imgui since it is my gui framework of choice. I already have my font glyphs for each char loaded. For the each page I want to render I go through each character and get the glyphs of the fonts character then render it. As of now my problem is the scaling of the pdf. I want to just use 1 pdf unit as 1 unit in my vertices coordinate data a 1:1 to make things simple I render my fonts with freetype which allows you to set the number of pixels for the width and height of the glyphs I want to use 48 for sharpness but I have noticed that every pixel setting I make has its own vertice cord that it will occupy I want to know the relationship between the coord and the pixels and how I can render my things proportionally because when I set my glm ortho to the size of the page the font looks out of proportion too large I need to scale it to proportion.

While I understand that this is for educational purposes, I want to emphasize that the techniques discussed here can be dangerous if misused. PDF exploits are serious security concerns and can be misused for malicious purposes. That being said, I'll provide an overview of how malicious actors can exploit PDFs for educational and security awareness purposes only.

How Do Hackers Execute Code from PDFs?

Hackers can use several techniques to execute arbitrary code from PDFs. These techniques leverage vulnerabilities in PDF readers or embed malicious actions into the PDF file itself.

Here are some common methods:

1. JavaScript Execution in PDFs

PDFs support JavaScript, and this can be used to trigger malicious actions. A PDF might contain malicious JavaScript code that runs when the PDF is opened or when a user interacts with it (e.g., clicking a button or annotation).

Example:

pdf

Copy

Edit

/JS (app.alert('Hacked!'))

When opened, this will show a popup, but malicious code can execute hidden operations.

Attackers can craft JavaScript that attempts to exploit vulnerabilities in the PDF viewer.

How It Works:

The JavaScript is embedded within the PDF.

It can trigger malicious payloads, such as downloading files or executing scripts on the system.

Real-World Example:

A well-known Exploit (CVE-2018-4990) targeted Adobe Acrobat by embedding a JavaScript payload in the PDF. This allowed arbitrary code execution by triggering vulnerabilities in how the PDF reader handled JavaScript.

2. Embedded Files (with Exploits) PDFs can contain embedded files, such as executables or scripts (like .exe or .vbs). If a user is prompted to open these embedded files, the code can be executed.

Example:

An attacker can embed a malicious .exe file inside the PDF and trick the user into executing it.

How It Works:

The file is embedded into the PDF.

Acrobat or other PDF readers may not block the extraction of the embedded files, allowing them to be executed.

Real-World Example:

In 2013, hackers used embedded files in PDFs to distribute malware like Zeus and RATs (Remote Access Trojans). The user would open the PDF and then download and execute the malicious file.

3. Exploiting Vulnerabilities in PDF Readers

Many PDF readers have security flaws that can be exploited when opening a specially crafted PDF. These flaws are often zero-day vulnerabilities, which are unknown to the software vendor at the time.

Common Exploits:

Buffer Overflow: By sending a crafted PDF with a buffer overflow, the attacker can overwrite memory and execute arbitrary code.

Heap Spraying: Exploiting heap management bugs to inject and run malicious code.

How It Works:

The attacker sends a PDF with an exploit targeting a specific vulnerability in a PDF reader (e.g., Adobe Acrobat).

When the PDF is opened, the vulnerability is triggered, allowing the attacker to run arbitrary code.

Real-World Example:

The CVE-2010-0188 vulnerability in Adobe Reader was an example of an exploit that allowed remote code execution by crafting a malicious PDF that exploited a flaw in how Adobe Reader processed certain objects.

4. Malicious URLs and Phishing Links

Hackers can use PDFs to launch phishing attacks by embedding malicious links or URLs in the document. Clicking these links can redirect the user to a malicious website that may attempt to install malware or steal credentials.

Example:

pdf

Copy

Edit

/URI (http://malicious-website.com)

When clicked, the URL takes the user to a malicious page that might attempt to exploit the browser or download malware.

How It Works:

The PDF contains a link or embedded action to open a malicious URL.

When clicked, the link leads to a page that automatically executes a script or downloads a malicious file.

Real-World Example:

In 2017, PDFs were used in phishing campaigns that led to malicious sites attempting to install Trojan malware or steal personal information.

5. Social Engineering in PDFs

Social engineering attacks often use PDFs to trick users into running malicious code or revealing sensitive information. These attacks may use fake alerts or lures that appear to be from legitimate sources.

For example:

Fake invoices or shipping notices may be embedded in the PDF, asking users to click a link or open a file.

These actions can trigger malware downloads or exploit vulnerabilities.

How It Works:

A PDF might appear to be a legitimate document (e.g., invoice, contract, or form).

It may ask the user to click on a link or open an attachment, which then leads to malicious actions.

Real-World Example:

Business email compromise (BEC) campaigns have often used PDFs to disguise malicious attachments or links that lead to malware downloads.

6. Abusing PDF Features to Hide Malicious Code

Some attackers exploit less obvious PDF features, like hidden layers or JavaScript objects, to hide malicious payloads. This method may avoid detection by security software or the user.

Invisible scripts or obfuscated code could be used to execute commands silently without any visible indicators.

Key Takeaways:

JavaScript in PDFs can execute code when the PDF is opened or when interacting with annotations.

Embedded files can contain executables that the user may accidentally run.

Exploiting vulnerabilities in PDF readers allows attackers to execute arbitrary code on the system. Phishing URLs and social engineering are often used to mislead the user into downloading or running malware.

⚠ Prevention Measures

Never open PDFs from unknown sources or those that seem suspicious.

Disable JavaScript in PDF readers (if possible).

Keep PDF readers up to date to patch security vulnerabilities.

Use trusted antivirus software to scan files and websites.

This overview is aimed at promoting security awareness and understanding how attackers can exploit PDFs. If you're developing or working with PDFs, it's critical to always stay aware of security best practices to protect yourself and others from these threats.