WHITE PAPER

How to Block Malicious File Uploads

OPSWAT.

Introduction

Although file uploads are necessary for employee productivity and for certain websites and web applications to perform their functions, they also offer an attack vector to cyber criminals. By concealing advanced threats that exploit vulnerabilities within common file types, attackers can compromise an end user or an entire system. This white paper will describe the best ways to prevent malicious file uploads while still allowing legitimate users to upload files.

Published February, 2021

SECTION **1.0**

The Problem of File Uploads

In order to keep a business running properly, you need to share files with employees, partners, and customers. File uploading is usually done by the departments that often handle sensitive data – e.g., accounting, HR, legal, and marketing.

However, users no longer need to install a rogue application in order to get infected – that can happen by opening what appears to be a resume, an invoice, a courier receipt, or any other productivity file. Also, attackers can benefit from poor input validation or even vulnerabilities in server-side processing solutions. They can easily gain control of the environment by either retrieving access keys (e.g. SSRF attack using AWS EC2 metadata endpoint) or RCE based on a vulnerability.

Document-borne malware is on the rise.[1] Therefore, any file coming inside an organization should be audited and analyzed, even when the sender seems to be a trusted, reliable source.

To ban file uploads altogether would be impractical. Clearly, it is necessary to make file uploading and importing more secure for businesses to function efficiently.

Productivity Files and File Uploads

The most commonly uploaded and shared files in office settings are:

Microsoft Office
DOC(X), XLS(X), PPT(X), etc.

Images
JPEG, PNG, TIFF, etc.

PDFs

Most day-to-day activities rely on these file types, and at first glance, they seem harmless. However, advanced features in these file formats can be exploited by attackers. Most people are aware of malicious macros, but Microsoft Office documents can contain many other kinds of advanced threats as well. For example, OLE objects disguised as embedded multimedia or script-enabled ActiveX controls can be configured by attackers to download malicious payloads.[2, 3] PDFs may contain JavaScript that performs malicious actions.[4] Images can carry malicious JavaScript.[5]

Additionally, malicious files can be disguised as one of these file types.[6] These are called “spoofed” files. There are several methods of concealing the true type of a file from users, and even from anti-malware security measures.

SECTION **2.0**

Best Practices

Every organization has different workflows and different security needs. When designing a strategy to keep productivity file uploads secure, it's important to assess your unique situation.

Common Assessment Criteria

Start by asking questions like

- How many restrictions can you add without impacting productivity?
- How much can you rely on user training? How confident are you that users will actually apply everything they learn in security training?
- How are you validating the files? Are you doing any pre-processing before you are making the files available? If so, how do you ensure they won't try to exploit the processing service?
- How confident are you that a simulated sandbox environment will replicate your live environment?

Also consider your use case

- When and why do users need files uploaded on your portal?
- What filetypes are used or needed?
- What are the risks of allowing those files to enter your organization?

If you are simply receiving scanned documents or resumes, collaborating with partners on draft agreements, or sharing invoices or POs

- Why would you allow a PDF with embedded JavaScript?
- Are you sure you can trust a document that contains hyperlinks, macros, OLE objects, or ActiveX controls?
- How do you know if an image is legitimate and hasn't been created by an attacker?

It's one thing to decide that any file containing scripts or macros should not enter an organization; it's another thing to enforce that policy. It is difficult to determine exactly what a file contains without opening it.

Therefore, further steps are necessary to defend against malicious files disguised as common productivity files.

Best Practices

Only allow certain types of file formats.

This is a simple but necessary step. Avoid unnecessary risks by filtering the types of files that enter your organization. The idea is to block any file that will not impact your team's productivity in order to avoid unnecessary risks. Make business-driven decisions about which kinds of files employees and users need, and which kinds are unnecessary. Doing so will only eliminate a small part of the risk of malicious file uploads, but it's a start.

Block unnecessary file types, disguised files, and spoofed files.

Identifying and verifying the true type of a file is complicated. A lot of file verification solutions rely on merely reading the file extension. This is more dangerous than not having a solution in place at all, since users will expect that any file that comes through is safe to open. In fact, that's not true – faking the true file type is a very common method of hiding malicious software, and many hackers use this technique.

Additionally, with the simplified interfaces of contemporary operating systems that don't display known file extensions, it's even easier for a spoofed file to hide in plain sight.

It is essential to find and implement a solution that can identify the true type of a file even when it is disguised.

Prevent DTD attacks.

Complex files allow Document Type Declaration, which are prone to XXE attacks, DoS, SSRF, and other risks. Insecure Deserialization is part of the OWASP Top 10 Web App Vulnerabilities. Sanitizing the content to remove any potential callbacks or data exfiltration crafted content is critical in preventing these kinds of attacks.

Restrict active content usage.

Now that you've identified which filetypes you will allow, make sure you define what will be allowed to be embedded in those files. It's meaningless to limit allowed filetypes to PDF, but not to verify what the PDF contains. Having a PDF which contains a Word document attachment which has a macro embedded is no better than allowing macro enabled documents from the start. Similarly, hackers can craft file uploads to exploit the systems parsing the files. You need to understand the scope of the application, which files are being shared, and what content is expected to be embedded in those files.

Don't make the exception a rule.

For instance, if only the design team needs to use and upload PSD and AI files, set customized rules for them rather than allowing everyone to use those file types. Keep the general allowed file types set to a minimum.

Set up security policies that exceed the bare minimum.

This may involve creating a custom solution for your application or organization. The best approach is to integrate with third-party anti-malware scanning software so that all file uploads are scanned for malware, and all files containing malicious content are detected. An anti-malware integration of this kind would require the use of anti-malware APIs. Advanced threats require more advanced preventative measures.

SECTION **3.0**

Four Steps for Managing Secure File Uploads

If file uploads are essential for your business to run, there are four steps you need to take in order to ensure that no infected, malicious, or otherwise compromised files are uploaded into your system.

1. Integrate with Antivirus APIs

The first and most important step to take is to scan all file uploads for known malware.

The workflow should be clear and simple: Any uploaded file should be analyzed for known malware, and the file shouldn't be made available to end users until the analysis is complete.

Depending on the API integration and the volume of files being uploaded, this can cause a delay that varies from a few hundred milliseconds to minutes. This might be a drawback for some businesses, so align your business needs to your implementation – if you need the files analyzed in a fraction of a second, you need to rely on an antivirus API integrated directly into your infrastructure.

For such an integration, you can either integrate directly with an antivirus API or use third-party solutions that expose an API on top of the antivirus API. There are a few solutions available that utilize one or multiple antivirus APIs, but the best tactic is to analyze uploaded files using multiple anti-malware engines. For better detection rates, use a multi-scanning solution instead of a single antivirus API.

2. Always Sanitize Productivity Files

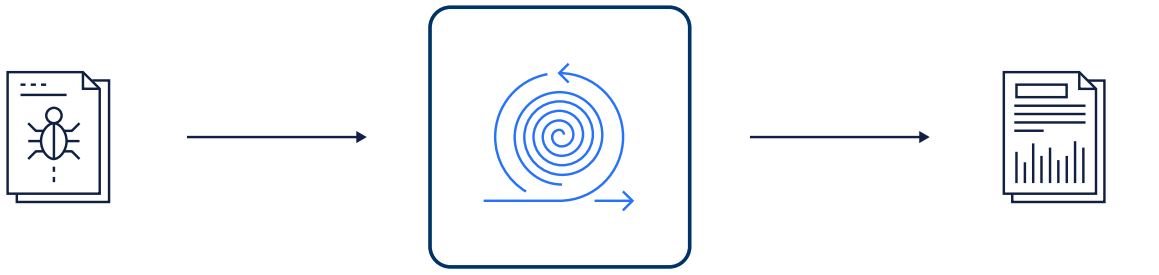
As discussed above, more and more attackers are using common file types to deliver targeted malware attacks. Malware, especially ransomware, can infect devices by using embedded objects such as scripts and macros in common file types used for productivity.

Macros were always a pain point in fighting against malware, and now there are even more advanced threats. It is hard to discover all known and unknown threats that may be present within files by using static analysis. The content of the file itself is usually not malicious, however, an embedded object can perform malicious behavior by downloading a malicious payload and getting access to the PowerShell.

Static analysis can be supplemented with dynamic analysis, but a sandbox environment will never perfectly match a user's real-time environment,[7] so malware evasion techniques don't have to be particularly advanced to evade detection.[8]

That's why the best solution for blocking these kinds of attacks is sanitizing them. The technology that was built exactly for sanitizing complex file formats is Content Disarm and Reconstruction (CDR).

What does “sanitizing files” mean?



Identify

Assume all files are malicious

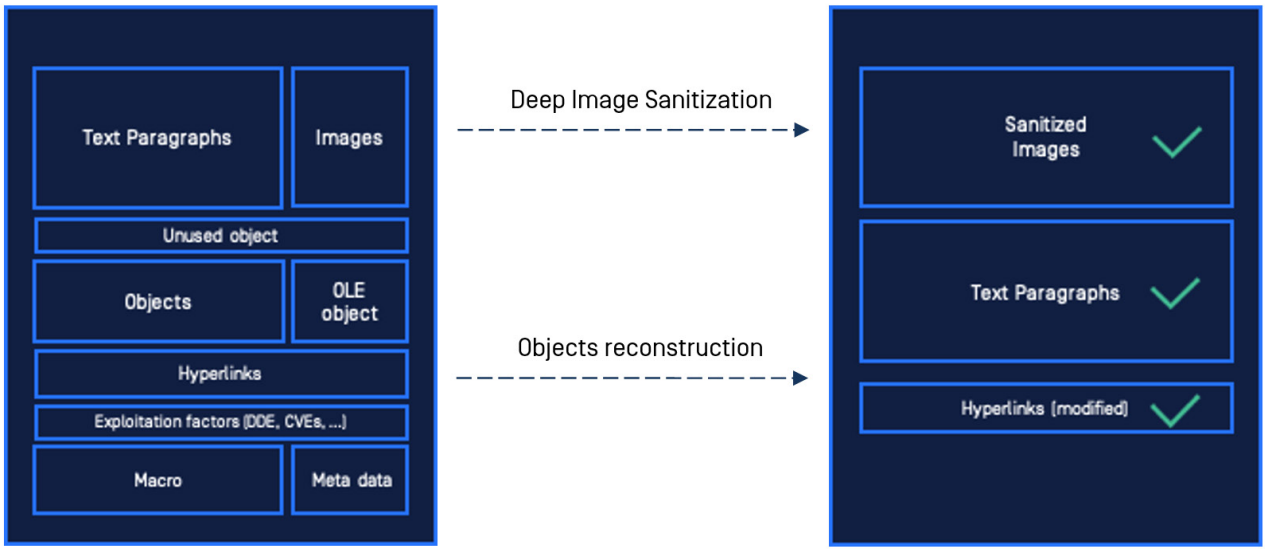
Disarm

Remove potential threats such as macros, scripts, and hyperlinks

Reconstruct

Rebuild files fast and convert file

Content Disarm and Reconstruction means taking a productivity file, breaking it down into small parts, and analyzing each part separately. For instance, any Word document can contain embedded images, links, tables, and so on. Each of those objects should be analyzed and sanitized independently as well.



As an example, if you allow resumes to be uploaded through your portal, a PDF with embedded JavaScript should be considered suspicious. Stripping the embedded JavaScript will remove the potential threat while keeping the PDF content and usability intact.

Content Disarm and Reconstruction can be considered a remediation tool that will allow you to accept productivity files while eliminating the risk of hidden malicious content.

With OPSWAT API offerings, data sanitization can be incorporated into any organization's internal solution for handling file uploads.

3. Extract Archive File Uploads

Archives are often necessary for productivity since they are a very common way of sharing multiple files at once. For this reason, system administrators may need to allow archive file uploads.

Unfortunately, files that ordinarily would be blocked could be hidden in an archive. Or worse, malicious actors could send a crafted archive that turns out to be an archive bomb.

To eliminate these threats, the archive's recursion levels should be limited, along with the number of files in the archive. For each file in the archive (iterate through all the levels), start scanning and sanitizing the files within.

These steps are difficult to implement unless you're able to use APIs for a customized solution. OPSWAT anti-malware APIs allow for this type of archive handling.

4. File Type Verification

Malicious files are often disguised as other file types. For instance, EXE files may be disguised as TXT files.[9] To prevent malicious executables or other spoofed files from entering a system, it is essential to identify and block files with incorrect extensions.

SECTION 4.0

How OPSWAT Helps

To summarize the information covered so far, it is crucial to follow these four steps to protect systems from malicious file uploads:

01. Scan files with one or more anti-malware engines
02. Sanitize files via Deep CDR
03. Extract archive files
04. Verify file types

MetaDefender

MetaDefender is a fully automated malware prevention system which provides innovative Content Disarm and Reconstruction (Deep CDR), vulnerability assessment, and multi-scanning technology. MetaDefender analyzes any kind of data and blocks, repairs, or recommends patching based on the findings.

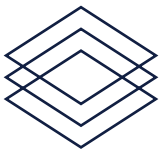
MetaDefender Platform

MetaDefender APIs

OPSWAT offers a rich set of REST APIs that system administrators and developers can use to integrate dynamic security features into their security architecture. These APIs can integrate with existing security features on file upload servers to completely block malicious file upload attacks.

MetaDefender anti-malware API integrations enable file upload security solutions to perform all four steps. A summary of how the solution works along with some technical details on how you can use OPSWAT APIs.

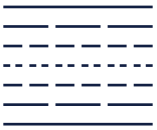
MetaDefender offers a REST API integration for:



01. Multi-scanning

Scan files for known and unknown malware (AI, signature and heuristic scanning)

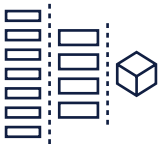
A direct and simple way to integrate with the most reputable anti-malware solutions by leveraging antivirus APIs



02. Deep CDR

Remove any embedded objects that might be malicious from productivity documents

Sanitize documents “hidden” in an archive



03. Archive Extraction

Block archive bombs

Increase malware detection ratio by opening archives and analyzing each file individually

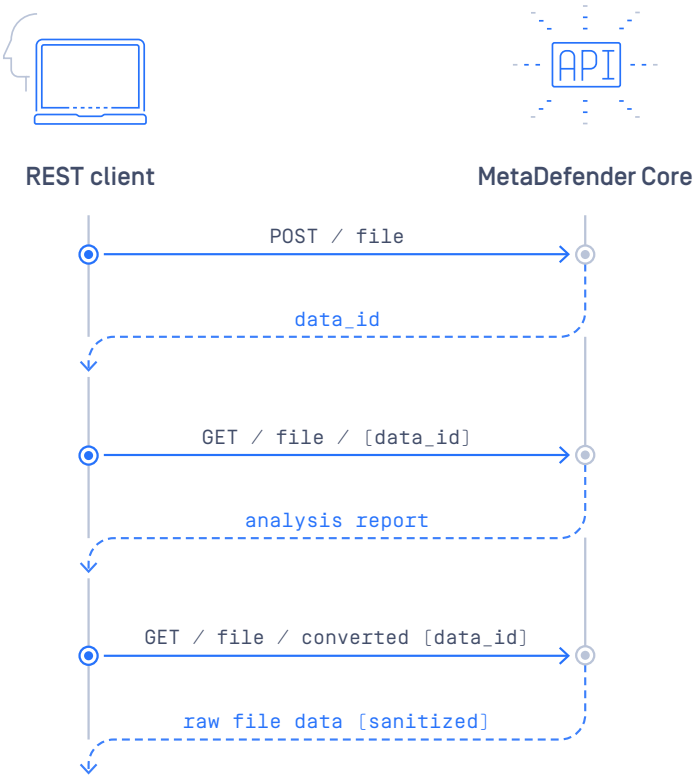
04. File Type Verification

Detect the true file type of a file

MetaDefender API Integration for File Uploads: Instructions

Analyze a file:

1. Submit a scan file request by calling the `/file` API endpoint
   - A. The process is done asynchronously and each request will be parallelized
   - B. The JSON response will provide a data_id, which is the unique identifier for the original request
2. Retrieve scan report
   - A. The scan report should be requested based on the received data_id
   - B. The request will be done to `/file/{data_id}`
     - a. The scan report will return partial data until the file is processed
     - b. Keep pooling until `scan_results.progress_percentage` is 100
3. Always check process_info node
   - A. The `process_info.result` will provide data on status of the file: allowed/blocked
   - B. Always check `process_info.post_processing` node
     - a. This will provide insights about the additional steps defined in the workflow
     - b. This node should be used to retrieve the sanitized productivity document



The integration flow is detailed in Appendix A.

MetaDefender Features

Environment

- Support for:
  - Windows 10
  - Windows Server (2008, 2008R2, 2012, 2012R2, 2016, 2019)
  - Linux
    - CentOS 7.0+, RedHat Enterprise 7.0+
    - Debian 9.0+
    - Ubuntu 16.04, 18.04
- Support for VMWare and VirtualBox
- Cloud deployments: AWS, Azure, GCP
- Container Support: Docker
- Cloud integration available via MetaDefender.com

Engines

- Over 30 anti-malware engines
- Over 100 filetypes supported for sanitization
- Over 30 supported archives
- Over 15,000 supported applications by the File-based Vulnerability Assessment

Integration

- REST API for automated integration into your workflow (HTTP/S)
- Leverage official integrations by 30 technology partners
- STIX export available for MetaDefender Core
- Threat intelligence feeds available for MetaDefender.com

Deployment

- Online or offline environments
- Air-gapped environments supported by MetaDefender
- Thousands of deployments worldwide, a third of which are in offline environments
- Remote assisted installations and workshops

Special Features

- Sanitize specialized productivity documents (JTD, HWP, XML)
- Notification of vulnerabilities within applications before installing them
- Protection from spoofing attacks and archive bombs
- Optimized archive scanning and sanitization

More Information

Sample reports are available:

- Weekly multi-scanning efficiency for top threats statistics MetaDefender.com/reports/statistics
- Data sanitization reports MetaDefender.com/reports/sanitization
- Outbreak reports MetaDefender.com/reports/outbreak

SECTION **5.0**

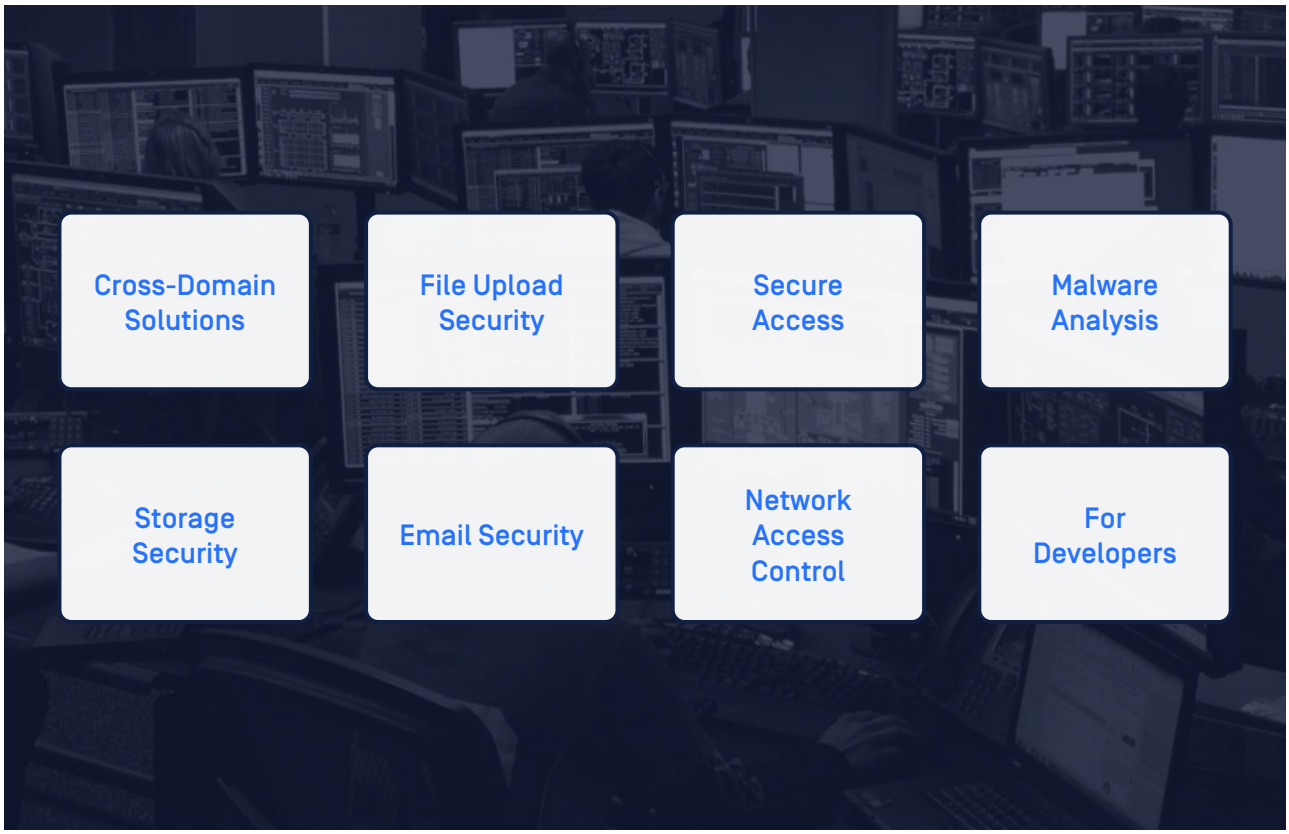
Conclusion

File uploads are a major potential threat vector for any organization. However, there are concrete steps that organizations can take in order to mitigate this threat vector. An API integration is the most efficient and effective way to implement a file upload security solution, and multiple resources are available for those who wish to do so.

SECTION 6.0

About OPSWAT

OPSWAT is a global leader in critical infrastructure cybersecurity that helps protect the world's mission-critical organizations from malware and zero-day attacks. To minimize the risk of compromise, OPSWAT Critical Infrastructure Protection (CIP) solutions enable both public and private organizations to implement processes that ensure the secure transfer of files and devices to and from critical networks. More than 1,500 organizations worldwide spanning Financial Services, Defense, Manufacturing, Energy, Aerospace, and Transportation Systems trust OPSWAT to secure their files and devices; ensure compliance with industry and government-driven policies and regulations, and to protect their reputations, finances, employees and relationships from cyber-driven disruption. OPSWAT. Trust no file. Trust no device.



Visit us on LinkedIn, Twitter, Facebook, and YouTube.

Contact us at opswat.com/contact for pricing information, evaluation accounts, technical presentations, or to request a quote.

SECTION **7.0**

Endnotes

1 Microsoft TechNet, "Before you enable those macros...": https://www.microsoft.com/security/blog/2014/12/30/before-you-enable-those-macros/

2 Microsoft TechNet, "Where's the Macro? Malware authors are now using OLE embedding to deliver malicious files": https://www.microsoft.com/security/blog/2016/06/14/wheres-the-macro-malware-author-are-now-using-ole-embedding-to-deliver-malicious-files/

3 Microsoft TechNet, "Behavior of ActiveX controls embedded in Office documents": https://blogs.technet.microsoft.com/srd/2009/03/03/ behavior-of-activex-controls-embedded-in-office-documents/

4 InfoSec Institute, "Analyzing Malicious PDFs": http://resources.infosecinstitute.com/analyzing-malicious-pdf/

5 "Stegosploit": http://stegosploit.info/

6 Grimes, Roger A. InfoWorld, "7 sneak attacks used by today's most devious hackers": http://www.infoworld.com/article/2610239/malware/7-sneak-attacks-used-by-today-s-most-devious-hackers.html

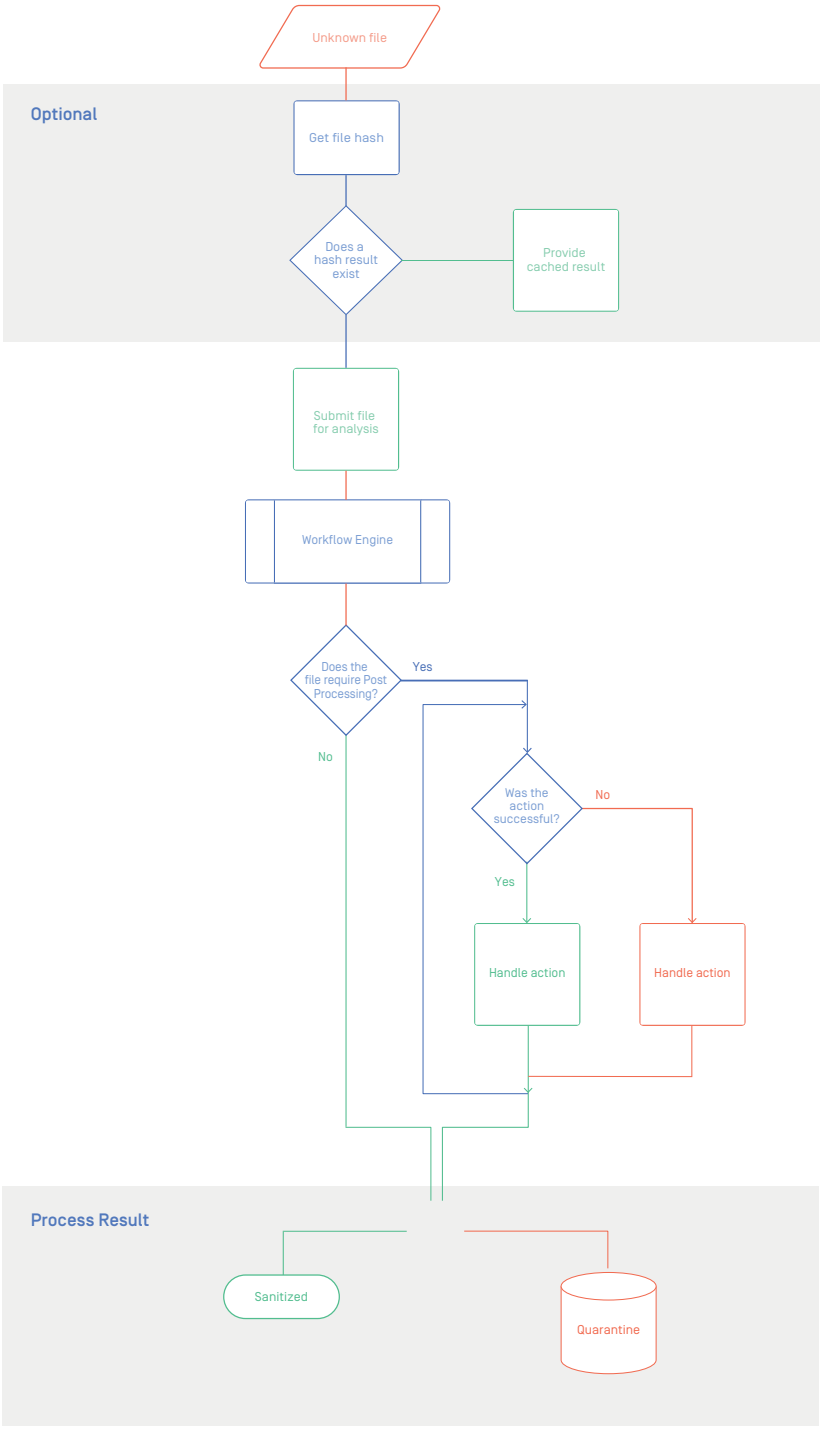
7 Dori, Amit. Check Point, "Spear Phishing 2.0 Adds Social Engineering & VM Evasion": https://blog.checkpoint.com/2016/05/20/spear-phishing-2-0-adds-social-engineering-vm-evasion/

8 Proofpoint, "Malicious Macros Add Sandbox Evasion Techniques to Distribute New Dridex": https://www.proofpoint.com/us/threat-insight/post/malicious-macros-add-to-sandbox-evasion-techniques-to-distribute-new-dridex

9 Raymond. Raymond.CC Blog, "3 Methods to Spoof Fake File Extensions in Windows": https://www.raymond.cc/blog/run-exe-files-as-jpg-png-gif-or-under-any-different-extension/

APPENDIX **A**

API Integration



OPSWAT.

Trust no file. Trust no device.