Mistrust Plugins You Must: A Large-Scale Study Of Malicious Plugins In WordPress Marketplaces

Ranjita Pai Kasturi, Jonathan Fuller, Yiting Sun, Omar Chabklo,
Andres Rodriguez, Jeman Park*, Brendan Saltaformaggio*
Georgia Institute of Technology

Abstract

Modern websites owe most of their aesthetics and functionalities to Content Management Systems (CMS) plugins, which are bought and sold on widely popular marketplaces. Driven by economic incentives, attackers abuse the trust in this economy: selling malware on legitimate marketplaces, pirating popular plugins, and infecting plugins post-deployment. This research studied the evolution of CMS plugins in over 400K production webservers dating back to 2012. We developed YODA, an automated framework to detect malicious plugins and track down their origin. YODA uncovered 47,337 malicious plugins on 24,931 unique websites. Among these, $41.5K had been spent on 3,685 malicious plugins sold on legitimate plugin marketplaces. Pirated plugins cheated developers out of $228K in revenues. Post-deployment attacks infected $834K worth of previously benign plugins with malware. Lastly, YODA informs our remediation efforts, as over 94% of these malicious plugins *are still active today.*

1 Introduction

Many modern websites are almost entirely constructed from plugins and themes, which place implicit trust on large amounts of un-vetted code with limitless access to the webserver. Our research uncovered that this trust is often broken for monetary gains and that malicious plugin authors are literally selling plugins packed with malware to unsuspecting victims. Worse still, we found that most malicious plugins sold on popular plugin marketplaces do not employ evasion or obfuscation techniques, preferring to brazenly *hide in plain sight.*

Popular content management system (CMS) plugin marketplaces generate over a billion dollars in revenue every year [1], but little has been done by the research community to evaluate, assess, and ensure the safety of the consumers (website owners). Past research studied malicious apps in the Google Play Store [2], malicious extensions on the Chrome Web Store [3], [4], and malicious packages in package registries [5]. Prior work also exposed malicious behaviors on webservers, such as the presence of vulnerabilities [6], [7], webshells [8], and backdoors [9], but none analyzed the underlying plugins which lead to many of these attacks. Further, the complexities of prior research solutions have prevented the average CMS-user from adopting them.

CMS website owners often rely on simple indicators such as plugin popularity, ratings, and reviews on the plugin marketplaces to determine that a plugin is safe to install on their website [10]. The diligent CMS-user may consult freely available [11] or commercial [12], [13] plugin vulnerability scan databases before installing a plugin. Unfortunately, these sources provide neither complete nor robust measures of security. Driven by economic incentives, attackers *buy the codebase* of popular free plugins, add malicious code, and wait for plugin users to auto-update [14]. In such cases, none of the commonly used simple indicators can help prevent malware from infiltrating the website.

Our research performed a global measurement of the malicious WordPress plugins ecosystem. We worked with CodeGuard[1], to analyze the WordPress plugins in over 400,000 unique webservers dating back to 2012. We uncovered 47,337 malicious plugin installs on 24,931 unique websites. Even worse, 3,685 of these plugin instances were sold on legitimate plugin marketplaces. Tracking the webservers and plugins over 8 years gave us a unique vantage point to study the temporal evolution of malicious plugins from a global perspective. We found that the number of malicious plugins on websites has steadily increased over the years, and malicious activity peaked in March 2020. Shockingly, 94% of the malicious plugins installed over those 8 years *are still active today.*

*Co-corresponding author.

[1]One of the largest corporate website security and backup solutions on the market.

Throughout our study, we found that solving this problem is challenging due to the diverse range of stakeholders in the CMS plugin ecosystem. Each has different motivations and visibilities into this malicious plugin problem. Website owners have full visibility over the webserver activity, but they rely on naive indicators when installing plugins. Hosting providers have no visibility into the plugin installations but need to ensure their hosting platform remains malware-free. Plugin marketplaces have visibility over the plugins they host but need a scalable and efficient measurement of the malicious plugins being sold on their marketplaces.

An ideal solution must ensure ease of use and reliable detection since plugins could be malicious anywhere in this supply chain: from the source marketplace to a post-deployment web attack (i.e., fake plugin injection).

To address these challenges, we developed YODA, an automated framework to identify malicious plugins and their origin. Towards usability, this can be integrated as part of the webserver hosting platform or deployed by the plugin marketplace. Website owners are often unaware of the plugins installed *or injected* into their website, so when deployed by a hosting provider, YODA starts by detecting a webserver's (possibly hidden) plugins. YODA also crawls popular CMS plugin marketplaces to identify each plugin's provenance, ownership, and global impact. Using YODA, website owners and hosting providers can identify malicious plugins on the webserver; plugin developers and marketplaces can vet their plugins before distribution.

Our 8-year study using YODA revealed several concerning facts: While the website owners trusted the plugin ecosystem and spent a total of $7.3M on *only the plugins in our dataset*, we found that this trust is often broken for the attackers' monetary gains. Attackers impersonated benign plugin authors and spread malware by distributing pirated plugins. YODA found 1,354 instances of pirated plugins responsible for one of the largest known malvertising campaigns [15], many of which are still active today. Furthermore, $41.5K was spent on malicious plugins sold on legitimate plugin marketplaces, and plugins that cost a total of $834K were infected post-deployment by attackers. We hope that YODA can regulate and reinstate the trust between all stakeholders in the plugin ecosystem. Lastly, we have made YODA's source code available at: https://cyfi.ece.gatech.edu/

2 Preliminary Study: Perilous Economy

Plugins are groups of files that work together to add aesthetic features and functionalities to a CMS website. Upon each visit to the website, the CMS loads all active plugins (i.e., executes plugin code) on the webserver.

Our Dataset. Our collaboration with CodeGuard furnished access to the nightly backups of over 400K unique WordPress websites. These backups contain the server-side files and their version-controlled changes collected from July 2012 to July 2020. They give YODA the vantage point of both an individual website owner as well as a hosting provider (i.e., access to the webserver files). This allows us to retroactively deploy YODA over 8 years by executing YODA on each nightly backup for every website. Note that CodeGuard anonymized the website owner profiles — only a random ID and the URL were linked to each website backup. Furthermore, we only analyzed the webserver files with no database access. All websites in our study are CodeGuard's active clients, i.e., if a client stops using CodeGuard's service all their data is immediately deleted, thus we will not have access to it.

Responsible Disclosure. The individual website-owners are anonymized by CodeGuard. However, all of CodeGuard's customers agree to their Privacy Policy whereby their data may be shared with third-parties to help CodeGuard safeguard their websites. Since we cannot directly contact the affected victims, we have alerted CodeGuard about our findings and they are processing the disclosure.

Plugin Marketplaces. WordPress plugins and themes generate millions of dollars in sales every year [1]. These plugins[2] are either created by an individual or teams of developers, including WordPress themselves. After detecting the plugins in our dataset (5.7M of them), we performed a preliminary study to understand the scale of this economy. We measured the plugin downloads and price data in July 2020 for the plugins in our dataset. This required scraping and cross-correlating data from the plugin code and online marketplaces. Table 1 lists the plugin marketplaces, the total number of plugins, and the unique number of plugins from these marketplaces in our dataset. As seen in Table 1, thousands of plugins are freely available on the WordPress repositories [16], [17] and software development platforms, such as Github [18]. In rare cases (below 0.5% in our dataset), some plugins are available on multiple marketplaces.

Paid versions of the plugins are sold through marketplaces, e.g., ThemeForest [20], CodeCanyon [19], and Easy Digital Downloads (EDD) [22]. Here, individual plugins are sold for as little as $2, while the bestselling plugins are valued at around $63. Table 1 Columns 4-6 highlight the plugin popularity in terms of the number of downloads. WP Plugins [16] is the most popular marketplace overall, with 7.5M average downloads per plugin. Some marketplaces do not sell individual plugins and instead provide a subscription service for all plugins at a flat rate. For example,

[2]Plugins and themes are together referred to as plugins.

	Markeplace	#Plugins		Downloads Range			Cost of Plugins			Money Spent	
		Total	Unique	Min.	Avg.	Max.	Min.	Avg.	Max.	Dataset	Global
Free Plugins	WP Plugins [16]	5,276,450	27,430	14	7.5M	260M	-	-	-	-	-
	WP Themes [17]	506,342	5,450	3	293K	7.4M	-	-	-	-	-
	Github [18][1]	448,324	5,155	0	3	9	-	-	-	-	-
Paid Plugins	CodeCanyon [19][2]	38,060	2,428	1	1.2K	9K	$2	$32	$1,104	$1.2M	$82.6M
	ThemeForest [20]	61,574	5,837	1	29.6K	611K	$2	$84	$499	$5.1M	$31.3M
	WPMU DEV [21][3]	5,984	110	55.4K	1.4M	10.5M	$15	$63	$190	$370K	$96.4M
	EDD [22][4]	13,123	245	-	-	-	$6	$49	$199	$643K	-
	Total	5,782,783	43,621	1	939K	260M	$2	$63	$1,104	$7.3M	$210.3M

1: Since Github does not provide the repository download info, we used the number of stars as a measure of popularity.
2: We found a plugin, Choco Drops [23], on CodeCanyon for $10,000,003. Since this is an outlier, it has been excluded here.
3: WPMU DEV charges a $15 monthly subscription to use any plugin on this marketplace. The price range reflects the yearly cost.
4: Downloads range was not publicly available for EDD and has been excluded in this table.

Table 1: The Economy of WordPress Plugin Marketplaces.

WPMU DEV [21] has a $49/month subscription and is the most popular paid plugin marketplace in our dataset with 1.4M average downloads per plugin. Less-popular plugins are also directly available from freelance developers or small businesses [24]–[26].

Since plugin marketplaces do not provide any price history, we used the reported download counts and the prices from July 2020 to estimate the money spent on these plugins. Table 1 shows that website owners from our dataset alone spent $7.3M at plugin marketplaces, and we estimate the revenue earned by these plugins globally to be over $210M based on a conservative estimate of the reported download counts. We found several plugins sold with an extended license for more developer support time. We considered the regular support pricing to estimate a lower-bound for spending in the plugin ecosystem. Our estimate is also confirmed by themeshunter.com, one of the biggest WordPress themes catalogue on the market [27].

Nulled Marketplaces. Since most paid plugin marketplaces do not offer a trial option, several marketplaces started a *"try before you buy"* initiative. Unfortunately, this gave rise to pirated "trial plugin" marketplaces, referred to as nulled marketplaces. Nulled plugins are pirated versions of originally paid plugins, freely distributed via nulled marketplaces (unbeknownst to the original creator). Generally, these plugins have been hacked or contain modified code to cause user harm or collect sensitive user data and made to work indefinitely without a license key [28]. Our study has found that, more often than not, nulled plugins introduce malicious code onto webservers (§5.3).

Insufficient Market Oversight. While these marketplaces are growing rapidly, the regulations to assess plugins are minimal. For example, as mentioned in Table 1, our study found a CodeCanyon plugin for $10,000,003 [23] with a note from the plugin author to not buy the plugin. The fact that the plugin author was able to set the price so high, to dissuade downloaders, rather than legitimately removing the listing underscores how little oversight these marketplaces provide. We also found that attackers include malicious behaviors in plugins then sell them on reputable plugin marketplaces, such as the WordPress repository. A report by Wordfence [14], a leading WordPress malware scanner, found nine popular plugins updated at source (i.e., the WordPress plugin store) with malicious code as part of a coordinated spam campaign (see §7). It is more urgent now than ever to study the impact of this problem and address the challenges toward securing the plugin ecosystem.

3 Design

Figure 1 shows an overview of the YODA pipeline. YODA first conducts Plugin Detection (§3.1). Hosting providers and website owners can deploy this to detect plugins on their websites. But, marketplaces or plugin developers can skip directly to YODA's Malicious Behavior Detection (§3.2) for a given plugin. Next, YODA identifies the Origin of Malicious Plugins (§3.3). Finally, it performs an Impact Study (§3.4) to understand the scale and impact of the plugin economy.

3.1 Plugin Detection

YODA detects all of the webserver's plugins by identifying the plugin root and all the associated files that belong to the plugin. Reliably detecting plugins based on the webserver files alone can be challenging because CMSs provide limited guidelines leading to a lack of code consistency. CMS-users (plugin developers, website owners, and attackers) often customize their plugins and do not follow coding guidelines. CMSs provide directories to maintain plugins and themes, but users often place them in random locations on the server, so we cannot solely rely on the directory structure to reliably detect plugins.

To address these challenges, YODA performs (1) *metadata analysis* to identify the plugin root files and

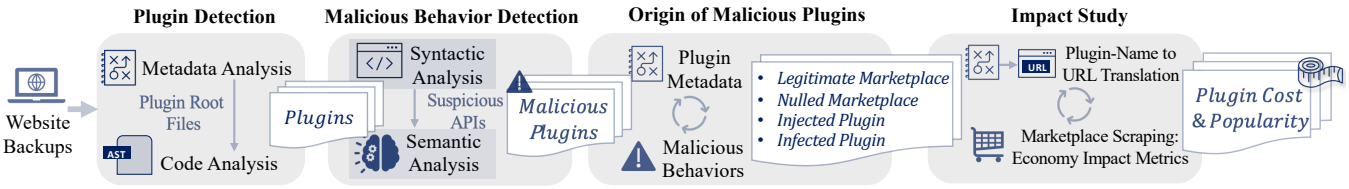

Figure 1: YODA Design Overview.

(2) *code analysis* to identify all of the associated files as part of the plugin (shown in Figure 1).

Metadata Analysis. YODA parses the comments from all of the server-side code files and performs regular expression matching to identify the plugin root files (i.e., files containing the plugin header, a specially formatted block comment that contains metadata about the plugin). This ensures that bad coding practices and possibly hidden plugins injected by attackers do not go undetected. A typical plugin header is shown in Figure 4, Appendix A. Note that WordPress only identifies and loads plugins with a header, and all plugins must contain a single plugin root. If an attacker tries to evade YODA's detection by dropping the header, the plugin will not be loaded by the WordPress core and remain dormant on the webserver. For every plugin root, YODA extracts and records the plugin metadata from the header, including the plugin name, plugin URI, author name, author URI, and plugin version. As we will see later (§4.1), we use this plugin version to understand how many CMS-users maintain their plugins updated to the latest version.

Code Analysis. With the plugin root files identified, YODA proceeds to find all associated plugin files $\mathbb{P}_i$, i.e., $\mathbb{P}_i = \{f_{i_1}, f_{i_2}, ...\}$ where f_{i_j} is the j^{th} file in plugin $\mathbb{P}_i$. To do this, YODA generates and parses the abstract syntax tree (AST) of all of the server-side code files in parallel and sub-directories of the plugin root. Since several CMS-users customize their plugins either by using configuration files or explicitly modifying the PHP code, YODA will detect f_{i_j} based on three scores, listed in decreasing order of importance. We use constant weights (3, 2, and 1) coupled with inverse exponentials to rank these scores since it is a common approach for ranking program modules [29]. While we could have chosen any decreasing range, we found that this combination produced distinct ranges that identify plugin files from non-plugin files.

1. Header Score. The existence of a plugin header n_j in a file f_{i_j} (computed during metadata analysis) is used to derive the header score h_j (weight = 3). Here, n_j can take values 1 (for plugin root with a single header) or 0 (for the associated files with no header). However, the header score is included in the reliability score with the highest weight to ensure a group of plugin-like files with no header is not incorrectly identified as a plugin.

$$Header\ \ Score \qquad h_j = 3 * n_j$$

2. Reference Score. YODA uses the number of reference calls m_j linking other files as part of the plugin to derive the reference score r_j (weight = 2). Each score is a sum of the individual contributions from all of the linked files towards the entire plugin. This contribution is scaled by an integer weight in the numerator to model the importance of linked files and an exponential in the denominator to account for a large number of referenced files. Here, the exponent x starts with n_j to further ensure the reference score contribution is lower than the header score.

$$Reference\ \ Score \qquad r_j = \sum_{x=n_j}^{n_j+m_j-1} \frac{2}{2^x}$$

3. API Score. The number of occurrences of plugin-specific API calls l_j (the full list of APIs is shown in Appendix B) is used to derive the API score a_j (weight = 1). Since APIs alone are insufficient to detect a plugin, l_j is scaled using a weight of 1 (numerator). Here, the exponent x starts with $n_j + m_j$ to ensure the API score contribution is lower than the header and reference scores.

$$API\ \ Score \qquad a_j = \sum_{x=n_j+m_j}^{n_j+m_j+l_j-1} \frac{1}{2^x}$$

The sum of all three scores for all plugin files in parallel and sub-directories of the plugin root is divided by the upper bound of this sum to calculate the reliability score $\mathbb{R}_i$ for each plugin $\mathbb{P}_i$.

$\forall$ plugin files $f_{i_j} \in \mathbb{P}_i$,

$$Reliability\ \ Score \qquad \mathbb{R}_i = \frac{\sum_{f_{i_j}} (h_j + r_j + a_j)}{2 * max(h_j, r_j, a_j)} * 100\%$$

The reliability score is a measure of the likelihood of a group of files being part of a plugin. If this score is greater than 95% for a group of files, YODA detects it as a plugin. We set the strictest possible threshold because we found that for a true positive plugin this score is always >98% and <55% otherwise.

An additional challenge (a special case of the above) is child plugins. They are extensions of the original plugin that enable the website owners to add customization, i.e., modify functionalities without having them disappear after an upgrade. YODA handles child plugin detection

Malicious Behavior	Semantic Models
Webshell	$Super\_Global[input] \rightarrow Exec$
Post Injection	$(URL \in Blacklist) \wedge (URL \rightarrow Download \rightarrow Add\_Post)$
Input Gating	$Super\_Global[password] \rightarrow Conditional \rightarrow Exec$
SSO Backdoor	$Create\_User \rightarrow Chng\_User\_Perm \rightarrow Register\_User \rightarrow Redirect\_NewUser\_Admin\_URL$
Library Function Exists	$Conditional \rightarrow Func\_Exists \rightarrow Create\_Func$
Spam Injection	$(URL \in Blacklist) \wedge (URL \rightarrow Download \rightarrow Add\_Content)$
Code Obfuscation	$(Jumbed\_Obfus \vee Long\_Line) \vee (Decode \rightarrow Exec)$
Blackhat SEO	$Conditional \rightarrow SE\_Bots \rightarrow (URL \in Blacklist) \wedge Download \rightarrow Replace\_Content$
Downloader	$(URL \in Blacklist) \wedge (URL \rightarrow Download)$
Function Reconstruction	$(Str1, Str2, .., StrN) \rightarrow Concat \rightarrow Create\_Func$
Insert User	$Create\_User \rightarrow Register\_User$
Malvertising	$(URL \in Blacklist) \wedge (URL \rightarrow Download \rightarrow (Redirect \vee Insert\_Popup))$
Fake Plugin	$Conditional \rightarrow Super\_Global[Str] \rightarrow Decode \rightarrow Exec \rightarrow Delete\_Payload$
Cryptominer	$(URL \in Blacklist) \wedge (URL \rightarrow Download \rightarrow File\_RW \rightarrow Chng\_File\_Perm \rightarrow Exec)$

Table 2: High-level Dataflow Sequence of the Semantic Malicious Behavior Models from Source to Sink.

by recursively searching through plugin sub-directories to find plugin roots and storing them as separate child plugins under their respective parent plugins.

Effective Plugin State. Since YODA can retroactively run on temporal webserver snapshots, it records the effective plugin state in each of these snapshots. For each plugin, YODA uses the individual file states of all plugin files, i.e., added (`'A'`), modified (`'M'`), no change (`'NC'`), or deleted (`'D'`) to derive the effective plugin state that could also take one of the four values: `A/M/NC/D`. If all individual plugin file states are added, deleted, or no change, then the effective plugin state is `'A'`, `'D'`, or `'NC'`, respectively. All other individual file state combinations produce an effective plugin state `'M'`.

3.2 Malicious Behavior Detection

Preliminary Study. We started by analyzing all plugins from 85 known-compromised website backups taken between April 2018 and June 2020. CodeGuard provided this subset based on signature-based AV alerts for well-known web malware. We also referenced all reports of malicious plugins being removed from popular marketplaces between 2013-2018 [14], [30]–[32] to identify and collect available malicious plugin samples. Since most of the removed plugins were not accessible on the marketplaces, we used the plugin name and version to scan our dataset and collect all additional malicious plugin samples[3].

We manually investigated all the plugins from above and identified 14 distinct malicious behaviors, listed in Table 2 Column 1. We will describe the modeling of these behaviors in the rest of this section. We also found that each of these behaviors had multiple implementations and using rule-based syntactic detection alone would quickly leave the rules obsolete. Further, state-of-the-art web malware detection relies on structure-aware semantic features (e.g., code implementations of webshell features) within a single code file [33], [34]. Existing techniques do not consider the interactions between file groups. This is necessary because attackers distribute malicious behavior implementations across multiple plugin files, thus evading existing techniques.

YODA addresses these challenges by employing both syntactic features (e.g., file meta-data, sensitive APIs) and context-aware semantic features of all plugin code files (e.g., AST with resolved file dependencies). Syntactic analysis uses data flow analysis to identify suspicious APIs being used as sinks in plugin code files. Owing to space constraints, this is presented in Appendix C.

Semantic Analysis. The presence of suspicious APIs alone does not equate to malicious plugin behavior. To ensure that the malicious behaviors are detected across multiple plugin files, YODA performs context-aware semantic analysis. In the dependency resolved ASTs, it marks all the sensitive APIs identified earlier as sinks and performs targeted inter-procedural backward slicing on the AST from each sink to the predefined sources using php-ast [35]. These source-sink dataflows, called *'semantic models'* are summarized in Table 2.

Note that YODA's models are both *composable* and *extensible*. For some dataflows, the sinks can also act as an intermediate node. For example, *Download* is a sink for the downloader malicious behavior and an intermediate node for the blackhat SEO, post injection, malvertizing, and spam injection behaviors. Since attackers extend existing techniques, this composability allows YODA to scale with evolving malware. Further, as new malware behaviors emerge (e.g., the recent trend of SSO Backdoor), analysts can easily extend YODA's models by composing existing primitive models with new API sinks used in the attack. Next, we describe each of the semantic models from Table 2.

1. Webshell. The plugin takes executable code as input via superglobal variables ("$Super\_Global[input]$" in Table 2) which is then passed to an *Exec* sink that

[3]Available at: https://cyfi.ece.gatech.edu/.

executes this code on the webserver.

2. Post Injection. The plugin code contains a URL that has been *Blacklist*ed as malicious by VirusTotal [36] or URLHaus [37], and it *Download*s content from this URL and inserts it as a WordPress post (*Add_Post*). We found that the URLs used by attackers are not always flagged as malicious by VirusTotal or URLHaus. We identified randomly generated strings used as throwaway domain names (e.g., `www.fatots.top`, `www.gacocs.com`) to deliver malicious content to these plugins. We provide this full list as part of the YODA source code.

3. Input Gating. Attackers protect their injected code based on a predefined password. Here, the 'password' parameter in a super global variable (*Super_Global*[*password*]) is set, it is conditionally evaluated, and code is executed (*Exec*) based on the conditional evaluation success. While this may appear like harmless password-protected code execution, benign plugins store client credentials in the website's database and do not employ only hard-coded passwords.

4. SSO Backdoor. Attackers are abusing the single sign on feature to create a backdoor via user accounts with admin privileges. Here, the plugin creates a user object (e.g., *Create_User* via `$user = array( 'user_login' => $uname, 'user_pass' => $pword)`), changes the user permissions to provide administrator privileges, registers this user with the CMS (e.g. *Register_User* via `wp_user_insert`), and finally redirects all requests to this new user's admin URL.

5. Library Function Exists. If the plugin finds a missing library function (*Conditional* $\rightarrow$ *Func_Exists*), it locally implements the function (*Create_Func*) to redefine it. While it is common to check if a function exists, benign plugins do not reimplement library functions but instead include the library.

6. Spam Injection. The plugin code contains a blacklisted URL (VirusTotal, URLHaus, or in-house curated URL list), and it *Download*s content from this URL and injects the downloaded content to the HTML output (*Add_Content*) each time the website is loaded.

7. Code Obfuscation. The plugin contains (1) jumbled obfuscation patterns (*Jumbed_Obfus*), (2) long lines of code with over 50 code instructions in the same line (*Long_Line*) during syntactic analysis, or (3) encoded strings passed to *Decode* and *Exec* sinks. These are the 3 predominant categories of code obfuscation seen in our study. YODA could identify different obfuscation variants, and the detection module is made available as part of the YODA source code.

8. Blackhat SEO. Attackers employ conditional checks to detect if the website is being loaded by search engine bots (*SE_Bots*), e.g., googlebot, bingbot, baiduspider. They *Download* SEO campaign content from a $URL \in Blacklist$ and replace concealed HTML elements (*Replace_Content*) in the plugins with this downloaded content. This impacts the website's indexing by search engines.

9. Downloader. The plugin *Download*s content from $URL \in Blacklist$. Note that, if YODA finds *Download* as an intermediate sink for other attack behaviors, it assigns the appropriate attack behavior and does not flag the plugin as a downloader.

10. Function Reconstruction. To evade signature-based AVs, attackers break suspicious function names to substrings ($Str1, Str2, ..$) that can then be concatenated to form the function name. Attackers then use PHP's *Create_Func* to create a function that internally performs an `eval()` or executes this function.

11. Insert User. The plugin creates a user object (*Create_User*) and registers this user account with the CMS (*Register_User*). Benign plugins hardly add new user accounts to the CMS. Different from SSO backdoor, this user is created for one-time use and this user's contents are not loaded each time a web page is requested.

12. Malvertizing. Attackers monetize plugins to serve malicious ads. The plugin *Download*s content from a $URL \in Blacklist$ and redirects website visitors to a malicious site or inserts a downloaded popup ($Redirect \vee Insert\_Popup$ in Table 2).

13. Fake Plugin. Attackers inject full-fledged plugins that not only give backdoor access but also run malicious code each time the website is loaded. In particular, fake plugins receive encoded payloads (generally using base64 decoding) from superglobal variables (*Super_Global*[*Str*]), *Decode* and *Exec* this payload, and then delete it (*Delete_Payload*).

14. Cryptominer. The plugin *Download*s a mining script from a $URL \in Blacklist$, writes it to a file (*File_RW*), changes the file permission to executable, and then *Exec*s the file.

3.3 Origin of Malicious Plugins

YODA then determines the origin of these malicious behaviors. This helps understand the different attacker entry points within the CMS ecosystem. Our preliminary study uncovered that the malicious plugin behaviors originate from one of these four sources.

1. Nulled Plugin Marketplace. Nulled plugins commonly include multiple malicious domains (adds redundancy during domain takedown) to download malicious content on the webserver. If YODA records downloader, malvertizing, or spam injection behaviors when the effective plugin state was `'A'`, and if the plugin contains multiple redundant blacklisted URLs, it is categorized as nulled based on its behavior. Also, if

the plugin name contains nulled marketplace metadata (e.g., "Shared on VestaThemes.com"), it is categorized as nulled based on its metadata. Note that not all metadata-based nulled plugins are malicious (§5.3).

2. Legitimate Plugin Marketplace. YODA marks the malicious origin as a legitimate plugin marketplace if: (1) one or more malicious behaviors are seen when the effective plugin state is `'A'`; or (2) the effective plugin state is `'M'` due to plugin version and/or author change[4]. Since some nulled plugins masquerade as legitimate plugins, YODA first categorizes nulled plugins with redundant malicious domains and excludes them from the legitimate marketplace category.

3. Injected Plugin. If YODA finds (1) fake plugin behavior when the effective plugin state was `'A'`, or (2) fake plugin and code obfuscation behaviors when the effective plugin state is `'M'`, the plugin is categorized as an injected plugin. Note, plugins with code obfuscation are not always injected plugins. Only if the plugin did not originate from nulled or legitimate marketplaces, then YODA marks it as an injected plugin since these plugins are not sold on marketplaces.

4. Infected Plugin. We found that malicious plugins on the webserver tried to increase the attack's coverage by hijacking other plugins. If YODA found malicious behaviors in a plugin with effective plugin state `'M'` in an already compromised website (i.e., it has one or more malicious plugins prior to the snapshot under analysis), it is marked as infected. If it was infected when the effective plugin state was `'A'`, YODA may incorrectly label an infected plugin as originating from a legitimate marketplace. This is resolved via cross-website verification.

Cross-Website Verification. Using backups from over 400K web servers, we employ cross-website verification as an additional guarantee for the malicious origin categorization applied at a single-website level. Note, legitimate marketplace, nulled marketplace, and injected plugin categories are mutually exclusive. However, plugins from all of these categories can be infected by other malicious plugins on the webserver. YODA performs a cross-website comparison of all malicious plugins originating from legitimate or nulled marketplaces. In particular, if the identified malicious behaviors are common across all websites, then the labeled categorization is validated as correct. Otherwise, they will be correctly relabeled as infected plugins.

3.4 Impact Study

The origin of malicious plugins in §3.3 reveals the broad attacker platforms used to victimize CMS users. To understand the scale of this impact on the plugin marketplaces, YODA extracts the *impact metrics* associated with each plugin (i.e., monetary impact in terms of plugin cost and popularity impact in terms of the number of downloads) by mapping the plugins in our dataset to the plugin marketplace it originated from. We chose the 7 most popular plugin markets — three unpaid (WordPress Plugins, WordPress Themes, and Github) and four paid (ThemeForest, CodeCanyon, WPMU DEV, and Easy Digital Downloads) — to perform this study. This can be challenging because the impact metrics extraction varies between markets due to the lack of code consistency.

To address this, we reverse-engineered the plugin-name-to-URL translation for these marketplaces, and YODA was programmed to scrape the impact metrics. YODA first constructs the URL to visit by appending the plugin name (e.g., twentytwenty) to the market-specific URL (e.g., https://wordpress.org/themes/) and performs a GET request on the effective URL (e.g.,https://wordpress.org/themes/twentytwenty) to determine if the plugin is in the marketplace. For some marketplaces (e.g., CodeCanyon), the required URL cannot be constructed solely from the plugin name. To address this, a search query is constructed using the plugin's name. The search results are parsed to find if the target plugin exists in the marketplace. This impact metrics extraction can be extended to other marketplaces by updating YODA with the new plugin-name-to-URL translation and scraping patterns.

After obtaining the plugin's marketplace listing (i.e., a successful GET request response), the impact metrics extraction is similar across marketplaces, specific only to the web page formatting. All available metadata on the listing is stored in a database for easier queries. This metadata consists of the plugin's latest version, plugin rating, cost, and the number of sales and downloads, which when applied to a large-scale study, reveal the impact of these plugins on the community. This data will be used to infer any correlations that may exist between malicious plugins, their cost, and their popularity.

4 Validating YODA

To validate YODA's design considerations, we used 120 unique WordPress websites collected between Apr 2018 and Feb 2021. 60 were compromised with web-attacks as classified by pattern-based AV and the remaining 60 were randomly chosen unbiased websites. We used this dataset to establish ground truth and validate YODA's accuracy in detecting plugins and malicious plugin behaviors on a local workstation running Ubuntu 16.04 with 32GB memory and 8 x 3.60GHz Intel Core i7 CPUs.

[4]The effective state can be `'M'` for several reasons such as code customization by the website owner, code injected by an attacker, etc. Still, the plugin version or the plugin author does not change.

Total #Websites: 120	Total #Plugins: 3,168						
Plugin Name	#Y	TP		FP	FN[3]	LV_m	LV_M
		EM[1]	C[2]				
Yoast SEO	47	27	18	2	0	26	38
Contact Form 7	41	28	12	1	0	23	28
Wordfence Security	37	30	6	1	0	3	26
Manage WP - Worker	34	20	14	0	0	14	23
Add From Server	31	19	12	0	0	16	24
Shield Security	29	12	16	1	0	4	13
WP Rocket	27	10	15	2	0	10	12
MainWP Child	25	18	6	1	1	11	19
Easy WP SMTP	24	12	12	0	0	9	13
Amazon Web Services	20	18	2	0	0	6	17
Simple Social Icons	19	19	0	0	0	15	15
WP Offload S3 Lite	18	17	0	1	0	7	14
Jetpack	17	10	6	1	0	2	12
Sharedaddy	15	12	2	1	0	13	13
Akismet Anti-Spam	14	13	1	0	0	1	11
Total Top 15	398	265	122	11	1	160	278
Total Overall	3,168	2,240	889	39	3	728	2,060

1: #W where the plugin exactly matches the ground truth plugin.
2: #W with customized plugins correctly identified by YODA.
3: #W with customized plugins incorrectly identified by YODA.

Table 3: Plugin Detection Evaluation.

4.1 Plugin Detection Evaluation

Ground Truth. We first evaluate YODA's plugin detection. As mentioned in §3.1, website owners often customize their plugins, either using configuration files or explicitly modifying the PHP code. Thus, it is difficult to verify that a detected plugin matches a known plugin from the marketplace. To evaluate YODA, we need to determine if each plugin that YODA detected is either: (1) an exact match (EM), (2) a true positive match with customization (C), (3) a false positive (FP), i.e., YODA labeled a non-existent plugin, or (4) a false negative (FN), i.e., YODA missed labeling a group of files as a plugin.

We used YODA to identify an initial list of plugins in all 120 website backups. This list contained EMs, Cs, or FPs (per above). To determine which, we created a ground truth plugin set by downloading these plugins from the plugin marketplaces. We also contacted the authors of paid plugins found in our dataset and received all versions of these plugins as well. We compared all the files (via MD5 hash) for each plugin detected by YODA against the files for the *same version* of the plugin within the ground truth set. If 100% of the files from the ground truth plugin matched those in the detected plugin, we classify it as an EM (Table 3 Column 3). Greater than 90% match is considered a true positive with customization (C, Column 4). If the comparison led to a less than 90% match, we classified this plugin as an FP (Column 5). In fact, customized plugins rarely differ by more than one file (we found only 8 instances of multi-file customization in our dataset), thus a 90% match is so strict that it is less favorable to YODA, but we aim to aggressively flag any FPs. We manually investigated all mismatches.

To check for the FNs, we pulled every version of all freely available plugins from the WordPress SVN[5] plugin repository. We also added all versions of the free and paid plugins from above. We then compared all file hashes from the downloaded plugins against the files in each website. If 90% or more of the plugin's files match files in the website and YODA did not mark the group of files as a plugin, we count this is as an FN.

Detection Results. In the 120 websites, YODA found a total of 3,168 plugin instances (#Y). Table 3 summarizes the results for all plugins and drills down into these results for the top 15 plugins based on their popularity in our dataset. Of the 3,168 plugin instances in Table 3, YODA correctly detected 3,129 (i.e., 2,240 + 889) plugins. 2,240 of these plugins exactly matched the ground truth dataset, and 889 plugins were customized — a TP rate of 98.7%.

We manually verified all the plugins marked as FP and found that only 11 of the 39 plugins were actually FPs (i.e., a group of files incorrectly identified as a plugin). Here, the website owner copied the plugin root file (containing plugin APIs and missing referenced files) to their home directory (likely part of customization or backup), misleading YODA into identifying a group of files as a plugin. The remaining 28 of the 39 plugins either redefined the base WordPress APIs or replaced them entirely with custom APIs; such heavy customization in a single file put them below the strict 90% match. Thus, despite using a strict measure for FPs, the FP rate is reasonable (1.2%). The 3 FNs we found were due to the website owners deleting the plugin header as part of customization. Since the header holds the highest weight for determining a plugin group, YODA missed identifying these plugins.

We now use this dataset and the plugin version extracted by YODA to understand if CMS-users keep their plugins updated to the latest version. Columns 7 and 8 show the number of websites that had the plugins at the latest minor version (LV_m) and the latest major version (LV_M). For example, if the latest plugin version on the marketplace is 4.3.6 and it matches our dataset plugin version, we count it as LV_m. If our dataset plugin version is 4.3.2, since the major version (i.e., 4.3) is still up to date, we count this as LV_M.[6] Only 40% (160 of 398) of the top 15 plugins and 23% (728 of 3,168) of all plugins were updated to the latest minor version. From Column 8, we find that about 70% (278 of 398) of the top 15 plugins and 65% (2,060 of 3,168) of all plugins are updated to the latest major version. Over 35% of all plugins used are clearly outdated.

[5]WordPress uses SVN to maintain version-controlled plugins.
[6]Latest major version includes all latest minor versions.

Total #Websites: 120	Total #Plugins: 3,132					
Malicious Behavior	#W	#GT	#Y	TP	FP	FP
Code Obfuscation	15	28	28	28	0	0
Webshell	19	23	26	23	3	0
Function Reconstruction	7	16	18	16	2	0
Downloader	7	12	14	12	2	0
Library Function Exists	10	13	14	13	1	0
Input Gating	4	13	13	13	0	0
Fake Plugin	3	7	7	7	0	0
Spam Injection	3	6	6	6	0	0
Malvertising	2	5	5	5	0	0
Insert User	2	3	3	3	0	0
Blackhat SEO	1	2	2	2	0	0
Post Injection	1	2	2	2	0	0
SSO Backdoor	1	2	2	2	0	0
Cryptominer	1	1	1	1	0	0
Total Malicious Plugins[1]	61	84	89	84	5	0
Total Benign Plugins	120	3,048	3,043	3,043	0	5

1: This is not the sum of the columns, but the total #websites and #plugins in the evaluation dataset with malicious behaviors.

Table 4: Evaluation of the Malicious Behavior Detection.

4.2 Malicious Behavior Evaluation

Ground Truth. After establishing confidence in YODA's plugin detection, we now evaluate the accuracy of identifying malicious plugins. We eliminated the 39 FP and included the 3 FN plugins from the same 3,168 plugins from above, and our team manually verified the server-side code files in all 3,132 plugins and tagged them with corresponding malicious behavior labels.[7] We then ran YODA's malicious behavior detection on all of these plugins and compared the labels assigned by YODA with the manually derived labels. The results are presented in Table 4.

Detection Results. In our dataset of 3,168 plugins across 120 websites, YODA reported 61 websites (#W) containing 89 plugin instances (#Y) that exhibit malicious behaviors whereas our manually labelled ground truth (#GT) showed that only 84 plugins across these websites were malicious. Recall, our dataset has 60 websites with known-compromises (i.e., web attacks detected), and 58 of these websites contained malicious plugins. In addition, YODA found 3 websites containing malicious plugins in the 60 randomly chosen websites. The malicious behaviors reported by YODA matched our ground truth for plugins from these 61 websites (i.e., TP). Based on our manual verification, we did not find any plugins that contained malicious behaviors missed by YODA, thus showing zero FNs.

Table 4 shows YODA produced FP detections for 8 behavior instances in 5 plugins. Our manual investigation revealed that 4 of these plugins used a combination of behaviors that resembled webshells (i.e., executing decoded content) and checking if the library function `base64_decode` exists and redefining it if not. Our investigation confirmed that these plugins did not show any outright malicious activity, but this rarely-benign code implementation misled YODA. Also, 2 plugins were falsely labeled as downloaders, due to VirusTotal falsely blacklisting the extracted URLs as malicious. This gives us confidence that YODA accurately detects plugins and malicious behaviors. Table 4 also summarizes the benign plugins in this dataset that we verified were not malicious.

[7] YODA did not have access to our manually derived labels.

5 Deploying YODA

#Websites	410,122		
Min. Duration	102 days	Min. #Plugins	1
Avg. Duration	406 days	Avg. #Plugins	49
Max. Duration	3,259 days	Max. #Plugins	68

Table 5: Dataset Summary.

We deployed YODA on the full dataset of 410,122 unique WordPress websites' nightly backups (Table 5). This dataset provides a realistic view of the plugin ecosystem because over 37% of the world's websites and over 63% of CMS-based websites run on WordPress [38]. It also allows us to retroactively deploy YODA over 8 years. The backups contain an average of 406 day-snapshots per website. Many backups went all the way back to 2012, representing some of the earliest customers of CodeGuard. Each website had between 1-68 plugins, with an average of 49 plugins per website. This high average shows that most website owners place *unwarranted trust* in plugins to keep their websites up and running.

Experimental Setup. We used Amazon Web Services (AWS) Elastic Compute (EC2) r5.2xlarge instances with 8 virtual CPUs and 64 GB of RAM to run YODA on the website backups. These instances were supervised by the AWS Batch job scheduling engine to deploy YODA on hundreds of backups in parallel.

5.1 Malicious Behavior Evolution

YODA found malicious plugin instances (#P) in 24,931 of the 410,122 websites (#W), shown in Table 6. As expected, over 10K malicious plugin instances used the age-old web attack techniques: webshells and code obfuscation. The infection ratio (IR, the ratio of #P to #W) shows a measure of infection spread. Several malicious behaviors have IR >3, implying that multiple plugins within the same website contain these same malicious behaviors. Closer inspection revealed that these are due to *plugin-to-plugin infection*: a single malicious plugin on the webserver infects multiple benign plugins, replicating the behavior.

Malicious Behavior	#W	#P	IR[1]	First Seen	Temporal Evolution (07-2012 - 07-2020)	Marketplace		Injected		Nulled		Infected	
						#W	#P	#W	#P	#W	#P	#W	#P
Webshell	7,921	10,279	1.3	Jul 2012		10	12	854	994	160	232	7,117	9,943
Code Obf	6,752	10,064	1.5	Aug 2012		0	0	409	558	1,055	1,214	5,509	8,819
Input Gating	5,928	23,140	3.9	Jul 2012		0	0	47	50	3,445	7,821	5,588	20,684
Downloader	2,314	5,944	3.6	Mar 2014		151	288	19	20	1,540	2,683	1,562	4,254
Spam Injection	1,202	3,723	3.1	Oct 2016		1,166	3,452	0	0	0	0	36	271
Lib Func Exists	2,233	3,576	1.6	Aug 2012		25	29	5	5	154	241	2,195	3,475
Blackhat SEO	1,358	1,714	1.3	Oct 2013		86	86	8	21	534	650	857	1,421
Fake Plugin	1,121	1,336	1.2	Jul 2014		0	0	1,121	1,336	0	0	0	0
Func Reconst	636	929	1.5	Jan 2016		3	3	52	54	12	13	579	890
Insert User	357	1,531	4.3	Dec 2015		0	0	266	266	2	6	292	1,490
Post Injection	281	1,407	5.0	May 2016		0	0	266	266	1	1	315	1,415
Malvertising	915	1,354	1.5	May 2017		12	13	0	0	894	1,330	13	13
SSO Backdoor	191	905	4.7	May 2019		2	2	0	0	36	91	190	879
Cryptominer	4	4	1.0	Jul 2018		0	0	4	4	0	0	0	0
Total[2]	24,931	47,337	1.9	Jul 2012		1,345	3,685	1,201	2,814	5,244	8,525	18,034	40,533

1: Infection Ratio (IR) is the ratio of #P to #W, shows a measure of infection spread.
2: This is not the sum of the columns, but the total #websites and #plugins with malicious behaviors in our dataset.

Table 6: Distribution and Temporal Evolution of the Malicious Behaviors Across all Websites in our Dataset.

Dating back to 2012, we studied the evolution of these malicious behaviors. Since the absolute number of websites in our dataset increased over time, in Table 6 Temporal Evolution, we plot the newly infected websites as a percentage of all malicious websites to remove dataset bias. While some attack behaviors were popular since late 2012, other behaviors such as spam injection (2016), malvertising (2017), and SSO backdoor (2019) were introduced recently. However, it is interesting to note that regardless of when they were first introduced, all of these behaviors are still prevalent in present-day malicious plugins. A closer look at the absolute values of the newly introduced malicious behaviors reveals that the number of malicious plugins peaked in March 2020, which notably coincides with the COVID-19 outbreak.

Thousands of malicious plugins originated from legitimate plugin marketplaces. Table 6's Marketplace Columns show their distribution (i.e., number of websites #W and number of plugins #P with malicious behaviors). Row 2 shows that *none of these plugins use code obfuscation techniques* — despite being sold on legitimate marketplaces they brazenly hide in plain sight. Attackers (rightly) assume that an average website owner will not inspect the plugin code before installing it on their webserver. In fact, we found instances of well commented malicious code in 2,379 of the 3,452 plugins that performed spam injection originating from legitimate plugin marketplaces. Evidently, these plugins enabled illegal monetization via blackhat SEO, downloader, and spam injection in 86, 288, and 3,452 plugin instances, respectively.

Attackers exploited the scalable CMS infrastructure to inject malicious plugins into websites. Table 6's Injected Columns show that the injected plugins aim to gain and maintain access to the webserver. They are injected without the website owners' knowledge and over 80% of these plugins had fake plugin behaviors (1,336), webshells (994), or obfuscated code (558). Although cryptomining is gaining popularity, we only found 4 injected cryptominer plugins on 4 websites revealing its infancy in pervading the CMS landscape.

We found 8,525 malicious nulled plugin instances in our dataset that exploit human vulnerabilities to rapidly spread malware. Table 6's Nulled Columns show that over 91% (7,821 of 8,525) of these plugin instances used input gating (i.e., password-protecting the publicly accessible code) to thwart competing attackers from introducing malicious payloads. We also found that the plugins introduced after December 2018 primarily employed downloader, blackhat SEO, and malvertizing behaviors in 2,683, 650, and 1,330 plugin instances, respectively, to infect other benign plugins.

It was concerning that over 40K plugin instances were infected post-deployment. Table 6's Infected Columns show that these plugins portray a variety of malicious behaviors. Most attackers employ behaviors such as webshells, obfuscation, and downloaders in 9,943, 8,819, and 4,254 plugin instances, respectively. Interestingly, over 50% (20,684 of 40,533) of these plugins employed input gating showing attackers' diligence in marking their conquered territories.

5.2 Fueling the Malware Economy

Next, we turned our attention to the economic drivers of these malicious plugins. Table 7 categorizes our results based on the origin of malicious behaviors, i.e., legitimate marketplaces, nulled marketplaces, and infected plugins.

Table 7 begins with malicious plugins originating from legitimate plugin marketplaces. About 70% of these (2,597 of 3,685) were found on 5 of the 7 most popular marketplaces. Our dataset alone constituted over $41K in purchases of malicious plugins from

Markeplace	Malicious			Downloads Range			Cost
	#P	#U[1]	%M[2]	Min.	Avg.	Max.	
Legitimate Marketplace							
WP Themes	523	62	1.1%	7.7K	336K	3.6M	-
WP Plugins	1,583	69	0.25%	4	945K	25M	-
Github	0	0	0%	-	-	-	-
WPMU DEV	132	2	1.8%	54K	510K	524K	$25.8K
CodeCanyon	164	10	0.4%	1	40	73	$6.8K
ThemeForest	195	22	0.37%	9	20K	213K	$8.9K
EDD	0	0	0%	-	-	-	$0
Subtotal	2,597	165	0.38%	-	-	-	**$41.5K**
Nulled Plugins							
WP Themes	1,074	59	1.08%	11K	203K	5.7M	-
WP Plugins	146	43	0.16%	65	4K	37K	-
Github	0	0	0%	-	-	-	-
WPMU DEV	4	1	0.9%	572K	572K	572K	$2.3K
CodeCanyon	2,085	122	5.02%	1	70	570	$82.3K
ThemeForest	3,059	223	3.82%	3	12K	213K	$142K
EDD	39	3	1.2%	-	-	-	$1.3K
Subtotal	6,407	451	1.03%	-	-	-	**$228K**
Infected Plugins							
WP Themes	9,776	1,864	34.2%	1	367K	7.4M	-
WP Plugins	8,049	6,520	23.8%	2	4M	260M	-
Github	15	1	0.01%	2	2	2	-
WPMU DEV	450	9	8.2%	187K	2M	10.5M	$88.2K
CodeCanyon	1,873	469	19.3%	1	62	563	$59.9K
ThemeForest	5,858	1,072	18.4%	2	10K	213K	$264K
EDD	634	57	23.3%	-	-	-	$422K
Subtotal	26,655	9,992	22.9%	-	-	-	**$834K**

1: #U: Number of unique malicious plugins, 2: %M: Percentage of the plugins on the marketplace that were flagged as malicious

Table 7: The Economy of Malicious Plugin Marketplaces.

legitimate marketplaces. We found 62 unique malicious plugins from WP Themes and 69 from WP Plugins (unpaid marketplaces), contributing to 1.1% and 0.25% of these marketplaces, respectively (%M Column). Furthermore, the malicious plugins from these marketplaces are extremely popular, averaging 336K and 945K downloads per plugin. We also found 34 unique malicious plugins sold on paid marketplaces.

Nulled plugins impersonate plugins from legitimate marketplaces. YODA extracts their popularity and cost from legitimate marketplaces. The cost represents the explicit losses incurred by the legitimate plugin authors. About 75% of the malicious nulled plugins (i.e., 6,407 of 8,525) in our dataset contain legitimate counterparts in these 7 popular marketplaces. Since nulled marketplaces distribute plugins free of cost, we did not expect to find plugins from unpaid marketplaces. Surprisingly, we found a total of 102 plugins from WP Plugins and WP Themes sold on nulled marketplaces. As expected, we also found that over 77% (349 of 451) of the nulled plugin counterparts were sold on paid marketplaces. Attackers impersonated 122 and 223 best-selling plugins from CodeCanyon and ThemeForest, respectively. Overall, the website owners from our dataset alone contributed to $228K in explicit loss to the plugin authors. This shows that attackers are successfully targeting psychological human vulnerabilities and the less-technical CMS users are installing pirated plugins.

Finally, Table 7 considers the origin of post-deployment infected plugins. About 65% of the infected plugins (i.e., 26,655 of 40,533) were downloaded from these 7 popular marketplaces. Since the plugins from WP Plugins and WP Themes are widely used, they are also commonly infected. 34.2% and 23.8% of plugins from WP Themes and WP Plugins became victims of plugin infections. Despite paying a premium for plugins from paid marketplaces, a significant number of these were found to be infected, i.e., 8.2% of WPMU DEV, 19.3% of CodeCanyon, 18.4% of ThemeForest, and 23.3% of EDD. The website owners spent a total of $834K on these plugins, only to find them compromised. This encapsulates the additional implicit cost of malware cleanup incurred by installing malicious plugins from legitimate and nulled marketplaces.

5.3 Nulled Marketplace Study

Since nulled plugins require some tampering with the WordPress backend (too complex for the typical CMS user), we did not expect to find many nulled plugin instances in our dataset. Surprisingly, Table 8 reveals 6,223 websites had at least one nulled plugin. We found that these plugins are gaining popularity by optimizing for search engine ranking (discussed in Appendix D).

Table 8 shows the nulled marketplaces extracted from the plugin metadata. If YODA identifies a plugin as nulled based on its behavior alone and if it cannot extract a nulled marketplace from the plugin metadata, we categorize the marketplace as 'Unknown'. Table 8 shows `vestathemes.com` as the most popular nulled plugin marketplace in our dataset, with 3,177 plugin instances (#P) downloaded across 2,398 websites (#W). Recall from §3.3, not all nulled plugins portray malicious behaviors. Columns 4-5 present the number of websites (#MW) containing malicious nulled plugins (#MP). Overall, over 97% of all nulled plugins deliver malicious behaviors (%M). In particular, 100% of the plugins we saw from `theme123.net`, `themelot.net`, and 'Unknown' marketplace were malicious.

Interestingly, the 'Unknown' marketplaces have distributed over 31% of all malicious nulled plugins (2,603 of 8,525) in our dataset. They impersonate the plugin author entirely and hide that they were downloaded from a nulled source as opposed to the other nulled marketplaces in Column 1. A comparison of the plugin header from an 'Unknown' nulled plugin and a legitimate marketplace plugin did not reveal any differences. Only after matching the code files were we able to tell a nulled malicious plugin apart from the legitimate plugin. However, since the website owners cannot compare the nulled plugin's code to the paid

Nulled Marketplace	#W	#P	#MW[1]	#MP[2]	%M	1^{st} Seen	1^{st} Mal. Seen	Popular Mal. Plg.	Cost	#Instances
vestathemes.com	2,398	3,177	2,283	3,057	96.2%	Aug 2014	Jul 2018	Flatsome [39]	$59	195
www.themes24x7.com	989	1,363	928	1,282	94.1%	Mar 2016	Mar 2016	WPBakery Page Builder [40]	$64	140
www.wplocker.com	841	1,035	829	1,017	98.3%	Nov 2013	Jan 2014	FormCraft [41]	$36	80
www.jojo-themes.net	133	141	109	117	82.9%	Feb 2016	Feb 2016	Gravity Forms [42]	$59	9
theme123.net	127	144	127	144	100%	Dec 2013	Dec 2013	WP Robot 5 [43]	$89	9
mafiashare.net	121	149	117	142	95.3%	Jan 2014	Dec 2015	BeTheme [44]	$59	9
www.wptry.org	79	99	79	98	98.9%	Apr 2020	Apr 2020	Woodmart [45]	$59	3
themlot.net	60	65	60	65	100%	Dec 2014	Dec 2014	BeTheme [44]	$59	4
Unknown	1,906	2,603	1,906	2,603	100%	Oct 2016	Oct 2016	Flatsome [39]	$59	252
Total[1]	6,223	8,776	5,244	8,525	97.1%	Nov 2013	Dec 2013	Flatsome [39]	$59	483

1: #MW: The number of websites with malicious nulled plugins, 2: #MP: The number of malicious nulled plugins.

3: The total here is not the sum of the columns, but the total #W and #P from nulled marketplaces in our dataset.

Table 8: Study of Malicious Plugins From Nulled Marketplaces.

Malicious Origin	Malicious		Cleaned Up			Reinfected			Still Infected		
	#W	#P	#W	% W	#P	#W	% W	#P	#W	% W	#P
Marketplace	1,345	3,685	324	24.1%	389	32	9.9%	32	1,090	81.0%	3,327
Injected	1,201	2,814	169	14.1%	221	21	12.4%	41	1,067	88.8%	2,608
Nulled	5,244	8,525	353	6.7%	471	63	17.8%	81	5,003	95.4%	8,115
Infected	18,034	40,533	2,174	12.1%	5,962	254	11.7%	551	16,881	93.6%	34,956
Total[1]	24,931	47,337	2,697	10.8%	7,042	336	12.5%	705	23,577	94.6%	40,787

1: The total here is not the sum of the columns, but the total #W and #P in our dataset.

Table 9: The Cleanup and Reinfection Distribution of Malicious Plugins.

legitimate plugin, it is impossible for them to identify the nulled plugin as malicious. However, YODA can detect malicious plugins by only analyzing its code files.

Table 8 also shows that most nulled marketplaces have been around for a long time, since 2013-2014. However, over 50% of the marketplaces displayed malicious behaviors starting in 2016. In particular, the 'Unknown' marketplaces have attempted to spread malware since 2016 and have been successful through 2020. The most popular nulled plugins cost between $36 and $89, with an average of $59 per plugin. 447 of the 483 popular malicious nulled plugin instances were Flatsome [39], a WordPress theme that would normally cost the website owner $59 and provided pre-defined layouts for user-friendly e-commerce shop features.

5.4 Are Infected Plugins Cleaned Up?

Lastly, Table 9 studies the plugin clean-up statistics to understand how attackers are evading website owners. Very few website owners (2,697 of 24,931 or 10.8% of the compromised websites overall) attempt to clean up the malicious plugins on their webserver. We hypothesize that those website owners are unaware of the malicious plugins or they cannot correlate malware side-effects (such as server slowdown) with the plugins. As seen in Table 9, 24.1% of websites with malicious plugins from legitimate marketplaces are cleaned up, the highest rate by far. Only 6.7% of nulled plugins are cleaned up, which further strengthens our hypothesis (§6) that despite much later adoption, nulled plugins provide robust persistence for attackers.

Of the 2,697 websites that attempted to clean up 7,042 malicious plugins, 12.5% of the websites (336 of 2,697) were reinfected. Interestingly, nulled plugins were most consistently reinfected (17.8% or 63 of 353 websites). Plugins downloaded from legitimate marketplaces show the least rate of reinfection (9.9%). This can be attributed to community engagement in identifying malicious plugins on legitimate marketplaces. Such plugins are either purged from the marketplace or their authors are forced to remove the malicious code.

Lastly, we measured the websites that remained infected up to the time of writing. Despite cleanup efforts, over 94% of all websites with malicious plugins remained infected. This proves that CMS plugins have provided a reliable webserver infiltration vector for nearly a decade.

6 Persistence of Malicious Plugins

To understand the persistence patterns of malicious plugins, Figure 2 shows a box plot measuring the number of days malicious plugins were identified on the webserver, categorized by their origin. The median persistence ranges between 189-209 days. Thus, over

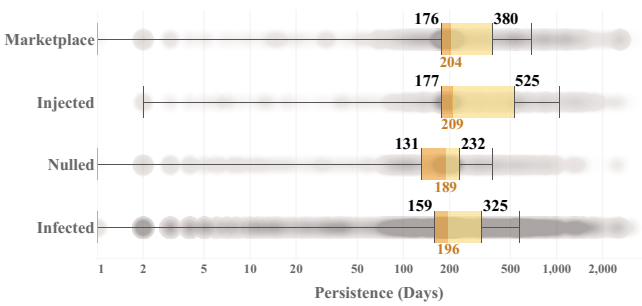

Figure 2: Persistence of Malicious Plugins.

50% of the malicious plugins persist for over 6 months. We also noted that over 80% of the remaining malicious plugins (those that persisted for less than 6 months) were introduced during Feb - Mar 2020 and persisted through the end of our study. This confirms our previous observation (Table 9) that 94% of the malicious plugins in our dataset installed over 8 years are still active today.

Popular plugins on legitimate marketplaces mostly introduce malicious behaviors via plugin updates. Thus, we assumed that these behaviors would be cleaned up with updates[8] as well. As seen from Figure 2, malicious plugins from legitimate marketplaces are not immediately identified at source and persist for 176 - 380 days. Recall from §4.1, over 60% of the website owners do not enable auto-updates and use outdated plugin versions. If these website owners happen to install a malicious version of a plugin from a legitimate marketplace, it persists for months or years.

Figure 2 also shows that the persistence of nulled plugins (131 - 232 days) is shorter compared to other origins. This can be attributed to the fact that even though nulled marketplaces existed since 2013, they gained popularity around 2018, and their blackhat SEO campaigns accelerated in early 2019. We found that once nulled plugins are installed on the webserver, they are rarely removed (§5.4). The website remains compromised since the website owner is unaware of the plugin's malicious intentions. So despite much later adoption, 25% of these plugins persist for over 232 days.

Notably, it is the injected plugins that win the persistence war. Over 75% of these plugins remain active for at least 177 days, and over 25% of these plugins persist for at least 525 days. This proves that injected plugins are never noticed by the website owners, who typically use GUIs to manage their CMS.

7 Case Studies

1. Malvertizing URLs. Discovered in 2019, the largest known malvertizing campaign downloaded content from malicious domains in plugins to the webserver [15]. To understand the lifecycle of these domains, we extracted 352 URLs from all malvertizing plugins in our dataset and analyzed the domain creation date, last updated date (i.e., registration renewed), and their registrars. Figure 3 shows the distribution of URLs created and updated over time. The majority of these URLs were created in 2018, but attackers are re-registering the same URLs with peak activity in 2021. Thus, the malvertizing campaign is still active (confirmed by their use in recent malicious plugins) and has evaded detection. In fact, only 56 of these URLs were no longer registered at the time of writing. We believe that these were throwaway URLs generated for a short targeted attack. We also found 38 of these URLs captured by the internet archive [46], further supporting our hypothesis.

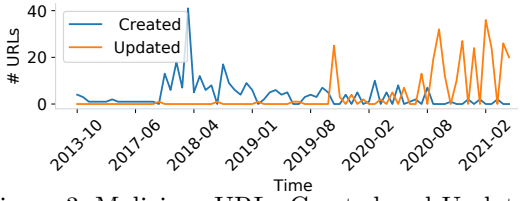

Figure 3: Malicious URLs Created and Updated.

2. Spam Injection Insights. Starting in 2016, a prolific spammer bought and updated several WordPress plugins for a coordinated spam campaign over a 4.5-year period [14]. Hoping to find how widespread this campaign was among our dataset, we drilled down into the spam injection plugins YODA detected. Apart from downloading malicious spam content from the spammer's own domains onto the webserver, we discovered these same plugins also collected details on visitors to the infected website, such as URL, IP, user agent, and other attacker-defined variables. Of the 3,723 spam injection plugins, 94% sent back IP and user agent using the PHP superglobal variables. Of these, 66% encoded the IP using `$_SERVER['REMOTE_ADDR']` and 34% used `$_SERVER['SERVER_ADDR']`. These plugins also sent attacker-defined variables `'p'` (set to 2, 29, or 9) and `'v'` (set to 11 or 18). While we cannot accurately decipher what these variables mean, we speculate that they identify the spammer's campaign and distribute profits similar to affiliate tracking. 6% of the plugins did not send any data back to the attacker. Interestingly, these where all the earliest cases that appeared in late 2016.

8 Limitations and Future Work

Additional CMS Platforms. YODA can accurately detect malicious plugins on WordPress-based websites. However, scaling to other CMSs only requires updating YODA's plugin detection and semantic models. YODA's modularity enables porting to other platforms by reviewing the API documentation of the target CMS.

[8]The marketplace takes down community-identified malicious plugins or mandates reverting the malicious behaviors.

We leave handling other CMSs as future work.

Static Behavior Detection. Since YODA relies on static analysis, it carries the limitations of static analysis. Like any static data flow analysis framework, YODA can identify obfuscated code but it cannot detect malicious behaviors within the obfuscated code. This is further discussed in Appendix C. YODA could be augmented with dynamic analysis [47], [48] to achieve better coverage of dynamic PHP code.

Semantic Model Evasion. Since the semantic models rely on data flows, they cannot be evaded by rearranging code, inserting junk code, or splitting the attack behavior across files. That said, attackers can try to evade YODA in two ways: (1) evolve to entirely new behaviors (e.g., the recently introduced SSO Backdoor behavior) or (2) novel implementations of known attack behaviors (via new PHP APIs). Such evolution is expected, and in both cases, new semantic models can be crafted for the new data source-sink combinations.

9 Related Work

Web Attacks. Past research studied web attacks as seen from the web browser [8], [49]–[53]. Other studies used webserver backups [9], [54] and high-interaction honeypots [55]–[57] to understand web attacks. Several techniques studied the role of hosting providers [58] and the response landscape from post-compromise notification campaigns [59], [60]. While these studies focused on generic web attacks, YODA analyzed the spread of malware via CMS plugins.

Our previous work, TARDIS [9] also analyzed nightly backups to investigate targeted long-lived malware at an entire-website granularity, but TARDIS is neither proactive nor fine-grained enough to vet previously-unseen plugins for malicious behavior. TARDIS's detection is coarse-grained as it relies upon strict temporal sequences of website-level indicators (e.g., stand-alone backdoor file injection followed by file deletion). Malicious plugins do not exhibit overt temporal sequences of such indicators (that TARDIS relies upon). They are deployed all at once and lie in wait until the website is loaded (e.g., blackhat SEO), requiring a plugin-centric detection and analysis.

Web Malware Analysis. Recent web-based malware analysis research analyzed targeted attack classes like webshells [33], [34], ad injection [61]–[63], survey scams [64], [65], cross-site scripting [66]–[69], PHP code and SQL injection [69]–[74], file inclusion attacks [75], [76], dictionary attacks [77], etc. Their adoption by website operators to detect malicious CMS plugins is limited by significant instrumentation and training complexities associated with these techniques. Conversely, YODA is an automated investigation framework, agnostic to targeted attack classes, and can be deployed by all stakeholders in the CMS ecosystem.

Measurement Studies. WordPress plugin research focused on measuring vulnerabilities [78]–[80] and comparing plugin ratings with vulnerability exploits [81]. Researchers also assessed the role of web hosting providers to detect compromised websites [82], studied malicious web apps [2], malicious browser extensions [3], [4], and malicious packages in package registries [5]. Caballero et. al. [83] measured pay-per-install malware distribution in benign software. However, unlike YODA, prior work has not studied the impact of malicious plugins on CMS marketplaces.

10 Conclusion

YODA provides an automated investigation framework that uncovered 47,337 malicious plugin installs on 24,931 unique websites, 94% of which *are still active today*. We have disclosed the results to CodeGuard and they are working on remediating the identified attacks.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments and feedback. In particular, we thank Professor Alexandros Kapravelos for his guidance while shepherding this paper as well as our collaborators at CodeGuard for their insightful comments and suggestions throughout this research. This work was supported, in part, by NSF under Award 1916550, DARPA under contract HR00112190087, and Cisco Systems under an unrestricted gift. Any opinions, findings, and conclusions in this paper are those of the authors and do not necessarily reflect the views of our sponsors or collaborators.

References

[1] *Is WordPress Really A 10 Billion Dollar Economy?* https://www.presstitan.com/is-wordpress-really-a-10-billion-dollar-economy/, [Accessed: 2020-05-08].

[2] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, "Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets.," in *Proc. 19th NDSS*, San Diego, CA, Feb. 2012.

[3] N. Jagpal, E. Dingle, J.-P. Gravel, P. Mavrommatis, N. Provos, M. A. Rajab, and K. Thomas, "Trends and lessons from three years fighting malicious extensions," in *Proc. 24th USENIX Sec.*, Washington, DC, Aug. 2015.

[4] A. Kapravelos, C. Grier, N. Chachra, C. Kruegel, G. Vigna, and V. Paxson, "Hulk: Eliciting malicious behavior in browser extensions," in *Proc. 23rd USENIX Sec.*, San Diego, CA, Aug. 2014.

[5] R. Duan, O. Alrawi, R. Pai Kasturi, R. Elder, B. Saltaformaggio, and W. Lee, "Towards Measuring Supply Chain Attacks on Package Managers," in *Proc. 2021 NDSS*, San Diego, CA, Feb. 2021.

[6] K. Soska and N. Christin, "Automatically detecting vulnerable websites before they turn malicious," in *Proc. 23rd USENIX Sec.*, San Diego, CA, Aug. 2014.

[7] J. Dahse and T. Holz, "Static detection of second-order vulnerabilities in web applications," in *Proc. 23rd USENIX Sec.*, San Diego, CA, Aug. 2014.

[8] L. Invernizzi, P. M. Comparetti, S. Benvenuti, C. Kruegel, M. Cova, and G. Vigna, "Evilseed: A guided approach to finding malicious web pages," in *Proc. 33rd IEEE S&P*, San Francisco, CA, May 2012.

[9] R. P. Kasturi, Y. Sun, R. Duan, O. Alrawi, E. Asdar, V. Zhu, Y. Kwon, and B. Saltaformaggio, "TARDIS: Rolling Back The Clock On CMS-Targeting Cyber Attacks," in *Proc. 41st IEEE S&P*, Online Conference, May 2020.

[10] *Is That WordPress Plugin Safe? 15 Warning Signs to Skip Downloading*, https://premium.wpmudev.org/blog/is-that-wordpress-plugin-safe-15-warning-signs-to-skip-downloading/, [Accessed: 2020-08-19].

[11] *WPScan Vulnerability Database*, https://wpvulndb.com/plugins, [Accessed: 2020-05-09].

[12] *RIPSTECH Vulnerability Detection*, https://www.ripstech.com/, [Accessed: 2020-05-09].

[13] *WebARX: Protect websites from plugin vulnerabilities*, https://www.webarxsecurity.com/, [Accessed: 2020-05-09].

[14] *9 WordPress Plugins Targeted in Coordinated 4.5-Year Spam Campaign*, https://www.wordfence.com/blog/2017/09/coordinated-plugin-spam/, [Accessed: 2020-05-08].

[15] *WP-VCD: The Malware You Installed On Your Own Site*, https://www.wordfence.com/wp-content/uploads/2019/11/Wordfence-WP-VCD-Whitepaper.pdf, [Accessed: 2020-06-27].

[16] *WordPress Themes*, https://wordpress.org/plugins/, [Accessed: 2020-08-08].

[17] *WordPress Themes*, https://wordpress.org/themes/, [Accessed: 2020-08-08].

[18] *Github*, https://github.com/, [Accessed: 2020-08-08].

[19] *CodeCanyon*, https://codecanyon.net/, [Accessed: 2020-08-08].

[20] *ThemeForest*, https://themeforest.net/, [Accessed: 2020-08-08].

[21] *WPMU DEV*, https://premium.wpmudev.org/, [Accessed: 2020-08-08].

[22] *Easy Digital Downloads*, https://easydigitaldownloads.com/, [Accessed: 2020-08-08].

[23] *The 10 Million Dollar Plugin: Choco Drops + Admob (Android Studio + Eclipse) Easy Reskin*, https://codecanyon.net/item/choco-drops-admob-android-studio-eclipse-easy-reskin/21423129, [Accessed: 2020-07-29].

[24] *PLR Products*, https://www.plrproducts.com/keyword-swarm-wp-plugin/, [Accessed: 2020-08-18].

[25] *Jeroen Sormani - WordPress Plugin Developer*, https://jeroensormani.com/, [Accessed: 2020-08-20].

[26] *VaultPress - Daily and Real-time Backups*, https://vaultpress.com/, [Accessed: 2020-08-20].

[27] *WordPress File Permissions: Complete Beginner's Guide*, http://www.themeshunter.com/stats/themeforest-net.html, [Accessed: 2021-09-09].

[28] *Why You Should Stop Using Nulled WordPress Plugins and Themes*, https://kinsta.com/blog/nulled-wordpress-plugins-themes/, [Accessed: 2020-10-07].

[29] M. Backes, S. Bugiel, and E. Derr, "Reliable third-party library detection in android and its security applications," in *Proc. 23rd ACM CCS*, Vienna, Austria, Oct. 2016.

[30] *Display Widgets Plugin Permanently Removed from WordPress.org Due to Malicious Code*, https://wptavern.com/display-widgets-plugin-permanently-removed-from-wordpress-org-due-to-malicious-code, [Accessed: 2020-10-12].

[31] *Abandoned and Removed Plugin Alerts*, https://www.wordfence.com/blog/2017/06/abandoned-removed-plugin-alerts/, [Accessed: 2020-10-12].

[32] *SweetCAPTCHA Service Used to Distribute Adware*, https://blog.sucuri.net/2015/06/sweetcaptcha-service-used-to-distribute-adware.html, [Accessed: 2020-10-12].

[33] Y. Li, J. Huang, A. Ikusan, M. Mitchell, J. Zhang, and R. Dai, "Shellbreaker: Automatically detecting php-based malicious web shells," *Computers & Security*, vol. 87, p. 101 595, 2019.

[34] H. Cui, D. Huang, Y. Fang, L. Liu, and C. Huang, "Webshell detection based on random forest–gradient boosting decision tree algorithm," in *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, IEEE, 2018, pp. 153–160.

[35] N. Popov, *A PHP parser written in PHP*, Jul. 2019. [Online]. Available: https://github.com/nikic/PHP-Parser.

[36] V. Total, "Virustotal-free online virus, malware and url scanner," *Online: https://www. virustotal.com/en*, 2012.

[37] *URLHaus online virus, malware and url scanner, author=URLHaus*, https://www.urlhaus.com, [Accessed: 2021-26-04].

[38] *CMS Market Share*, https://w3techs.com/technologies/overview/content_management, [Accessed: 2020-06-12].

[39] *Flatsome | Multi-Purpose Responsive WooCommerce Theme*, https://themeforest.net/item/flatsome-multipurpose-responsive-woocommerce-theme/5484319, [Accessed: 2020-09-23].

[40] *WPBakery Page Builder for WordPress*, https://codecanyon.net/item/visual-composer-page-builder-for-wordpress/242431, [Accessed: 2020-09-23].

[41] *FormCraft - Premium WordPress Form Builder*, https://codecanyon.net/item/formcraft-premium-wordpress-form-builder/5335056, [Accessed: 2020-09-23].

[42] *Gravity Forms*, https://www.gravityforms.com/pricing/, [Accessed: 2020-09-23].

[43] *WP Robot 5 - Your Blog On Autopilot*, https://wprobot.net/order/, [Accessed: 2020-09-23].

[44] *BeTheme - Responsive Multi-Purpose WordPress Theme*, https://themeforest.net/item/betheme-responsive-multipurpose-wordpress-theme/7758048, [Accessed: 2020-09-23].

[45] *WoodMart - Responsive WooCommerce WordPress Theme*, https://themeforest.net/item/woodmart-woocommerce-wordpress-theme/20264492, [Accessed: 2020-09-23].

[46] *Internet Archive: Wayback Machine*, https://archive.org/web/, [Accessed: 2021-05-10].

[47] P. Wrench and B. Irwin, "A sandbox-based approach to the deobfuscation and dissection of php-based malware," *Saiee Africa Research Journal*, 2015.

[48] A. Naderi-Afooshteh, Y. Kwon, A. Nguyen-Tuong, M. Bagheri-Marzijarani, and J. W. Davidson, "Cubismo: Decloaking server-side malware via cubist program analysis," in *Proc. 35th ACSAC*, 2019.

[49] "To Get Lost is to Learn the Way: Automatically Collecting Multi-step Social Engineering Attacks on the Web," in *Proc. 15th ACM ASIACCS*, Taipei, Taiwan, Oct. 2020.

[50] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad, "Webwitness: Investigating, categorizing, and mitigating malware download paths," in *Proc. 24th USENIX Sec.*, Washington, DC, Aug. 2015.

[51] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, N. Modadugu, *et al.*, "The ghost in the browser: Analysis of web-based malware.," *HotBots*, vol. 7, pp. 4–4, 2007.

[52] A. Sudhodanan, S. Khodayari, and J. Caballero, "Cross-Origin State Inference (COSI) Attacks: Leaking Web Site States through XS-Leaks," in *Proc. 2020 NDSS*, San Diego, CA, Feb. 2020.

[53] C.-A. Staicu and M. Pradel, "Leaky images: Targeted privacy attacks in the web," in *Proc. 28th USENIX Sec.*, Santa Clara, CA, Aug. 2019.

[54] A. Naderi-Afooshteh, Y. Kwon, A. Nguyen-Tuong, A. Razmjoo-Qalaei, M.-R. Zamiri-Gourabi, and J. W. Davidson, "Malmax: Multi-aspect execution for automated dynamic web server malware analysis," in *Proc. 26th ACM CCS*, London, United Kingdom, Nov. 2019.

[55] O. Starov, J. Dahse, S. S. Ahmad, T. Holz, and N. Nikiforakis, "No honor among thieves: A large-scale analysis of malicious web shells," in *Proc. 25th WWW*, 2016.

[56] O. Catakoglu, M. Balduzzi, and D. Balzarotti, "Automatic extraction of indicators of compromise for web applications," in *Proc. 25th WWW*, 2016.

[57] D. Canali and D. Balzarotti, "Behind the scenes of online attacks: An analysis of exploitation behaviors on the web," in *Proc. 20th NDSS*, San Diego, CA, Feb. 2013.

[58] D. Canali, D. Balzarotti, and A. Francillon, "The role of web hosting providers in detecting compromised websites," in *Proc. 22nd WWW*, Rio de Janeiro, Brazil, May 2013.

[59] B. Stock, G. Pellegrino, C. Rossow, M. Johns, and M. Backes, "Hey, you have a problem: On the feasibility of large-scale web vulnerability notification," in *Proc. 25th USENIX Sec.*, Austin, TX, Aug. 2016.

[60] F. Li, Z. Durumeric, J. Czyz, M. Karami, M. Bailey, D. McCoy, S. Savage, and V. Paxson, "You've got vulnerability: Exploring effective vulnerability notifications," in *Proc. 25th USENIX Sec.*, Austin, TX, Aug. 2016.

[61] S. Arshad, A. Kharraz, and W. Robertson, "Identifying extension-based ad injection via fine-grained web content provenance," in *Proc. 19th RAID*, Evry, France, Sep. 2016.

[62] K. Thomas, E. Bursztein, C. Grier, G. Ho, N. Jagpal, A. Kapravelos, D. McCoy, A. Nappa, V. Paxson, P. Pearce, N. Provos, and M. Abu Rajab, "Ad injection at scale: Assessing deceptive advertisement modifications," in *Proc. 36th IEEE S&P*, San Jose, CA, May 2015.

[63] X. Xing, W. Meng, B. Lee, U. Weinsberg, A. Sheth, R. Perdisci, and W. Lee, "Understanding malvertising through ad-injecting browser extensions," in *Proc. 24th WWW*, 2015.

[64] A. Kharraz, W. Robertson, and E. Kirda, "Surveylance: Automatically detecting online survey scams," in *Proc. 39th IEEE S&P*, San Francisco, CA, May 2018.

[65] J. W. Clark and D. McCoy, "There are no free ipads: An analysis of survey scams as a business.," in *Proc. 6th USENIX LEET*, Washington, D.C., United States, Aug. 2013.

[66] W. Melicher, A. Das, M. Sharif, L. Bauer, and L. Jia, "Riding out domsday: Toward detecting and preventing dom cross-site scripting," in *Proc. 2018 NDSS*, San Diego, CA, Feb. 2018.

[67] B. Stock, S. Lekies, T. Mueller, P. Spiegel, and M. Johns, "Precise client-side protection against dom-based cross-site scripting.," in *Proc. 2014 NDSS*, San Diego, CA, Feb. 2014.

[68] G. Wassermann and Z. Su, "Static detection of cross-site scripting vulnerabilities," in *Proc. 30th International Conference on Software Engineering (ICSE)*, Leipzig, Germany, May 2008.

[69] M. Backes, K. Rieck, M. Skoruppa, B. Stock, and F. Yamaguchi, "Efficient and flexible discovery of php application vulnerabilities," in *Proc. European Symposium on Security and Privacy (EuroS&P)*, IEEE, Paris, France, Apr. 2017.

[70] D. R. Sahu and D. S. Tomar, "DNS pharming through PHP injection: Attack scenario and investigation," *IJ Computer Network and Information Security*, vol. 4, pp. 21–28, 2015.

[71] V. Yerram and G. V. R. Reddy, "A solution to php code injection attacks and web vulnerabilities," 2014.

[72] Z. S. Alwan and M. F. Younis, "Detection and prevention of sql injection attack: A survey," *International Journal of Computer Science and Mobile Computing*, vol. 6, no. 8, pp. 5–17, 2017.

[73] N. Singh, M. Dayal, R. Raw, and S. Kumar, "Sql injection: Types, methodology, attack queries and prevention," in *Proc. 3rdComputing for Sustainable Global Development (INDIACom)*, IEEE, New Delhi, India, Mar. 2016.

[74] A. Pramod, A. Ghosh, A. Mohan, M. Shrivastava, and R. Shettar, "Sqli detection system for a safer web application," in *Proc. 2015IEEE International Advance Computing Conference*, IEEE, Bangalore, India, Jun. 2015.

[75] H. F. G. Robledo, "Types of hosts on a remote file inclusion (rfi) botnet," in *Proc. Electronics, Robotics and Automotive Mechanics Conference*, Jun. 2008.

[76] O. Katz, "Detecting remote file inclusion attacks," *White Paper. Breach Security Inc., May*, 2009.

[77] A. K. Kyaw, F. Sioquim, and J. Joseph, "Dictionary attack on wordpress: Security and forensic analysis," in *2015 Second International Conference on Information Security and Cyber Forensics (InfoSec)*, 2015.

[78] J. Ruohonen, "A demand-side viewpoint to software vulnerabilities in wordpress plugins," in *Evaluation and Assessment on Software Engineering*, 2019.

[79] O. Mesa, R. Vieira, M. Viana, V. H. Durelli, E. Cirilo, M. Kalinowski, and C. Lucena, "Understanding vulnerabilities in plugin-based web systems: An exploratory study of wordpress," in *22nd International Systems and Software Product Line Conference*, 2018.

[80] I. Cernica, N. Popescu, and B. ţigănoaia, "Security evaluation of wordpress backup plugins," in *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, 2019.

[81] T. Koskinen, P. Ihantola, and V. Karavirta, "Quality of wordpress plug-ins: An overview of security and user ratings," in *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, 2012.

[82] D. Canali, D. Balzarotti, and A. Francillon, "The role of web hosting providers in detecting compromised websites," in *Proc. 22nd WWW*, Rio de Janeiro, Brazil, May 2013.

[83] J. Caballero, C. Grier, C. Kreibich, and V. Paxson, "Measuring pay-per-install: The commoditization of malware distribution.," in *Proc. 20th USENIX Sec.*, San Francisco, CA, Aug. 2011.

[84] *WordPress File Permissions: Complete Beginner's Guide*, https://www.malcare.com/blog/wordpress-file-permissions/, [Accessed: 2021-06-03].

[85] *DooPlay Theme WordPress*, https://www.downloadfreethemes.co/dooplay-v2-1-3-8-movies-and-tv-shows-wordpress-theme/, [Accessed: 2020-06-27].

A WordPress Plugin Header

```
/**
 * Plugin Name:       My Basics Plugin
 * Plugin URI:        https://example.com/plugins/the-basics/
 * Description:       Handle the basics with this plugin.
 * Version:           1.10.3
 * Requires at least: 5.2
 * Requires PHP:      7.2
 * Author:            John Smith
 * Author URI:        https://author.example.com/
 * License:           GPL v2 or later
 * License URI:       https://www.gnu.org/licenses/gpl-2.0.html
 * Text Domain:       my-basics-plugin
 * Domain Path:       /languages
 */
```

Figure 4: A Typical WordPress Plugin Header.

A plugin header (shown in Figure 4) is a specially formatted PHP/CSS block comment that contains metadata about the plugin, such as its name, author, version, license, etc. At the very least, the header comment must contain the Plugin Name. An attacker could try to evade YODA's plugin detection by dropping the plugin header. However, this would work against the attacker: without a valid plugin header containing at least the Plugin Name metadata, the plugin will never get loaded by WordPress core and hence remain dormant on the webserver.

B WordPress Plugin Architecture

```
has_filter, add_filter, apply_filters,
apply_filters_ref_array, do_action_ref_array,
remove_filter, remove_all_filters,
doing_filter, has_action, add_action, do_action,
remove_action, remove_all_actions, doing_action,
register_uninstall_hook, current_filter,
register_deactivation_hook, did_action,
register_activation_hook
```

Figure 5: List of WordPress Plugin APIs.

WordPress plugins are installed in dedicated plugin directories on the webserver. Figure 5 shows the plugin-specific API calls commonly used in WordPress plugins and themes. These are called *Hooks* that enable one piece of code to interact/modify another piece of code at specific, pre-defined spots, thus helping the plugin interact with the WordPress core.

It is up to the website owner to set the plugin's user access permissions [84]. Based on these permissions, the website owner controls the directories that a plugin can access for read, write, and execute. Plugins can either be manually updated or set to auto-update upon which WordPress downloads and installs any updates from the plugin store. However, WordPress cannot automatically update plugins from 3rd-party marketplaces. These need to be manually updated. Since most plugins have both read and write permissions in the plugin installation directory, a malicious plugin can scan for other plugins to inject malicious code, thus infecting it. Malicious plugins installed outside the plugin installation directory can use a CMS account or webshell to first change the access permissions (such as in SSO Backdoor attacks) and then infect other plugins.

C Syntactic Analysis

Classes of Suspicious Sinks	Suspicious API description
Exec	Execute code
File_RW	File read/write
Decode	Decode functions
Download	Download from URLs
Create_Func	Define function from string inputs
SE_Bots	References to search engine botnames
Chng_File_Perm	Changes the file permissions
Chng_User_Perm	Changes the user permissions
Create_User	Creates a default user account
Register_User	Registers a user account to the CMS
Add_Post	Adds a new post
Inseert_Popup	Adds code to display a popup
Add_Content	Appends content to HTML metadata
Replace_Content	Replaces old content with new content
Func_Exists	Check if a function exits
Redirect	Redirects to the URL passed as input
Delete_Payload	Deletes the downloaded payload
Redirect_NewUser_Admin_URL	Redirects to the new user's admin URL

Table 10: Classes of Suspicious API Sinks.

YODA generates the AST for all of the plugin code files and parses it to record the sensitive API classes summarized in Table 10. These APIs will later form the sinks for semantic dataflow analysis. Table 10 shows the notations used for the classes of suspicious sinks and the API class description. For example, the *Decode* class denotes decode functions such as `base64_decode, json_decode`. The list of these sinks was identified by studying past research as well as industry reports of malicious plugins being removed from popular marketplaces between 2013-2018 [14], [30]–[32]. YODA may miss identifying sinks that are based on novel implementations of the attack behaviors

such as updates to the PHP language. Since this is a rare event, YODA's models can be updated easily when necessary.

Plugins can have inter-file dependencies that invoke suspicious APIs indirectly. An intuitive solution for handling inter-file dependencies is to analyze each plugin code file together with its dependencies, but this may lead to the repeated analysis of common dependencies and possible resource exhaustion given too many dependencies. Therefore, to increase efficiency and reduce failures YODA eliminates the inter-file dependencies by recursively replacing the dependencies with their respective ASTs via modularized API usage analysis which analyzes each dependency only once. This also handles the case of cyclic dependencies if any.

The dynamic nature of the PHP language (e.g., dynamically evaluating code) can introduce challenges to accurately detecting the suspicious sinks. If the malicious plugin generates new code that was not available during static AST generation, then YODA cannot access the sinks in the new code. However, we found that the tactic of function splitting to evade pattern-based detectors was more prevalent than new code generation. YODA handles function splitting by concatenate the individual function pieces defined in the AST to reconstruct the intended function. Lastly, if the plugin uses an external input for function creation (e.g., new code fetched from a URL), YODA cannot reconstruct the entire function at the AST-level.

YODA then analyzes all of the plugin code files to collect syntactic measurements, specifically: (1) the number of files and filetypes in a plugin, (2) the effective plugin state (described in §3.1), (3) the longest code-line length (termed *Long_Line*) and (4) the presence of UTF-8 encoded characters or obfuscation patterns [9], [48] such as a combination of `'O'`s and `'O'`s (termed *Jumbled_Obfus*). These measurements will be used to detect the code obfuscation behaviors in semantic analysis (Table 2).

D SEO By Nulled Plugins

Nulled marketplaces existed since 2013, but they gained popularity around 2018, and their blackhat SEO campaigns accelerated only in early 2019. A Google search for any "_____ WordPress plugin/theme free download" almost always has a nulled marketplace in the top five results. Figure 6 shows the search engine results for one of the popular WordPress themes, DooPlay, normally priced at $80 [85]. Here, the highlighted four of the top five results on Google search lead us to nulled marketplaces that are known for distributing malicious plugins and themes. In Nov 2019, WordFence alerted the community about a rouge blackhat SEO malware campaign via nulled malicious plugins and themes [15]. Despite this knowledge, the attackers have successfully maintained their ranks on the search engine results.

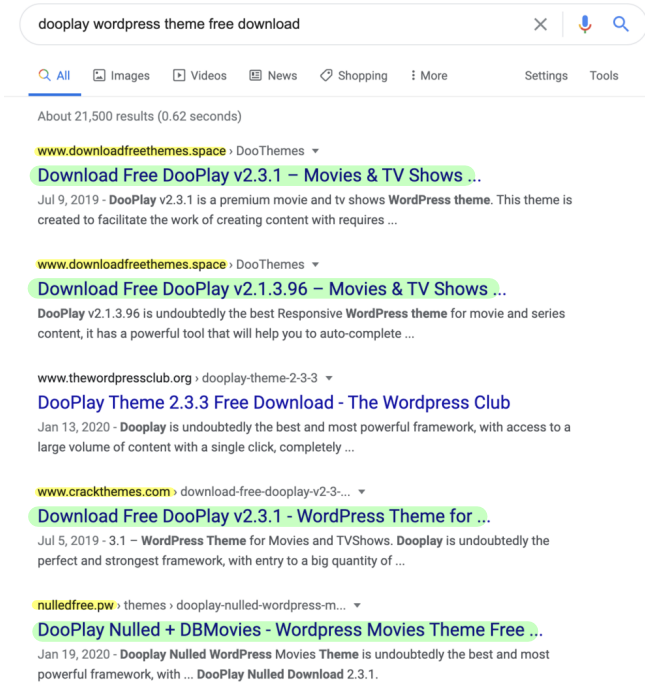

Figure 6: Google Search Results of a Typical Paid Plugin.