Messaging, Malware and Mobile Anti-Abuse Working Group (M³AAWG)

M³AAWG Best Common Practices for Mitigating Abuse of Web Messaging Systems, Version 1.1

Updated March 2019 (2010)

The reference URL for this document is www.m3aawg.org/WebMessagingAbuse

Table of Contents

Updated in this Version

A new section has been added to the March 2019 version 1.1 that includes some suggest ions for managing the collection, storage and indexing of data that may shed light on potential abuse. Another added section discusses multifactor authentication and its role in protecting Web systems from abuse. This version also clarifies the appropriate methodology for tracking and grouping IPv6 addresses and updates the "Content Filtering" section to reference more sophisticated sanitization methods.

Introduction

As spam filters have improved at blocking direct connections from spammers, cybercriminals are increasingly turning to Web-based messaging systems to transmit their content. By volume, the preponderance of this spam still occurs directly on traditional "Web Mail" sites, where criminals write automated scripts that mimic user actions and send outbound spam. However, these same abusive techniques have been used outside the email domain with spammers entering content on blogs and discussion boards; abusing instant messaging and SMS portals; and misusing notification, share-with-a-friend, instant messaging and "status" features.

Many techniques are employed to prevent or mitigate these attacks and this document details the Best Common Practices for protecting Web-based messaging systems. Its intended audience is technical professionals responsible for Web messaging feature development working in engineering, information technology (IT), network operations or a more general abuse prevention role.

This paper is the result of cooperative efforts within the Messaging, Malware and Mobile Anti-Abuse Working Group. M[3]AAWG is the largest global industry association working against spam, viruses, denial-of-service attacks and other online exploitation. Its members include ISPs, network and mobile operators, key technology providers and volume sender organizations. Its members represent over one billion mailboxes worldwide and have contributed their expertise in developing this description of Web messaging abuse prevention practices.

Typical Attacks

The unifying concept of Web messaging abuse is that the abusers interact with a Web user interface (i.e., HTTP(S)-based), through which they submit content that is either displayed online or distributed to target users.

In addition to direct spam email from a Web messaging product, several other examples of this type of abuse are:

- "Spammy" content added to event invitations or e-cards
- "Ridealong spam," where spammy content is added as a preamble to a "share this article" or "invitation to connect" system-generated message
- Spammy comments added to a blog, discussion board or media sharing site
- Spammy "status" notices on an instant messenger or a social network profile
- SMS spam sent through a Web->SMS gateway or providers of REST APIs that provide SMS sending capabilities

As more and more notification channels are added online – including RSS readers, newsfeeds, status updates, Twitter clients and instant messaging – the temptation for spammers continues to increase.

Monitoring and Alerting

The first stage in protecting a site lies in understanding the degree and frequency of abuse. Lacking adequate monitoring, many sites are caught unaware, only realizing an attack has been brewing when it reaches a critical level.

Unfortunately, there is no generic approach to monitoring. Instead, administrators must conduct an audit of the resources attackers may wish to abuse and implement a tracking solution for each. To facilitate this audit, several example areas that most properties should monitor on an ongoing basis are:

- User/Account Metrics
  - New user registrations
  - User logins
  - User password changes
  - User account deactivations
- Abusive Transactions
  - Visits to non-existent pages
  - Suspicious query strings
  - Visits to uncommon ports
- Potentially Suspicious Transactions
  - Total actions completed (e.g., messages sent, comments posted)
  - Identities who reached any of the limits or heuristics suggested above

The above is just an example of the types of data that can be collected. Log-scraping is likely to be the most common collection method. However, if hooks are available directly in the Web application, then those can be very useful. You may also need to work with your developers or vendors to ensure the relevant information is logged and/or the relevant hooks are available.

The importance of monitoring and collecting data about potential abuse cannot be underestimated. Firstly, it is generally easier to get a budget to address the problem if hard statistics are collected. Secondly, it helps to understand exactly what the abuse problems are. For example, it might be assumed that most abuse is the result of compromised accounts, whereas careful analysis of the relevant metric may reveal that, in fact, it is the trial account signup process that is being abused.

One of the main considerations is how to collate, store and index such data. Some suggestions follow:

- Consider using some form of database to store data for a period of time. Much better analysis can be performed if data is stored for a week or longer, as this can help to find attackers who might otherwise go undetected (low-n-slow style attacks).
- Ideally store information about the device used to login by storing the User-Agent header string (or some parsed extracts of it). This can be extremely useful in determining potential abuse, as attackers are unlikely to be providing exactly the same User-Agent string as legitimate users. Of course, this can also be used to identify User-Agent information that is only associated with abusers.
- Consider collating data between your Web Mail system, IMAP/POP servers, and MTAs. This can lead to even more valuable insights by detecting abuse across all those systems, as well as sharing information about detected abusers between all three.

Proactive Defense

In conjunction with the operational monitoring techniques detailed above, the bulk of service providers' time and efforts must be spent on proactive defense with systems put in place to limit the number and types of attacks. There are three key areas to consider in defending against Web messaging abuse:

1. A potential abusers' access to the UI (user interface)
2. The content that a user is allowed to submit
3. The distribution breadth and format of the content

Each of these stages represents an area that must be both monitored and protected against abuse since a vulnerability in any of these can be exploited by spammers. The sections below address each area.

UI Access

It almost goes without saying, but if abusers are prevented from accessing the website by various technologies like those discussed below, their ability to send spam ends there. Yet many providers focus entirely on content filtering instead, leaving vulnerabilities in this area.

Authentication

The most common technique used to limit access to a particular Web feature is authentication. While far from a panacea, requiring a user to create an account before completing an action increases the cost to abusers, thus reducing the amount of abuse. The impact of authentication can be considered in simple, economic terms: Each "step" the abusers need to take to accomplish their goals has a "cost," so requiring them to create an account and log in increases the setup cost, which in turn decreases the "return on investment" of the attacks. Obviously, the cost is not infinite, so if the value of the attack remains greater than the cost, abuse will continue.

Registration/provisioning and account compromise are at the heart of authentication: The difficulty of creating or obtaining an account is a major factor in the "cost" to the abuser. Nearly all of the messaging-specific techniques in this document can and should be used to secure account registration and provisioning as well. While account compromise and password cracking are outside the scope of this document, one possible solution for some types of service is to rely on an outside provider. Rather than building a registration and authentication system, companies can accept credentials via API from Microsoft, Verizon Media Group (AOL and Yahoo), Facebook, Twitter or other services and avoid the pains and perils of building a secure implementation from scratch. Alternatively, many companies will choose to use identity management systems that are in place for other services such as Web portals, as such authentication is typically handled by those systems. Modern protocols such as OpenID Connect are strongly encouraged when integrating with third-party or in-house identity management providers as they provide a standard and interoperable way to provide user authentication and authorization.

Multifactor Authentication

A property of Web applications is that they lend themselves well to multifactor authentication methods since the user interface is fully controllable. (Note that if an external identity management system is used for authentication as described in the previous section, then support for second-factor authentication would typically be provided as part of that service).

Multifactor authentication attempts to confirm a user's identity by asking them to present two or more pieces of evidence (or factors) based on knowledge (something they know like a password); possession (something they have, for example, a software or hardware security key); or inherence (something they are, for example, biometric information). Multifactor authentication is not a panacea but it is extremely useful in combating abusers using compromised accounts since the username and password are not enough to authenticate.

The first factor for Web applications is typically username/password. A second factor is typically then based on possession and includes the following methods (this document does not consider the relative merits of the different methods):

- SMS – Sending a one-time code to a mobile phone number via the SMS messaging service. This relies on the user having possession of their mobile phone to retrieve the code.
- One-Time Password (TOTP/HOTP etc.) – This requires the user to use an application (such as Google Authenticator, Duo Mobile, etc.) or a hardware device to generate a one-time password from a shared secret key.
- U2F (Universal Second Factor) – A standard for using hardware devices (security tokens/keys) that use public-key cryptography for the second factor.

If implementors choose to use a multifactor solution, care should be taken in how it is rolled out to users. Simply forcing all users to enroll for a second factor is likely to cause a great deal of disruption and support calls. One technique to reduce this is to concentrate initially on those users who have fallen victim to account compromise in the past or who are considered more vulnerable. This can be implemented at the same time as making multifactor optional for all users and allowing self-enrollment.

One way to reduce the potential disruption of multifactor authentication is to perform "risk-based" enforcement. For example, only request the second factor when something suspicious about a login is detected or always request the second factor when a certain action is requested, such as changing the password or sending an email to a large number of recipients (see "Challenge" section below). Additionally, implementors may give users the ability to "remember" specific devices, thus they would not need to provide second factor information when using that device in future.

CAPTCHA

A close second in common UI-level prevention techniques is the use of CAPTCHA – most often a visual puzzle difficult for computers to recognize, but also occasionally an audio puzzle, mathematical puzzle or trivia question – as a requirement before completing an action. Like authentication, CAPTCHA is not a silver bullet but undeniably reduces the incidence of abuse. That said, the "cost" to abusers of solving CAPTCHA has been declining rapidly due to advances in Optical Character Recognition (OCR) and the use of low-cost "sweatshops" employing human CAPTCHA-solvers, so it should always be used in conjunction with other techniques. As with authentication, it is recommended that implementers stay away from developing "homemade" CAPTCHA solutions and should instead rely on a field-tested and actively maintained solution such as Google's reCAPTCHA.

Web Page "Hardening"

A third method of limiting abusers' access to a Web feature is by "hardening" the interface against automated scripting. Some techniques include randomizing or varying the structure of the page (e.g., changing the HTML "ID" parameter of required input fields), obfuscating the structure of the page (e.g., using JavaScript to encrypt field names or AJAX to dynamically present fields), and laying traps (e.g., using CSS to create invisible fields with the expectation that humans will skip over them but robots will reveal themselves by entering data).

Unfortunately, many of these techniques have the unintended consequence of decreasing accessibility for blind users or others who do not rely solely on visual cues when completing a webpage. Additionally, purists will correctly claim that obfuscation provides no true security and that these techniques will, at best, present a temporary hassle for determined attackers. While true, every additional hassle increases the cost to abusers. Locking the door does not guarantee you will not be robbed but leaving the door ajar does make you an inviting target for casual or opportunistic thieves.

Rate Limiting and Dynamic Blocking

The final topic in the UI Access category involves dynamically limiting whether and how often certain users, computers or browsers can access your Web service. "Rate limiting" – in which a system regulates the allowable quantity and frequency of a particular action – should be employed on almost all Web services that accept or relay user-generated content (UGC).

The determination of an attacker's identity is always a major challenge in internet security. While clearly the broad topic of identity is outside the scope of this document, for rate limiting purposes it is necessary for a service provider to choose a stable "handle" upon which rate limiting can be performed. The most common attributes are a combination involving one or more of:

- Remote IP address
- The browser User-Agent string
- Username (where applicable)
- Session ID or cookie value (where applicable)
- Computed machine identifier (e.g., a MAC address, CPU serial number or OS-provided identifier)

Several providers have successfully employed a solution based on the open-source "mod security" Web server plugin while others have accomplished this task with a simple counting server. With either tool, at its most basic, administrators maintain a count for each action clustered along one or more of the identity dimensions listed above. For example, counters might be configured to limit the number of messages to "50 per hour and 100 per day for each user ID," after which the Web server will redirect user requests to an error page. Care must be taken that all access points for a feature are protected (for example, all of the servers behind a load-balancer), and that the protection cannot be circumvented by a misbehaving Web browser (e.g., if the attacker ignores an HTTP redirect code).

One major difficulty with rate limiting is the definition of the rules. Care must be taken to handle accidental spikes in user behavior (e.g., the case where a good user attempts to send "I just had a baby" to his whole list) and misinterpretation of clustering dimensions (e.g., the IP address of an internet cafe or public library may represent 20 users, or a business traveler may log in from a large number of unusual locales).

Attention must also be paid to the user interface presented when the limit is reached. "False positives" can be completely mystifying to users. For example, in an effort to not divulge the limits to abusers, instead of a straightforward message such as "You are not allowed to send more than 100 messages per hour," the user is given a vague message such as "Suspicious activity detected..." But these vague error pages can generate costly customer support volume. Oftentimes, the spammers either already know the limits or will continue sending even beyond the limits. However, legitimate users may be completely unaware they have reached the limit because they are sharing a NAT IP address with their roommate's laptop, which has a bot and is hemorrhaging spam.

Dynamic blocking can be considered a degenerate case of rate limiting, where a forbidden IP address is allowed exactly zero transactions per day. Some providers have used third-party data feeds to determine the blocklist – for example the Spamhaus Auth BL may be applicable – but care must be taken when adapting a blacklist to an unintended purpose. (For example, the Spamhaus PBL lists IP addresses that should not connect to port 25 but it is completely inappropriate for blocking port 80.)

A special note on IPv6: In almost all Web messaging abuse scenarios, the remote user's IP address is a valuable identifier for rate limiting since it is fairly robust against spoofing or tampering. As the migration from IPv4 to IPv6 radically expands the remote IP space, care must be taken in the design of rate limiting systems to accommodate this much larger signal space. Implementers should consult the paper M³AAWG Policy Issues for Receiving Email in a World with IPv6 Hosts[1]. Concerns to keep in mind around rate limiting are:

- IPv6 Addresses should not be tracked as individual (i.e. /128) addresses; instead ranges should be tracked by default. Since the IPv6 address space is so large, ISPs typically assign each customer a block of address space that never changes. For example, the smallest allocation is typically /64, although the size of the range assigned to each customer varies by ISP from a /64 to as much as /48.
- Lack of remote IP scarcity may require lowering thresholds

Carrier-grade NAT routers – which hide dozens or hundreds of consumers behind a single IPv4 address – may also necessitate raising thresholds.

As the adoption of IPv6 continues to increase, implementors are advised to keep these concerns in mind and to continue to consult up-to-date information from M³AAWG about evolving best practices for IPv6.

Web Application Security

In general, the security of the Web application should adhere to the principles laid out in the OWASP Top 10 (https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project).

Content Filtering

Least Privilege

The strongest suggestion for content filtering follows the security principle of "least privilege:" Only allow the types of content that you absolutely must, ideally using a “whitelist” approach rather than a “blacklist,” as the

[1] M³AAWG Policy Issues for Receiving Email in a World with IPv6 Hosts, September 2014, http://www.m3aawg.org/sites/maawg/files/news/M3AAWG_Inbound_IPv6_Policy_Issues-2014-09.pdf

latter approach can quickly lead to a "whack-a-mole" problem as attackers discover new ways to get around the blacklist.

However, sanitization of user-generated content is still extremely valuable and necessary. Ideally the sanitization engine will be more sophisticated than simply stripping out script tags, as there are a myriad of vulnerabilities that exploit such primitive techniques, usually by encoding content in a way that the sanitizer does not recognize the script tag but a browser will. For example, a sanitizer based on the rendering code of a widely used browser will be far more effective in detecting potential threats.

Attachments or file uploads, where permitted, should be restricted by file type or extension and ideally scanned for malware.

Another aspect of least privilege is restricting the character count of the message payload. In the note accompanying a shared news article, there is little need to accept 1000 characters of input. Restricting the message to 200 characters supports the intended need while reducing the value of this field to spammers.

Heuristics and Spam Filtering

Heuristic filtering – for example, comparing submissions to a blacklist of forbidden URLs, text strings or other tokens – can further encourage spammers to look elsewhere. Unfortunately, heuristics must often be manually created and maintained; yet they can often be easily evaded – for example, by writing "V.1agra" or using a URL redirector – making them a costly and complicated option for site administrators. These heuristic approaches must almost always be accompanied by a scalable process for populating and expiring rules.

Alternatively, open-source or proprietary spam filtering engines can be integrated with the submission process to scan messages for known spam patterns before accepting the message for delivery.

Reputation Models

More flexible than heuristics filtering are online and offline reputation models which dynamically evolve based on patterns in outbound content and sender profiles. For example, a model could be constructed to examine user trust that considers the account creation date, the number of positive actions performed, and the number of suspicious actions performed. Upon a questionable submission, the system could consult this model to determine the prior probability that the message is spammy.

Batch Analysis

The most flexible approach to message filtering includes examining a very broad feature vector for patterns that indicate suspicion or trustworthiness. To do so, clustering algorithms are run on a batch of recent messages to find unexpected patterns – perhaps accounts that registered on June 7 from an IP in 1.2.0.0/16 are all sending messages with 103 words. Those patterns are then loaded back onto the outbound filters, and in this case, subsequent messages matching this pattern might be flagged for further scrutiny.

Distribution

The final aspect to consider in mitigating Web-based abuse is the degree of distribution of the suspected message. The most compelling aspect for would-be spammers is how many potential customers will receive the message. Therefore, techniques that limit the distribution can be effective at combating the abuse itself.

Recipient Limiting

In an email or direct messaging system, administrators may choose to limit the number of recipients an untrusted user may include in a message without arousing suspicion. In systems with implicit distribution, such as a blog or a comment system, this may take the mode of required content moderation or a quarantine of new posts.

Responses to Abuse

Following the Content Filtering stage, a good message is "sent" (i.e., posted, stored, tweeted, etc.) while a bad message should go through one of three paths:

- Challenge: A UI can be presented to the user for further validation
- Quarantine: The message can be delayed or forwarded for moderation
- Reject: The message is not delivered; it can be silently deleted or overtly rejected and sent back to the submitting user

Challenge

When content filtering determines the message falls into a grey area between "definitely good" and "flagrantly bad," administrators may wish to present a challenge to the submitting user for further validation. Oftentimes this takes the form of a CAPTCHA – e.g., "Before accepting this post, please solve this puzzle"– but increasingly, sites are using two-factor authentication methods as discussed above.

Just as with the content submission itself, care should be taken that the challenge UI is similarly layered with security protections. Over-reliance on CAPTCHA can create a vector for attackers to send flagrant messages for a reasonably low cost. A significant vulnerability exists if quasi-bad messages can be distributed for just the price of an automated CAPTCHA solution.

Quarantine

In certain circumstances, automated filters may need more time to examine a message – for example, to see whether an unusually large volume is being submitted. In some cases, manual moderation may be required. This is most common with photo sharing sites.

Reject

The only decision around rejecting content is whether or not to alert the user. While doing so does convey information that spammers could use to adjust their tactics, silently deleting a message can generate customer support volume as well as tremendous difficulty troubleshooting any system bugs. Most providers have adopted a compromise position, displaying an error message back to the user but keeping its content fairly vague or generic.

Conclusion

With spammers and botnet herders increasingly moving toward scripting Web interfaces, the three sections above—Monitoring and Alerting, Proactive Defense, and Responses to Abuse – lay out a series of techniques that site administrators can use to protect against or mitigate the effects of abuse. Beginning with understanding the magnitude of the problems and leading through techniques for addressing them, this document lays out a comprehensive approach for site administrators to combat cyber abuse of their Web messaging systems.

As with all documents that we publish, please check the M³AAWG website (www.m3aawg.org) for updates.

M3AAWG032-2019