Outsourcing Malicious Infrastructure to the Cloud

Georgios Kontaxis, Iasonas Polakis, Sotiris Ioannidis
Institute of Computer Science
Foundation for Research and Technology Hellas
{*kondax, polakis, sotiris*}*@ics.forth.gr*

***Abstract*—Malicious activities, such as running botnets, phishing sites or keyloggers, require an underlying infrastructure for carrying out vital operations like hosting coordination mechanisms or storing stolen information. In the past, attackers have used their own resources or compromised machines.**

In this paper, we discuss the emerging practice of attackers outsourcing their malicious infrastructure to the Cloud. We present our findings from the study of the first major keylogger that has employed Pastebin for storing stolen information. Furthermore, we outline the traits and features of Cloud services in facilitating malicious activities. Finally, we discuss how the nature of the Cloud may shape future security monitoring and enhance defenses against such practices.

I. Introduction

Malicious activities, such as running botnets, phishing sites or keyloggers, require an underlying infrastructure for carrying out vital operations like hosting coordination mechanisms or storing collected information. In the past, attackers have used their own resources or compromised machines to store stolen user information (from keyloggers or phishing schemes) or coordinate their activities (issue new orders, push updates, etc) as in the case of botnets. Both cases entail disadvantages for the attackers. In the first case, their actions can be traced back to them. In the second case, the infrastructure is not reliable as compromised servers can be identified and patched.

In this paper, we highlight the emerging practice of attackers outsourcing their malicious infrastructure to the Cloud. In other words, we discuss a new trend on the Internet where attackers switch from IRC channels to Twitter accounts for coordinating their botnets and from private FTP sites to public user-content hosting sites such as Pastebin. We present the findings from our study of the first major keylogger that has employed Pastebin to upload stolen information. Furthermore, we analyze the nature of Cloud services in respect to the needs of attackers. Finally, we discuss how such shift towards Cloud-based infrastructure may shape future security monitoring and defense mechanisms.

The contributions of this paper are the following:

- We present a study on the first major case of a keylogger using Pastebin to upload its stolen information. To the best of our knowledge, this is the first study concerning the use of public content-sharing sites as malicious dropzones.
- We discuss the nature of Pastebin-like sites and present scenarios where their features can be employed by malware to coordinate their actions and store information. We also provide proof that attackers are already discussing and developing techniques based on the, sometimes unique, features of such Cloud services.
- We discuss the future of Pastebin-enabled keyloggers, along with new ways that attackers can take advantage of the Pastebin service to enhance other nefarious activities, such as botnet coordination.
- We propose Cloud-oriented security methods for detecting and monitoring this new generation of malware.

II. Related

Security researchers at Kasperky Labs [1] report that there is an increasing trend for botnets to move their command and control channels away from IRC and into the Web. They claim that the number of HTTP-based coordination channels outnumbers the IRC servers by a factor of 10 to 1, as a result of monitoring and aggressive action against the latter.

Authors in [2] acknowledge and study the trend of botnet command and control services evolving from traditional IRC-based approaches to the Web and, specifically, to social-networking sites such as Twitter. Furthermore, they point out the irony of attackers hiding their infrastructure in plain sight in popular Web sites so as to improve the unobservability of their operations.

Symantec in [3] and also security researchers in [4] discuss the nature of a new type of malware that employs Twitter as part of its command and control infrastructure. In detail, the malware accesses the public timeline of messages (tweets) of a specific Twitter account that contains base64-encoded strings. These strings contain URLs to pastebin-like sites that in turn carry DLL and executable files also in base64-encoded form.

A technical report by Balatzar et al. [5] exposes the Web 2.0 orientation of a new generation of malware with social network propagation components and an infrastructure, from command and control services to malware repositories and storage of stolen information, entirely over HTTP applications.

In 2009, 10K Hotmail passwords were leaked [6] to the Web via public uploads to Pastebin.com. In 2010, 10K credit card numbers were also leaked [7] on the same service.

In 2010, the use of Pastebin and similar sites has been suggested [8] as a way for criminals to anonymously store, exchange or even advertise samples of stolen information, with little or no risk of liability.

Holz et al. in [9] perform dynamic analysis of malicious software (e.g. keyloggers) in an automated fashion. Their goal is to discover online repositories that each malware uses to upload stolen information. Furthermore, they analyze the nature and content of the stolen data, thus providing an insight into the back-end of such underground operations. Their investigation of popular keyloggers has led them to privately-owned or compromised Internet servers that execute attacker-provided scripts (e.g. PHP) to implement the necessary functionality for storing and managing the stolen information.

Our work focuses on the emerging trend of keyloggers and other malicious software employing inherent functionality of public Cloud services, such as user-content-hosting.

III. Background

In this section, we outline the generic behavior of key-logging malware (keyloggers) and also introduce the basic features of the Pastebin service.

A. Keyloggers

A keylogger is a piece of computer code, usually packaged as a stealth program, that records all keys struck on a keyboard. It falls in the category of malicious software as it is often employed for stealing sensitive user information, such as passwords or financial information, as the owner types them (e.g., while logging in a service or making an online purchase). Keyloggers may simply record all key strokes or be more sophisticated and capture keyboard activity right after a predefined sequence of keystrokes (e.g., after the user types mybank.com, presumably in the Web browser's address bar). Subsequently, keyloggers place the captured information in a log file and upload it to an online location (or dropzone) accessible by the attacker. Sophisticated keyloggers present data in a structured form, perhaps identifying URLs ("http" or "www" strings) and grouping keystrokes accordingly. Simpler tools provide access to a chaotic stream of keystrokes for the attacker to extract useful information from.

B. Pastebin

Pastebin sites were originally conceived as clipboard-like collaboration places where developers could conveniently share source code, logs and other text-based content without having to worry too much about the original formatting getting corrupted or other problems associated with trying to share structured text over e-mail or instant messaging applications. A registered account is not required. One may use the service to upload (or paste) arbitrary blocks of text online and receive a URL (pointer) to that content. He may then share that URL with their colleagues or friends or keep it private for personal use. Returned URLs are of the form http://pastebin.com/⟨ID⟩, where the ID appears to be random, perhaps the product of a hash function. The lifetime of such content is user-defined and may vary between 10 minutes to 1 hour, 1 day, 1 month or for ever. Due to the service's orientation towards developer, it offers syntax highlighting. By default, pasted content offers no syntax highlighting, never expires, is public and carries no identifying title or username (anonymous). All public pastes appear in a "recent posts" timeline and can therefore be accessed by anyone. Crawling the keyspace of Pastebin IDs is not an option as the space is quite large (7-8 characters long, [A-Za-z0-9]) and randomly populated. Finally, an interesting feature of Pastebin is the support for arbitrary subdomains. Any guest of the service may type http://⟨anything⟩.pastebin.com and will be presented with a valid view of the service. Any pastes created under that arbitrary subdomain are not referenced by http://pastebin.com/⟨ID⟩ but by http://⟨anything⟩.pastebin.com/⟨ID⟩ and for that matter do not appear in the public timeline. To access such content, one requires both the random ID and the domain prefix.

IV. The Pastebin Incident

In this section we present, to the best of our knowledge, the first major case of a keylogger using Pastebin (or any other Cloud service for that matter) to upload its stolen information. We introduce the timeline of events, provide technical details regarding our efforts to capture and study the incident and, finally, share the outcome of our analysis regarding the nature of the malware.

In May 2010 a large number of entries, containing raw streams of what appeared to be keystrokes, began appearing on Pastebin. The presence of lists of usernames and passwords is not new to the service, but this specific type of entries quickly became a trend while constantly increasing in volume and ended up dominating the content being uploaded to the service during that period of time.

Pastes that fell in this category carried the same set of characteristics: anonymous entries, with no syntax highlighting or specific internal structure, comprised of "[ ]" blocks carrying what appeared to be titles of Web browser windows (e.g., "Internet Explorer - Facebook.com" or "Mozilla Firefox - Hotmail.com"), followed by a stream of keystrokes. Their frequency was so high that they dominated, almost completely, the "recent posts" list in the service's homepage.

The overwhelming volume of such pastes caught our attention and, at the same time, emerged as an issue in security-related blogs [10], [11]. Their sudden appearance and sudden increase in volume, indicated the launching of a new keylogging tool that employed Pastebin to upload the stolen information. We started to monitor this incident and analyze its characteristics as it appeared to be the first

of its kind. The keylogging tool was later identified by BitDefender [1] as *Trojan.Keylogger.PBin.A*.

A. Dataset Collection

In this section we outline our collection methodology of keylogger data (or pastes) on Pastebin. Moreover, we elaborate on the completeness of our collected trace and discuss certain limitations of our approach.

Active Crawling. A straightforward crawl of all uploads on Pastebin was not an option, so we resolved to a more elegant solution. Pastebin uploads are assigned a random-looking ID of the form http://pastebin.com/[0-9A-Za-z]+. Since the ID distribution appears to be random, a simple iteration over the available namespace would be inefficient in terms of time. However, by default the last 8 pastes created appear in a "recent posts" timeline on the homepage of Pastebin. We periodically downloaded the list, parsed its entries and fetched any pastes that we could attribute to the output characteristics of the keylogger. By examining, fast enough, the "recent posts" timeline, we were able to gather all or almost all keylogger pastes as soon as they were created.

We developed an infrastructure that periodically downloaded the "recent posts" timeline, examined it for entries matching our search criteria and downloaded any pastes that were determined to be a match. In detail, we downloaded the homepage of Pastebin and parsed the entries present in the recent timeline. Entries in that list contain the username or title of the paste, along with a label describing its content type (later used for syntax highlighting, e.g. C, HTML, PHP). Furthermore, all entries are active hyperlinks leading to the page of each paste. The pastes containing keylogger output were created using default parameters, i.e., by an anonymous user and without a type specification. Our infrastructure followed the hyperlinks for pastes matching the default parameters, downloaded the respective pages and performed regular expression string matching to identify structural properties of the keylogger pastes. As mentioned earlier, each keylogger paste had its streams of keys grouped around lines of the format "[<browser> - <window title>]" (e.g., "Internet Explorer - Facebook.com"). If the downloaded pastes exhibited this kind of structure, they were considered valid matches and kept for further analysis. Otherwise, they were discarded.

To secure the completeness of our collection methodology we took steps to ensure that our polling rate (periodic download of the timeline) was fast enough to capture all interesting pastes before they disappeared from the timeline (during period of increased activity) and also not to frequent so as to avoid stressing the service when not necessary. In that light, each time we downloaded the timeline, we calculated the dice coefficient of its entries with the one downloaded previously. A dice coefficient above 0.75 meant that 75% of the entries in the timeline had not changed since the previous time we had downloaded the list; we considered this as an indication that we were polling the service too fast in a period where few uploads were made per second. In that case, we slowed down our polling rate. On the other hand, a dice coefficient of less than 0.25 meant that only 25% of the entries in the timeline remained the same, which was fine, meaning we had not missed any entries between our last poll and the current one. But if a sudden burst of activity occurred we could miss some entries. To avoid such a case, we increased our polling rate. Overall, the average value of the coefficient was 0.80, indicating a sufficient polling rate, able to provide a complete keylogger data trace.

Time-machine Crawling. Our previous crawling method allowed an efficient gathering of all subsequent pastes. However, we also wanted to check if similar pastes had been uploaded in the past, how long ago and plot their volume as a function of time. For that matter, we employed the advanced tools provided by the Google search engine. In detail, Google Search allows one to search for a keyword within a specific domain and also limit the query scope using time constraints. To perform our backwards search we placed a query similar to "site:pastebin.com <search heuristic>" and limited the scope to one month at a time.

B. Limitations

As mentioned earlier, by default all pastes are created as public and appear on the "recent posts" timeline. By examining that timeline with a fast-enough rate, one may compile a complete or near-complete set of the public pastes created. Here we describe two cases where such an approach may not be that effective.

A paste may be explicitly set to being private and therefore will never appear in the public timeline. Considering that all pastes, public or private, are assigned random-looking IDs, there is no way for a third party to discover that paste other than guessing character sequences from a very wide namespace.

Furthermore, the use of arbitrary subdomains also allows the creation of pastes that do not appear on the service's homepage. Posts on those subdomains may still be public but one would have to know the specific prefix before pastebin.com to access their respective timeline.

We believe that, in this case study, such limitations did not apply. However, as we discuss in Section V, future strings of keyloggers could employ such techniques to hide their presence from the public.

C. Analysis

Data Volume. Using our active (or forward) crawling technique, we gathered an almost-complete set of all new pastes that matched our heuristics regarding the content and structure of the keylogger pastes. Figure 1 plots the volume

[1] http://www.bitdefender.com/

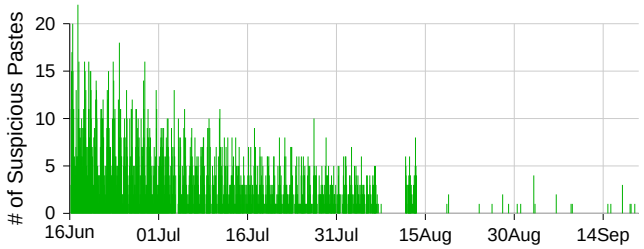

Figure 1: Volume of PBin.A stolen information on Pastebin, per hour. (Active Crawling)

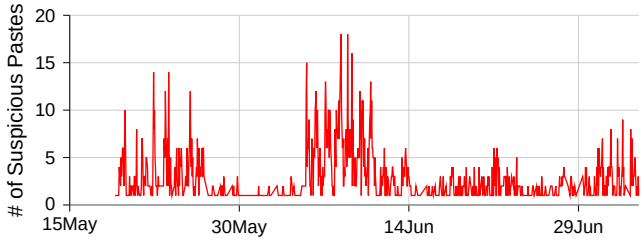

Figure 2: Indexed Volume of PBin.A stolen information on Pastebin, per hour. (Google Search)

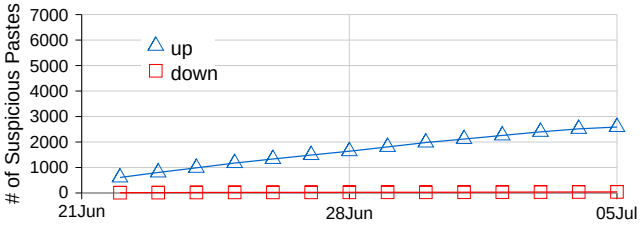

Figure 3: Cumulative Volume of available and taken-down PBin.A entries of stolen information on Pastebin, during the middle of the activity period.

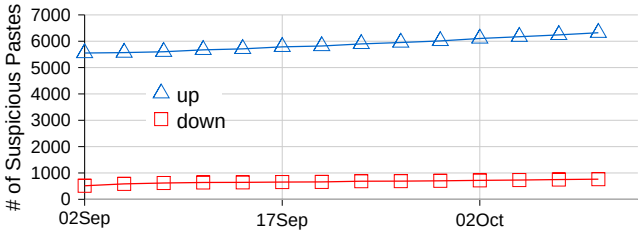

Figure 4: Cumulative plot of available and taken-down PBin.A entries of stolen information Pastebin, after the malware had almost ceased its operation.

of new pastes over a period of 4 months. A closer look reveals diurnal patterns, an expected phenomenon attributed to daylight cycles and user behavior which has been extensively documented in [12]. Moreover, there is a steady decrease in volume, indicating that the shrinking presence of this keylogger in infected computers.

Furthermore, we employed our backwards crawling technique and, using Google Search, calculated the volume per day up to 6 months before the time we started the active crawling. While we received search results for the entire search period, pastes matching our heuristics began appearing in May 2010, indicating that the appearance of keylogger data on Pastebin was a new trend and we had caught it almost as soon as it started. Figure 2 plots the volume of pastes over time. There are no pastes before May 19. Let there be noted that this figure presents the Google-search indexed volume of Pastebin entries which is essentially a sampled dataset. On the other hand, Figure 1 visualizes the almost complete set of pastes during our period of active crawling.

Takedown Rate. During the first few days since keylogger data started appearing in large volume on Pastebin, the service claimed [11] that efforts were being made to identify those entries and remove them. To verify that claim, we periodically checked the availability of pastes collected by our crawler. Figure 3 plots the cumulative volume of available pastes over the period of our active crawling, using a one day granularity. Since pastes were created using default options and were set to never expire, we can attribute any unavailability to takedown efforts. However, one may see that efforts were not sufficient in identifying existing data or keep up with the rate of new data being uploaded. Moreover, in September 2010, almost a month after the data uploading activities of the keylogger had nearly ceased, the stolen information still remained online and, apparently, takedown efforts had completely ceased. (Figure 4).

D. Strings Identified

Although our analysis heuristics were strict, ensuring that the set of pastes were indeed part of the keylogger output, our crawling heuristics were pretty relaxed, resulting in more than one different strings of keyloggers being identified by their output. Besides the primary keylogger string being analyzed here, the rest were also interesting in terms of data. One batch of entries contained a URL and clearly marked “username” and “password” fields indicating that the keylogger was context-aware and was able to extract only the necessary information. Also, some other batches were site-exclusive and contained usernames and passwords, each one for a specific site indicating the presence of very particular keyloggers.

E. Information Gathered

We closely examined the stolen information being uploaded to Pastebin by the keylogger. Figure 5 presents a screenshot of an example entry. As one may notice, while the content is structured to group recorded keys by the application window in which they were recorded, the malware does not attempt to identify or organize the stolen

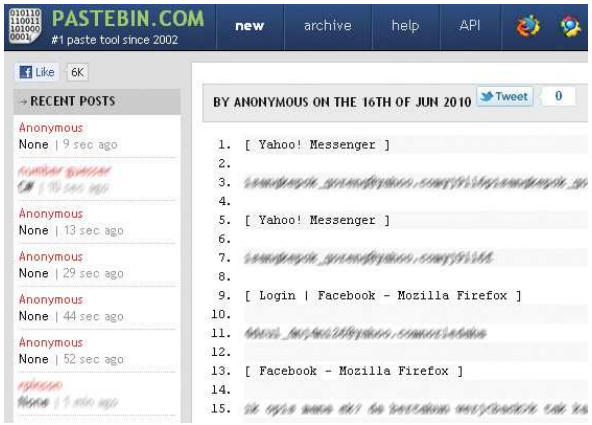

Figure 5: Screenshot of a Pastebin entry containing stolen information by Keylogger PBin.A.

information in any way. As a result, the uploaded entries were a chaotic stream of recorded keyboard keys. We noticed that most keys were recorded twice, indicating perhaps a bug in the keylogger's implementation or an excessive polling rate for keyboard events, such that it recorded almost each event twice. The next question was whether the keylogger uploaded stolen information in fixed-size batches. Thus, we studied the size distribution of the entries the malware uploaded on Pastebin. Figure 6 demonstrates that the malware does not operate on a fixed-number-of-bytes basis but rather employs a time frame after which any stolen information is uploaded. Let it be noted that the average size of an entry was 6 KBytes and the median was 1 KByte.

The next step was to attempt the reconstruction of the data using the following automated methodology: we treated each entry (paste) as standalone and grouped the streams of keys by their respective title in the application window, contained inside clearly marked blocks "[ ]". This resulted in a concatenated stream of keys per application window, for every uploaded entry. We removed duplicate keys, i.e. identical characters adjacent to each other but only in the cases where such combinations did not exist in the English language. We removed special keys such as <BACK> (backspace). We tokenized the stream using spaces as a delimiter and looked up the resulting words in a dictionary. For words not found, we looked them up in Google search and leveraged the "did you mean" or "search instead" feature of the search engine to resolve them. Finally, we ended up with a pretty much readable transcript of the user's input. Overall, we extracted hundreds of URLs, usernames, passwords, e-mail addresses and Instant Messenger conversations.

F. Correlation with Traditional Keyloggers

A subset of the stolen information entries contained the IP addresses of the respective infected computers. To investigate a possible correlation with hosts infected by traditional

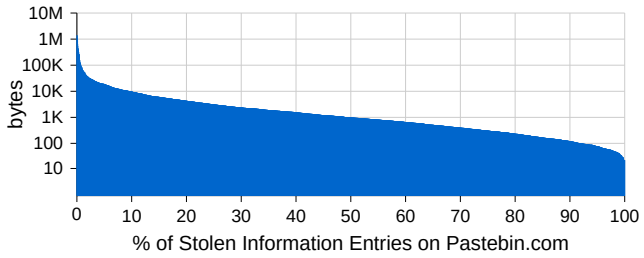

Figure 6: Distribution of stolen-data batches in terms of size.

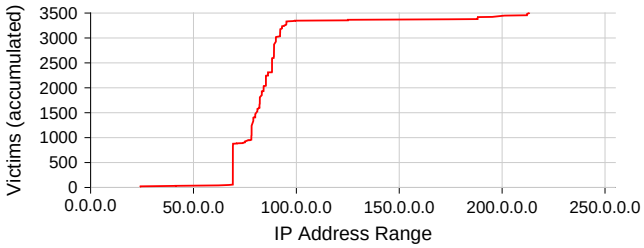

Figure 7: Cumulative distribution of keylogger victims across the IP address space.

keyloggers, we plotted (Figure 7) the cumulative distribution of victims across the IP address space. Interestingly, the most dense IP address range matches similar ranges found also by Holz et al. [9] when studying traditional keylogger data, which also match ranges of host computers known to be infected with malware such as the ZeuS botnet.

V. Outsourcing to the Cloud

In this section we discuss the trend of outsourcing malicious infrastructure to the Cloud. We present how attackers may benefit from the nature of the Cloud and detail the use of certain traits and features that may contribute to more sophisticated types of malware.

A. Economics

Public Cloud services are very cheap or free to use. One could consider examples from social networks such as Twitter, or services like Pastebin and Rapidshare. Traditionally, attackers had to maintain their own dedicated hardware and network connectivity.

B. Reliability

Cloud services aim to provide reliable uptime service for at least 99% of the time. Traditionally, attackers use compromised computers of victims which can be shut down or cleaned from infections. If that happens, the attacker will lose all stored information (e.g. keylogger data) and may also lose the command and control point which coordinates the activities.

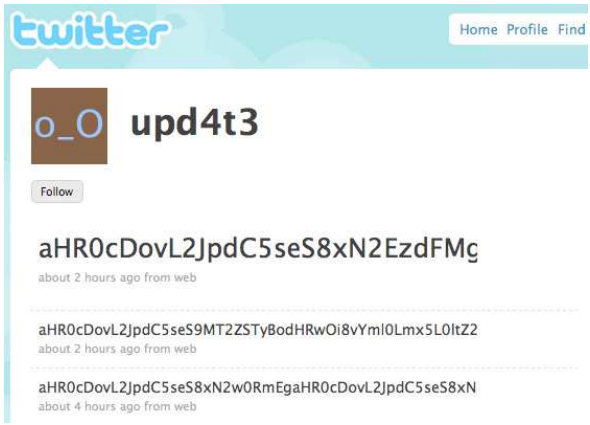

Figure 8: Twitter profile being used to command and control (C&C), pushing BASE64-encoded information.

C. Scalability

Cloud services are designed to scale in terms of storage, processing power or bandwidth. A private FTP site on a compromised server may run out of space, while an infected host acting as a command and control point may be overwhelmed by the number of network connections.

D. Unobservability

Cloud services offer practical anonymity. Traditionally, stolen information is transmitted directly back to the attacker or to a compromised system under his control. At the same time, malware in need of coordination (e.g. botnets) contacts an attacker-provided exchange point. An attacker employing his own resources to support the back-end infrastructure can be traced, identified and prosecuted. There is always the option of using a random compromised PC for that purpose but, as mentioned earlier, this may sacrifice reliability. On the other hand, infected victims employing the Cloud for malicious activities, does not differ from a large population of users using the Cloud for benign purposes. For instance, bots uploading stolen information on Pastebin blend in with a plethora of users sharing source code or arbitrary text on a daily basis. Moreover, in the Cloud it is much harder for a prosecuting authority to retrieve user records from an international service.

E. Plausible Deniability

An attacker contacting a compromised system or a private dropzone can be traced and charged with malicious activity. As no legitimate user will ever contact an infected workstation on a random TCP port bound by a backdoor, anyone who connects is probably malicious and his IP address can be used to identify him. For instance, a discovered dropzone employed by a keylogger may be kept online by security researchers to find out who will come and collect the stolen information. When this happens, it will be very hard for the attacker to deny his actions. On the other hand, an attacker using the Cloud to download stolen information does not differ from a plethora of other users using the Cloud for benign purposes. Moreover, he is able to create enough noise so that he cannot be tied beyond any doubt with malicious activity. For instance, one may visit all new entries on Pastebin, keep the ones uploaded by his keylogger and discard all the rest. Such activity will not be any different from, for example, our crawler presented in the previous section.

F. Unique Features and Flexibility

The Cloud offers a certain amount of flexibility in the use of the services it provides and a series of unique features that would otherwise be unavailable to the attacker. Here we discuss more sophisticated methods for storing keylogger data on Pastebin, along with other ways that the service's features can be leveraged for serving nefarious purposes, such as supporting botnets.

Sophisticated Keylogger Pastes. During our work on this paper, we have discovered discussions in underground forums of the security community, dating back to the beginning of 2010, exploring sophisticated ways of placing keylogger data on Pastebin and similar services. For instance, in a well known underground hacking forum, there is a discussion about using the arbitrary subdomain feature of Pastebin to create highly-dynamic private areas for uploading data. As mentioned earlier, any URL of the form http://foo.pastebin.com will produce a valid service page and any data uploaded through that page will not appear under the public home page of the service but under the new subdomain. Therefore one has to know "foo" in order to access that area and parse the "recent posts" list. The forum discussion suggests this technique as a way of uploading keylogger data without being noticed, contrary to the incident presented in this paper. Furthermore, they also suggest that this private subdomain should change frequently to avoid detection or prevent access to the entire set of uploaded data if the private URL is discovered. Their approach for the keylogger to discover the private subdomain, prior to uploading its data, is to poll an external Web page, that will act as a coordination point. However, one could implement a function that generates a subdomain string, using the current time and day or week or month, similar to traditional domain flux techniques employed by botnets [13]. This way no coordination point is required and the attacker remains in sync with the current Pastebin private area at all times.

Private C&C Pastes. In the same underground forum, one may read on employing Pastebin as a coordination point (Command and Control or C&C) for traditional botnets. The idea is to hide the address of the coordination point or change it continuously so to minimize damage if its current location is discovered. Furthermore, any pastes created will

```
Function UploadtoPastBin(ByVal text As String)
  Dim wc As New Net.WebClient
  Dim pl As String = "paste_code=" & text
  Return System.Text.Encoding.ASCII.GetString(
   wc.UploadData("http://pastebin.com/api_public.php",
    "POST", System.Text.Encoding.ASCII.GetBytes(pl)))
End Function
```

Listing 1: VB.NET Source Code for uploading to Pastebin.com Source

have a very small lifetime (minutes or hours) after which they will expire and be removed from the service. In detail, the discussion is about employing domain generation algorithms and applying their output to form private Pastebin subdomains. As mentioned in various posts, the creation of such a subdomain is instant, while traditional domain registration is expensive in terms of time, effort and money.

Overall, we believe that Pastebin is a very flexible service and can be leveraged for various malicious actions. As a matter of fact, in the same underground forum, Listing 1 was found in a Hacking forum titled "Post to PasteBin.com Source (Good for keyloggers)", a fact which indicates that attackers are actively developing and sharing modules of code to be used in malware.

VI. Discussion

Here we discuss how the security community can respond to this new generation of Cloud-supported malicious software. In other words, the public nature of the Cloud may present opportunities for security researchers to develop novel methods for identifying malware that employ such practices.

In general, free Cloud services are public. So far, we have discussed how attackers are able to, essentially, upload information to the Cloud in a practically anonymous fashion, and later access it to facilitate information exchange. The public nature of such services results in their content being indexed by search engines and cached. Therefore, it is easy for anyone, besides the attackers themselves, to locate and access information even if it has been removed from the original site, e.g. deleted by the attackers or taken down by the site's administrators. Moreover, malware, for instance botnets, usually targets a large number of victims. As a result, their large-volume output on the Cloud will be detected in a site like Twitter or Pastebin.

Traditional security practices inspect the *input of malware* to identify it and take measures against it. Such input includes inbound network behavior (e.g. network scanning or packet signatures) or malicious executables on PCs. However, lately a large portion of attacks has moved inside the user's Web browser. As a result, it is invisible at the network level, bypasses firewalls and intrusion detection systems. We propose the use of *malware output* in the Cloud (e.g. Twitter, Pastebin, etc.) as a heuristic for detecting and tracking new and emerging threats. For instance, we could monitor Twitter for a new botnet's command and control messages, and keylogger data or general information leakage on Pastebin.

Anomaly Detection on Cloud Services. Certain services, such as Twitter and Google search, employ heuristics to prevent abuse of their resources from automated scripts. Pastebin also intends to implement similar behavior. Such practice can be extended to form anomaly detection systems that not only block certain users from accessing the service, but also produce alerts or signatures for emerging automated activities. For instance, the appearance of a large number of IP and e-mail addresses in Pastebin entries, or other popular user-content hosting sites, can be detected and the victims be warned. Another example is the inspection of the indexed portions of malicious infrastructure. In detail, one can search the Web via popular search engines for certain keywords that will reveal lists of passwords or other sensitive information and perhaps investigate their replication across multiple sites on the Web.

Global View of the Attacks - Warning System. Traditional security practices require a plethora of distributed network monitors or sensors in order to acquire an accurate view of the attacks on a global scale. By monitoring the output of malware on the Cloud, one needs only a single point of observation in order to be able to inspect instances of malicious infrastructure on the Internet.

VII. Conclusion

In this paper we have focused on an emerging practice of attackers; outsourcing the infrastructure required to support their malicious acts to the Cloud. We present an empirical study on the first major usage of a Cloud service, Pastebin, by a keylogger for storing stolen information. We evaluate this trend by analyzing the nature of Cloud services in terms of facilitating the requirements of attackers and present the benefits of the Cloud along with unique features that provide new capabilities to the attackers and increase the efficiency of past practices. Finally, we discuss how a shift towards the Cloud may shape security monitoring practices, and propose countermeasures for such malicious activities.

Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement 257007. This work was supported in part by the Marie Curie Actions – Reintegration Grants project PASS. We thank the anonymous reviewers for their valuable comments. Georgios Kontaxis, Iasonas Polakis and Sotiris Ioannidis are also with the University of Crete.

REFERENCES

[1] "Kaspersky labs report: Irc botnets dying... but not dead," http://threatpost.com/en_us/blogs/report-irc-botnets-dyingbut-not-dead-111610.

[2] E. J. Kartaltepe, J. A. Morales, S. Xu, and R. Sandhu, "Social network-based botnet command-and-control: emerging threats and countermeasures," in *Proceedings of the 8th international conference on Applied cryptography and network security*, 2010.

[3] "Symantec - Official Blog: Twitter + Pastebin = Malware Update," http://www.symantec.com/connect/blogs/twitter-pastebin-malware-update.

[4] "Arbor Networks - Twitter-based Botnet Command Channel by Jose Nazario," http://asert.arbornetworks.com/2009/08/twitter-based-botnet-command-channel/.

[5] "The Real Face of KOOBFACE: The Largest Web 2.0 Botnet Explained," http://blog.trendmicro.com/the-real-face-of-koobface/.

[6] "10,000 Hotmail passwords mysteriously leaked to the web," http://www.theregister.co.uk/2009/10/05/hotmail_passwords_leaked/.

[7] "Mybanktracker - mastercard credit card numbers leaked by wikileaks supporters," http://www.mybanktracker.com/bank-news/2010/12/08/mastercard-credit-card-numbers/.

[8] "A Treasury of Dumps," http://blog.damballa.com/?p=695.

[9] T. Holz, M. Engelberth, and F. C. Freiling, "Learning more about the underground economy: A case-study of keyloggers and dropzones," in *ESORICS*, 2009.

[10] "Malware City Blog - Keyloggers Posting on Webpages," http://www.malwarecity.com/blog/keyloggers-posting-on-webpages-831.html.

[11] "Krebs on Security Blog - Cloud Keyloggers?" http://krebsonsecurity.com/2010/06/cloud-keyloggers/.

[12] D. Dagon, C. C. Zou, and W. Lee, "Modeling botnet propagation using time zones," in *NDSS*, 2006.

[13] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: analysis of a botnet takeover," in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009.