Master of Sheets: A Tale of Compromised Cloud Documents

Jeremiah Onaolapo
Northeastern University
jonaolapo@neu.ccs.edu

Martin Lazarov
University College London
martin.lazarov.12@alumni.ucl.ac.uk

Gianluca Stringhini
Boston University
gian@bu.edu

***Abstract*—As of 2014, a fifth of EU citizens relied on cloud accounts to store their documents according to a Eurostat report. Although useful, there are downsides to the use of cloud documents. They often accumulate sensitive information over time, including financial information. This makes them attractive targets to cybercriminals. To understand what happens to compromised cloud documents that contain financial information, we set up 100 fake payroll sheets comprising 1000 fake records of fictional individuals. We populated the sheets with traditional bank payment information, cryptocurrency details, and payment URLs. To lure cybercriminals and other visitors into visiting the sheets, we leaked links pointing to the sheets via paste sites. We collected data from the sheets for a month, during which we observed 235 accesses across 98 sheets. Two sheets were not opened. We also recorded 38 modifications in 7 sheets. We present detailed measurements and analysis of accesses, modifications, edits, and devices that visited payment URLs in the sheets. Contrary to our expectations, bank payment URLs received many more clicks than cryptocurrency payment URLs despite the popularity of cryptocurrencies and emerging blockchain technologies. On the other hand, sheets that contained cryptocurrency details recorded more modifications than sheets that contained traditional banking information. In summary, we present a comprehensive picture of what happens to compromised cloud spreadsheets.**

I. Introduction

It is hard to imagine life without online accounts, for instance, webmail accounts for business and personal communication, e-commerce accounts for online shopping, and cloud storage accounts for convenient document storage and sharing. As of 2014, 21% of EU citizens relied on cloud accounts to store their documents.[1] This shows the widespread utility and adoption of cloud storage platforms. However, there are downsides to the use of cloud accounts. Like most online accounts, cloud accounts often accumulate sensitive information over time, for instance, financial and personal secrets. This makes them attractive targets to cybercriminals seeking to steal and monetize such information [12].

It is hard to study attacker behavior in online accounts and documents unless one is in control of a large online service. Hence, there is limited research literature in this space. Previous work has shown that cybercriminals target online accounts and services to steal financial information from them, and trade stolen information via various outlets [5]. Such financial information includes payment card information, cryptocurrency wallets, and online banking details. The advent of cryptocurrencies has introduced a new wave of cybercriminals targeting users and platforms, and stealing digital money (cryptocurrency wallets), as seen in the 2014 high-profile attack on a cryptocurrency exchange known as Mt. Gox[2] ($460 million in losses).

To understand what happens to compromised cloud documents containing financial information, we set up 100 fake payroll sheets comprising 1000 fake records of fictional individuals. For comparison, only five decoy sheets were deployed in a related previous study [8]. We scaled up experiments by a factor of 20, compared to [8]. We populated the sheets with traditional bank payment information, cryptocurrency details (unlike [8] that relied on traditional bank information only), and payment links. We also installed scripts in the sheets to notify us about the activity of visitors in them. To lure cybercriminals and other visitors into visiting the sheets, we leaked links pointing to the sheets via paste sites. By doing so, we mimicked the modus operandi of cybercriminals that steal and distribute stolen financial information online.

We ran experiments and collected data for a month. We observed 235 accesses across 98 sheets. Two sheets were not opened. We also recorded 38 modifications in 7 sheets. We present detailed measurements and analysis of accesses, modifications, edits, and devices that visited fake payment URLs in the sheets (with emphasis on IP addresses, browsers, and operating systems).

Contrary to our expectations, bank payment URLs received many more clicks than cryptocurrency payment URLs despite the popularity of cryptocurrencies and emerging blockchain technologies. On the other hand, sheets that contained cryptocurrency details recorded more modifications than sheets that contained traditional banking information; 38 modifications to 7 cryptocurrency sheets and no modification to bank sheets. We also observed attempts by cybercriminals to cover their tracks—one out of every three persons that visited payment URLs covered their tracks while doing so, by visiting via TOR network.

In summary, we present a comprehensive picture of what happens to compromised Google spreadsheets. The findings in this paper will help other researchers to understand what

[1] https://ec.europa.eu/eurostat/statistics-explained/index.php/Internet_and_cloud_services_-_statistics_on_the_use_by_individuals

[2] https://www.wired.com/2014/03/bitcoin-exchange/

happens to stolen cloud documents and providers of cloud services looking to understand ways to secure accounts and assets on those cloud services. This is essential because our daily activities depend heavily on cloud services.

II. Background

In this section, we describe cloud documents, with specific focus on Google Sheets, and explain why Google Sheets constitutes a good fit for our experiments.

A. Cloud documents

Word processing, desktop publishing, and data processing tasks can be carried out on local machines using desktop tools such as Microsoft Word, Scribus, and Apache OpenOffice Calc, among others. It is also possible and easy to use cloud-based tools for such tasks. They usually do not require complex installation processes unlike their desktop counterparts. They also allow users to collaboratively edit documents from any location. Examples of cloud-based tools for creating and editing cloud documents include Google Sheets, Microsoft Office 365, and Zoho Office Suite. These tools offer remote document hosting and editing services, and are accessible via a web browser. Next, we describe Google Sheets, the cloud-based platform that supported our experiments in this paper.

B. Google Sheets

Here, we focus on Google Sheets, a cloud-based data processing tool that allows users to create and modify sheets, and carry out data processing tasks on those sheets. Google Sheets also enables users to extend the functionalities of their sheets by incorporating scripts in them, leveraging the power of Google Apps Script[3] (a scripting engine for building lightweight web applications and augmenting Google Apps). This makes Google Sheets a good fit for our experiments since the embedded Google Apps Script engine allows us to instrument sheets to "phone home" (report activity data).

To create sheets, a user will first have to set up at least one Google account to host sheets. Afterwards, the user can create new sheets via a web browser. Alternatively, users can upload existing sheet data, for instance, comma-separated values (CSV) files that already contain data formatted in rows and columns. Users can edit cells in sheets, delete rows and columns of cells, perform computations and transformations on cells, and delete entire sheets, among other operations.

For collaborative purposes, the owner of a sheet can configure the sheet to allow other users or visitors to view, comment on, or edit the sheet. Inviting collaborators to such sheets usually involves explicitly granting them specific permissions (to *view* or *edit*). The sheet owner can also generate a *long link* that points to the sheet, such that anyone that knows the long link can view or edit the sheet, depending on the privilege level assigned to the long link. The sheet owner can then send the long link to collaborators. They will visit the long link to gain access to the sheet. Figure 1 shows an example of a sheet configuration setting that allows visitors with knowledge of the long link to edit the sheet.

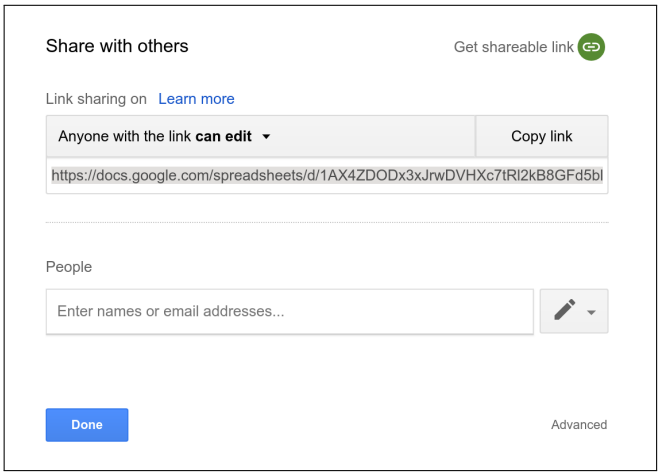

Fig. 1. One of the ways to share a sheet with collaborators is by generating a long link that points to the sheet. In this example, anyone that knows the long link (highlighted in grey) can view and edit this sheet. Alternatively, the sheet owner can explicitly enter collaborators via the "People" field.

III. Method

In this section, we describe the process of creating and instrumenting sheets prior to experiments, and how we exposed the sheets to cybercriminals. We also describe the data collection infrastructure that powered experiments in this paper.

A. Creating honey sheets

We created sheets containing two types of financial information, namely traditional bank payment information (bank account numbers and sort codes) and cryptocurrency information (Bitcoin addresses). We designed the sheets to look like payroll spreadsheets by including salary information. Note that the sheets in [8] did not include any cryptocurrency information while half of our sheets in this paper did.

Fake data in cells. We created 100 sheets and generated fake personal data to fill their rows and columns (1000 records). We also included salary information from `Monster.co.uk`,[4] a website that provides salary information to the general public.

Fake banking information. We included traditional banking information (fake sort codes and fake bank account numbers) in half of the sheets, following the conventions of the following popular UK banks: HSBC, Lloyds Bank, Santander, Barclays, and Standard Chartered.

Fake Bitcoin addresses. We needed fake but realistic-looking cryptocurrency information for the other half of honey sheets. To this end, we generated 500 fake Bitcoin addresses, following Bitcoin address specifications described on a Bitcoin wiki,[5] and included them in 50 sheets.

[3] https://developers.google.com/apps-script/overview

[4] https://www.monster.co.uk/career-advice/article/uk-average-salary-graphs

[5] https://en.bitcoin.it/wiki/Address

Honey URLs. To observe if visitors to the sheets were going to carry out attacks on the "account owners" listed in the sheets, we included some fake payment URLs, which we refer to as *honey URLs*, in the sheets. These honey URLs, which point to non-existent pages on bank websites and cryptocurrency exchanges, allow us to track clicks on them. To track clicks, we leveraged the functionality that *link shorteners* provide. By including short URLs (honey URLs) in the sheets instead of actual destination URLs, we achieve our goal of click tracking (via click analytics functionality provided by link shorteners) and hide the true destination of honey URLs. We chose `cutt.ly`, a link shortener that provides a free click analytics dashboard and an API that allows easy download of click analytics data.

This concludes the process of creating honey sheets and adding fake financial data to them. Next, we describe the data collection infrastructure that was deployed to monitor honey sheets.

B. Data collection

In this section, we present the main components of the honeypot infrastructure that was deployed to collect data from honey sheets (see Figure 2). Next, we describe its key components.

Safehouse webmail account. We installed scripts (Google Apps Script) in each sheet to report changes in the sheet back to us via a dedicated *safehouse webmail account*. Precisely, the scripts send notification emails containing periodic snapshots of sheets to the safehouse webmail account. We then retrieve those emails via an email client and parse them to compare snapshots of sheets automatically. This allows us to record differences in snapshots and changes in sheets over time.

Honey URL analytics. As mentioned in Section III-A, short URLs in honey sheets provide information about clicks on them. This includes information about click origin (country), click count, and device information (that is, the device that was used to click on the link). We collected click analytics data once daily by leveraging `cutt.ly` analytics API (recall that we used `cutt.ly` link shortener service to create honey URLs), and stored it locally in JSON files, for later analysis.

Bouncy web server. The `cutt.ly` analytics API provides useful click analytics data but does not reveal IP addresses of people that visit honey URLs. To overcome this limitation, we configured a third of the `cutt.ly`-generated honey URLs to point to a custom web server under our control, otherwise known as a *bouncy web server*. This web server enables us to record IP addresses and additional header information (which short URL analytics do not provide). On receiving a request for a web resource, the bouncy web server parses the request path and redirects the visitor to a bank website, if the request path contains the token "banking-8102," a cryptocurrency exchange website if the request path contains "crypto-8102," or `google.com` if the request path does not contain either of those tokens.[6] The "bouncy" behavior of the web server helps to keep up the appearance of visiting "payment links" and hides the existence of the bouncy web server.

Health inspector. To inspect the state of the honeypot system (to ensure that all components work as expected), we periodically run the health inspector to check that latest activity reports have been retrieved from the safehouse webmail account. It also examines click analytics data for recency. Out-of-date data indicates that one or more components of the honeypot infrastructure have failed.

[6]In case the reader wonders what "8102" stands for in request paths, it has no special significance. It is simply year "2018" written backwards.

TABLE I
TO LURE VISITORS TO HONEY SHEETS, WE LEAKED LONG LINKS POINTING TO THE SHEETS THROUGH PASTE SITES ON THE SURFACE WEB AND THE DARK WEB. WE CHOSE THESE PASTE SITES BECAUSE THEY ALLOW PUBLIC PASTES.

Name	*Type*	*URL*
Pastebin	Surface Web	https://pastebin.com/
Paste.org.ru	Surface Web	http://paste.org.ru/
Stronghold	Dark Web (via TOR)	http://nzxj65x32vh2fkhk.onion/

C. Leaking long links

Previous work has shown that cybercriminals often post samples of their loot via online outlets usually to brag about their prowess or attract potential buyers [12]. Mimicking their modus operandi, we leaked long links[7] pointing to the sheets on paste sites (see Table I), to lure cybercriminals to visit the sheets. Each long link was leaked along with a short description, for instance, "leaked payroll" or "bitcoin payment lists." We configured each sheet in a way that anyone could access and edit it, provided they know the long link that points to it.

Prior to leaking the 100 long links, we divided them into five chunks, each chunk comprising 20 long links. We leaked all chunks twice daily to ensure good temporal coverage on paste sites, thus compensating for timezone differences among visitors to the paste sites. We also randomized the order of links in each chunk prior to leaking, thus ensuring that each long link had a fair chance of being visited. After leaking the long links, we recorded accesses to sheets and tracked clicks on honey URLs inside them.

D. Threats to validity

We acknowledge that there are some factors that may affect the validity of our findings. First, our honey sheet data comprises fake financial data which may be obvious under close scrutiny, and can possibly influence the behavior of visitors. Second, our honey URLs (embedded in sheet data) are short URLs, and short URLs are generally treated with suspicion. This may negatively affect the perception of visitors to honey sheets. Third, we leaked long links pointing to the sheets through paste sites only. Our findings may not be representative of malicious activity in cloud documents stolen via other outlets, for instance, malware-laden endpoints or

[7]Long links look like this: https://docs.google.com/spreadsheets/d/1AX4ZDODx3J***. On the other hand, honey URLs look like this: https://cutt.ly/B***.

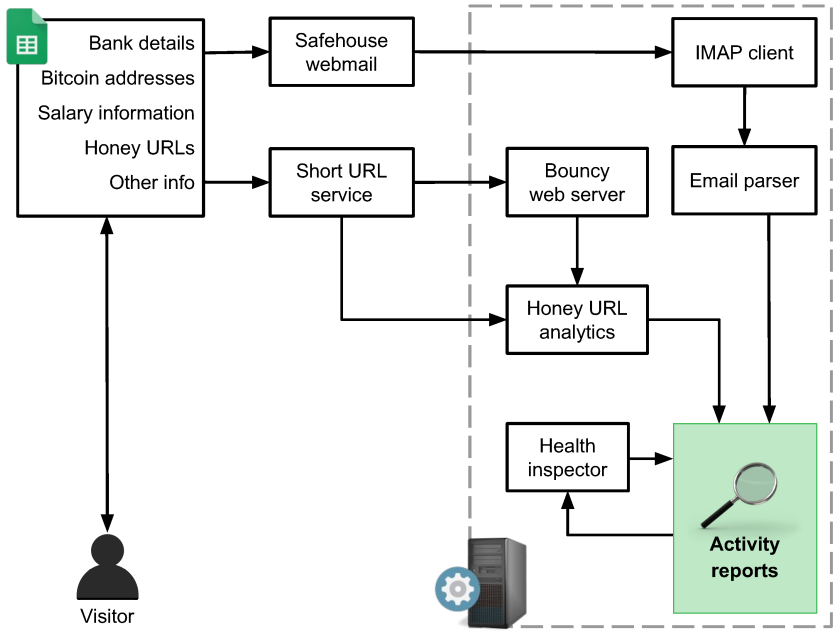

Fig. 2. Honey sheets infrastructure.

underground forums. Finally, there is also the possibility that automated tools (crawlers) visited the long links in addition to human visitors. This may affect the validity of our findings. To mitigate this risk in future work, it is possible to incorporate an additional CAPTCHA-like step in the process of accessing the sheets to ensure that only manual accesses by humans pass through. This can be achieved by leaking links that point to a web domain under our control which will serve up an interstitial page containing the CAPTCHA. If a visitor passes the CAPTCHA (and thus prove they are indeed human), they will be redirected to the sheet that they were trying to access in the first place. However, this approach may discourage visitors from proceeding because of the increased cognitive workload that CAPTCHA solving involves.

Despite these concerns, this paper offers insights into malicious activity in compromised cloud documents.

E. Ethics

The experiments in this paper involve deceiving cybercriminals into interacting with cloud documents. In line with standard ethical practices, we took the following precautions. First, we used fake financial data (randomly generated) in the sheets. Thus, we ensured that no real person or account was harmed in our experiments. Second, to avoid spamming other accounts, we did not leak credentials of the Google accounts that hosted our honey sheets. We only leaked the long links that point to honey sheets, thus limiting the possible harm that our experiments may cause otherwise. Third, we obtained approval from our institution prior to running experiments.

IV. Data Analysis

In this section, we present detailed measurements of visitor activity in the honey sheets.

A. Activity overview

We conducted experiments from July 11, 2018 until August 14, 2018. During this period, 98 sheets were accessed 235 times.[8] These sheets comprise 48 sheets containing banking information and 50 sheets containing cryptocurrency information. We recorded 38 modification events during which 7 sheets were modified by visitors. We observed 219 clicks on honey URLs. Those clicks originated from 30 countries.

B. Timing of activity in sheets

Leak to first access. First, we set out to understand how long it took for visitors to access the sheets after we leaked long links pointing to them. Let us denote the time of first leak as t_{leak}. For each opened sheet, we record the time of its first open event (first access) as t_0 and compute the time lag between leak and first access as $t_0 - t_{leak}$. Figure 3 shows a CDF of time lags. Less than 10% of opened sheets were visited within the first 22 hours since first leak. However, accesses increased rapidly afterwards—by the 25th hour since first leak, 80% of the sheets had been opened. It is possible that the initial time lags between first leak and first accesses were due to reluctance of visitors to visit links, since links can be potentially malicious, generally speaking.

[8]We define an access as a file open event, in other words, a sheet open event.

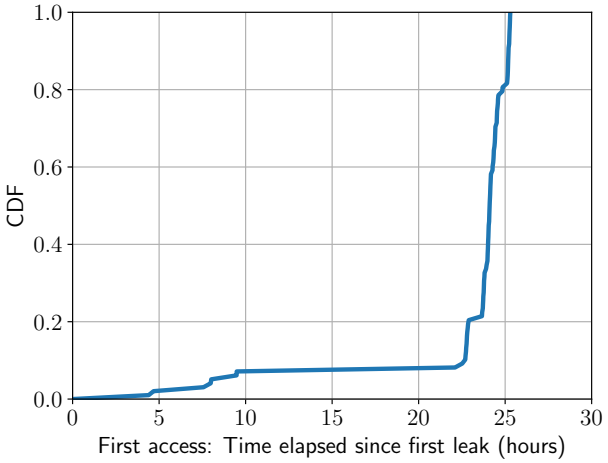

Fig. 3. Time difference between leak and first access (CDF). Less than 10% of opened sheets were visited within the first 22 hours since first leak. However, accesses increased rapidly afterwards—by the 25th hour since first leak, 80% of the sheets had been opened.

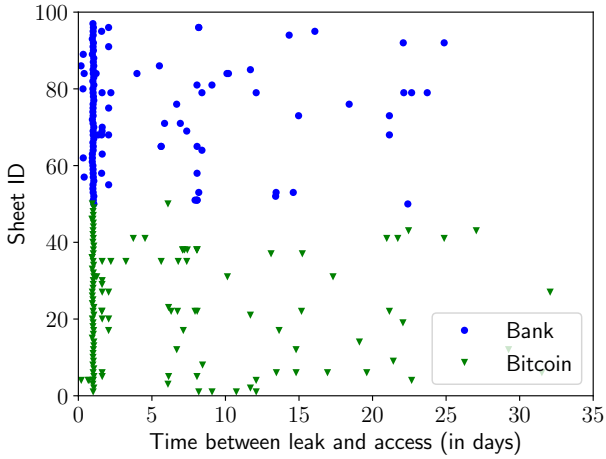

Fig. 4. Timeline of accesses. 98 sheets received 235 accesses. These comprise 48 bank sheets ("Bank") and 50 cryptocurrency sheets ("Bitcoin"). Note that accesses to cryptocurrency sheets spanned a longer time period than accesses to bank sheets.

Timeline of accesses. Next, we set out to understand the spatial patterns (with respect to time) of all accesses during experiments. 98 sheets received 235 accesses. These comprise 48 bank sheets and 50 cryptocurrency sheets. Let us denote the time of a given access as t_a and the time of first leak as t_{leak}. For each access, we computed its relative access time as $t_a - t_{leak}$. We then plotted a timeline of accesses (see Figure 4), with the time of first leak t_{leak} as the reference point. Figure 4 corroborates our previous findings in Figure 3—it shows sparse accesses during the first day since the initial leak. From the beginning of the second day, it shows a sharp increase in accesses to bank sheets and cryptocurrency sheets. Figure 4 also shows that accesses to cryptocurrency sheets spanned a longer time period than accesses to bank sheets—the last access we recorded in a bank sheet was on the 25th day after first leak, whereas we observed accesses in cryptocurrency sheets afterwards. Next, we study the modifications that visitors made to some of the honey sheets.

C. Modifications and edits in sheets

We observed 38 modifications in 7 cryptocurrency sheets. No bank sheet was modified. A closer look at the modified sheets revealed that most of the modifications were recorded when visitors resized columns in sheets (changes made to sheet structure are recorded as modifications). This happened in cryptocurrency sheets because visitors wanted to view Bitcoin addresses, which are long strings, partly obscured in the default states of the cryptocurrency sheets. Interested visitors thus had to resize the Bitcoin wallet column for a better view.

Next, we studied modifications that resulted in changes to values of cells in sheets (otherwise known as *edits*). We observed that a Bitcoin address in cell D4 of a cryptocurrency sheet was replaced with another Bitcoin address. We looked up the new Bitcoin address on a Bitcoin address verification tool (`blockchain.info`), but it returned no result. We also looked up our list of fake Bitcoin addresses to see if it was copied from another cryptocurrency sheet, and this lookup also yielded no result. This indicates that the Bitcoin address entered by the visitor was either a yet-to-be-used Bitcoin address that belonged to them (with intent to commit fraud by receiving payments meant for the original recipient listed on the compromised sheet), or a fake Bitcoin address made up by them.

In another cryptocurrency sheet, we observed that one of its records (fields B10—E10) had been replaced with values that were exactly the same record. This indicates that a visitor (accidentally) "cut" the original values and pasted them back in the sheet. The next record was modified similarly, with most values intact, except for the Bitcoin address field. The Bitcoin address of that record was replaced with a different string[9] that did not fit the specification of Bitcoin addresses and was also absent from the list of fake Bitcoin addresses we initially generated. We observed another edit in a separate cryptocurrency sheet in which the Bitcoin address of one of its records was replaced by a copy of the string mentioned previously. This indicates that the same visitor modified both sheets (by pasting that string in both sheets).

In summary, the majority of sheet modifications comprised column resizing actions by visitors, while actual edits involved changes to Bitcoin addresses. Next, we study the patterns of clicks on payment URLs within the sheets.

D. Click activity

Recall that we included two types of honey URLs in the sheets, namely bank URLs and cryptocurrency URLs. In this section, we present measurements of clicks on those URLs. We recorded 219 clicks on honey URLs, comprising 135 clicks on bank URLs and 84 clicks on cryptocurrency URLs. Those

[9]String: `qzpweklwh85u0h2x44ffv4tsfhxww96v8c7kylnwyu`. We are yet to figure out what it stands for.

TABLE II
SUMMARY OF CLICKS ON HONEY URLS. DIRECT HONEY URLS LEAD VISITORS DIRECTLY TO THE DESTINATION URL (BANK OR CRYPTOCURRENCY PAGE), WHILE BOUNCY URLS SURREPTITIOUSLY ROUTE VISITORS THROUGH OUR BOUNCY WEB SERVER BEFORE REDIRECTING THEM TO THE DESTINATION URL.

Type of honey URL	*Click count*
Direct bank	98
Bouncy bank	37
Direct Bitcoin	69
Bouncy Bitcoin	15
Total	**219**

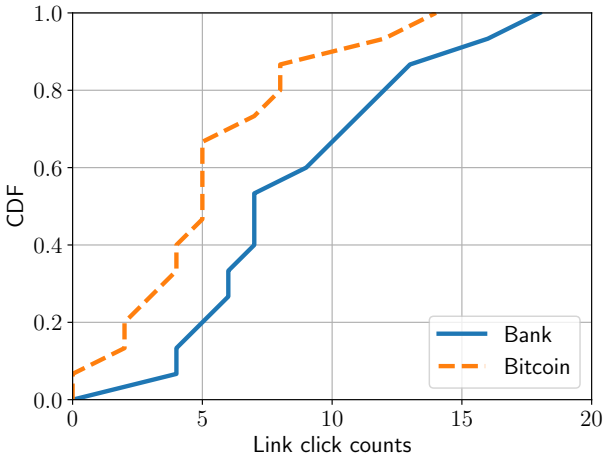

Fig. 5. URL click counts. The bank link with the highest click count recorded 18 clicks while the cryptocurrency link with the highest click count recorded 14 clicks.

clicks originated from 30 countries. We present a detailed summary of click counts in Table II.

Click counts. We wanted to observe differences in clicks on bank URLs and cryptocurrency URLs. To this end, we counted those clicks, by link type, and plotted CDFs of click counts. Contrary to our expectations, honey URLs of the bank type consistently received more clicks than honey URLs of the cryptocurrency type. We expected the opposite to happen, given the recent surge in interest of the general public in cryptocurrencies and blockchain technologies. The bank link with the highest click count recorded 18 clicks while the cryptocurrency link with the highest click count recorded 14 counts, as Figure 5 shows.

Statistical test. To test the statistical significance of differences in click counts by link type, we relied on the two-sided Kolmogorov-Smirnov (KS) test to examine the CDFs in Figure 5. The null hypothesis states that both samples under examination belong to identical statistical distributions. The output of the test is a KS statistic and p-value. A small KS statistic or high p-value shows that we cannot reject the null hypothesis. The KS test returned an inconclusive result (statistic=0.4667, p-value=0.0515).

Click locations. During the analysis of `cutt.ly` click analytics data (on honey URLs), we collated a list of countries that clicks originated from. We also carried out geolocation (country-level resolution) of IP addresses that visited our bouncy web server. We used *IP-API*,[10] an IP geolocation service that provides timezone and location information for IP addresses, to achieve this. We then plotted the resulting locations, comprising 30 countries, on a world map, as shown in Figure 6. As the map shows, most of the countries are located in Europe. It is possible that some visitors connected to the sheets and clicked on honey URLs via proxies or VPNs. We found some TOR exit nodes (see Section IV-E) among the IP addresses that visited the bouncy web server via honey URLs.

E. System configuration of accesses

In this section, we study the devices that visitors used while clicking on honey URLs in sheets.

IP addresses and TOR exit nodes. Recall that a subset of honey URLs point to our bouncy web server, which allows us to collect IP addresses of visitors clicking on them, in addition to click analytics. We recorded 35 IP addresses that visited the bouncy web server from 20 countries. 12 of the IP addresses were TOR exit nodes. Note that this is only a subset of the IP addresses that visited the honey sheets—not all visitors to honey sheets click on honey URLs. Also, only a third of our honey URLs, the ones that point to the bouncy web server, can track IP addresses. Hence we have a partial view of IP addresses. Nevertheless, it is surprising that 34% of the recorded IP addresses were TOR exit nodes. It shows that one out of every three persons that visited our honey URLs covered their tracks while doing so, by visiting via TOR network.

Browsers. We extracted browser information from click analytics data and grouped browser-clicks by URL type. Figure 7 shows the distribution of browsers that were used to visit honey URLs. Visitors that clicked on honey URLs had an unusual preference for Firefox—the top browser responsible for more than 80% of clicks on bank and cryptocurrency URLs. We also observed clicks from Chrome, Opera, and other browsers.

Operating Systems. We also extracted information about the operating systems of devices that connected to honey URLs. Visitors that clicked on honey URLs had a preference for Windows devices, as shown in Figure 8. In cryptocurrency URL clicks, even though Windows devices dominate, we observe a slightly wider range of operating systems than devices that clicked on bank URLs. In both URL types, we observed a small fraction of clicks from Android devices. Cryptocurrency URLs recorded a tiny fraction of visits from iPhones and Linux devices, both of which were absent from clicks on bank URLs. This indicates that cryptocurrency URLs attracted a slightly more diverse set of visitors than bank URLs.

V. DISCUSSION

Implications of our findings. We observed differences in document modifications, depending on the content of the

[10]http://ip-api.com

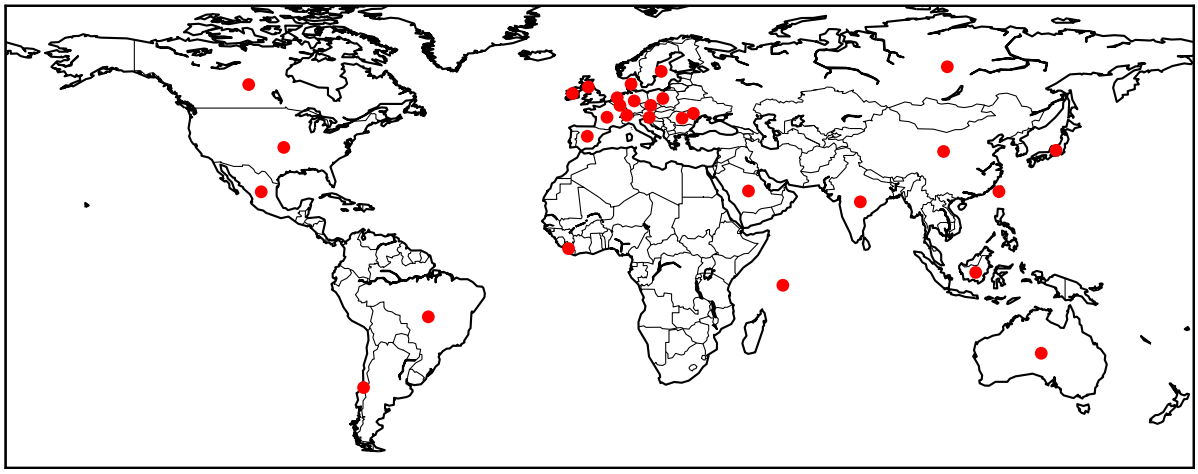

Fig. 6. Origins of clicks on honey URLs.

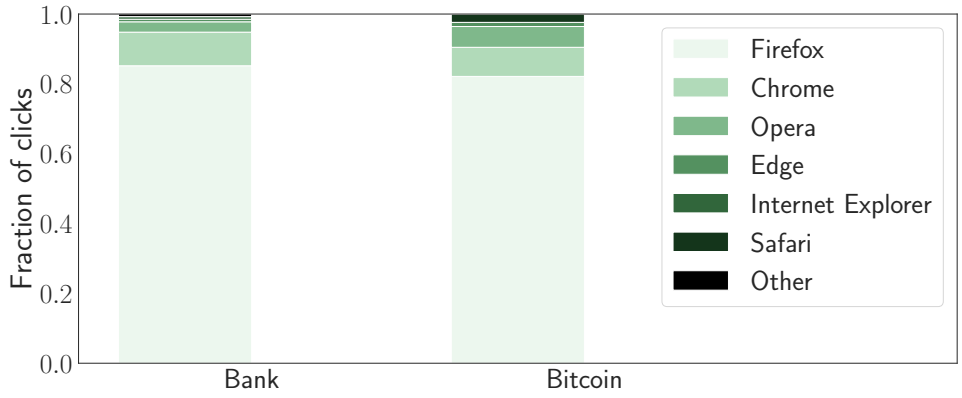

Fig. 7. Distribution of browsers.

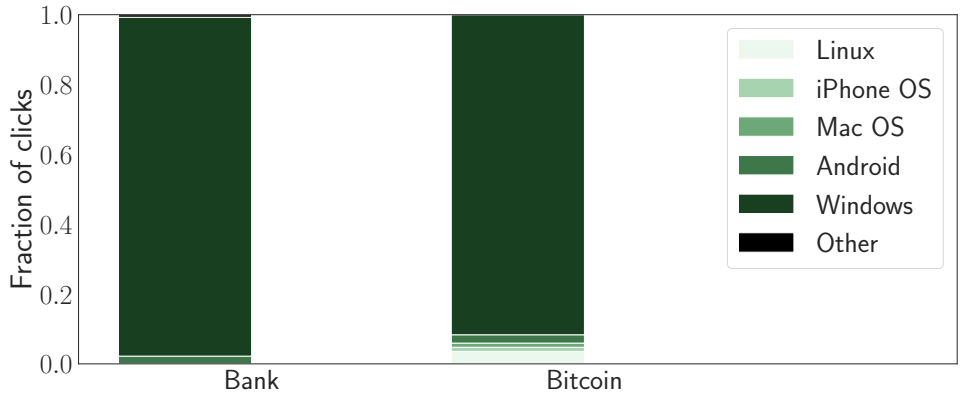

Fig. 8. Distribution of operating systems.

documents. Particularly, documents that contained cryptocurrency information were subject to more modifications than documents containing banking information, despite receiving fewer accesses than sheets that contained bank information. Similarly, we observed differences in URL clicking behavior across different types of URLs in documents. This knowledge can be used during the development and training of detection systems to protect cloud documents. Such detection systems could be built to adapt their statistical models depending on document type and content. We also recorded defacement activity (and "cut-and-paste" activity) in the sheets. For instance, we observed an instance in which a meaningless string was pasted in two sheets, among other modifications. Such behavior likely deviates from regular everyday use of cloud documents and could potentially help in identifying anomalous behavior in them. Even though such actions occur during benign account usage, a surge in potentially destructive activity could be flagged as an anomaly (potentially malicious) and they would trigger additional automatic checks and reviews by human operators. In other words, it is possible to develop and train tools based on machine learning methods that do not require balanced training datasets of positive and negative examples (one-class Support Vector Machines, for instance) on "normal" document activity. Malicious activity will likely deviate from normal activity and will thus be flagged as such.

Limitations. We had limited visibility into the sheets because of unauthenticated accesses (visitors do not need to authenticate before interacting with the sheets). As a result, we were able to record only a subset of IP addresses that visited the sheets, not the entire set. Also, it is hard to update scripts in honey sheets—such updates must be carried out and tested manually in each sheet. Once experiments are in motion, such updates may taint experimental results.

Future work. In the future, we plan to continue exploring the ecosystem of stolen accounts and gaining a better understanding of the underground economy surrounding them. We will explore ways to make honey documents more believable to attract more cybercriminals and keep them engaged. We intend to set up additional scenarios, such as studying attackers who have a specific motivation, for example, compromising accounts that belong to political activists. We also intend to carry out further studies on the impact of demographic attributes (including employment status, religious affiliation, and political affiliation, among others) of online accounts and documents on the behavior of cybercriminals that gain illicit access to them. These will provide comprehensive insights into attackers' motivations and resulting activity.

VI. Related Work

Honeypots based on online accounts have been deployed to study social spam in OSNs [16], [9], [14] and email spam [13]. DeBlasio et al. [3] studied compromised websites by registering on those websites using honey webmail accounts. They monitored illegitimate accesses to the honey accounts that happened as a result of data breaches on those websites. They observed attackers that leveraged the problem of password reuse across online services. Other studies also investigated the behavior of criminals in compromised webmail and cloud document accounts via honeypots [2], [10], [8]. Kedrowitsch et al. [7] explored ways to improve Linux sandboxes for analysis of evasive malware. Barron and Nikiforakis [1] deployed honeypot machines and observed how system properties of those machines influenced the behavior of attackers. Kapravelos et al. [6] used honeypots to study malicious browser extensions and highlighted the huge risks that malicious browser extensions pose to users. Wang et al. [15] investigated malicious web pages by deploying VM-based honeypots that ran on vulnerable operating systems. Stone-Gross et al. [11] hijacked the Torpig botnet for ten days by taking advantage of weaknesses in communication protocols of the botnet. Stringhini et al. [14] studied social spam using 900 honeypot profiles. Bursztein et al. [2] investigated manual hijacking of accounts rather than automatic hijacking by botnets. They show that manual hijacking is not common, and demonstrate that phishing is the primary method that manual hijackers use to acquire user credentials. To understand the phishing ecosystem, Han et al. [4] deployed sandboxed phishing kits, recorded live interactions of various parties with those kits, and shed light on the phishing life cycle.

VII. Conclusion

To shed light on what happens to compromised cloud documents, we set up 100 sheets containing fake financial records of fictional individuals. We then lured cybercriminals and other visitors into visiting the sheets. We observed 235 accesses across 98 sheets and 38 modifications in 7 sheets. We presented detailed measurements and analysis of resulting activity in the sheets. In summary, we have presented a comprehensive picture of what happens to compromised cloud spreadsheets.

Acknowledgments

We thank the anonymous reviewers for their comments. This work was supported by EPSRC grant EP/N008448/1 and a Google Faculty Award. Jeremiah Onaolapo was supported by the Petroleum Technology Development Fund (PTDF) of Nigeria.

References

[1] T. Barron and N. Nikiforakis, "Picky Attackers: Quantifying the Role of System Properties on Intruder Behavior," in *Annual Computer Security Applications Conference (ACSAC)*, 2017.

[2] E. Bursztein, B. Benko, D. Margolis, T. Pietraszek, A. Archer, A. Aquino, A. Pitsillidis, and S. Savage, "Handcrafted Fraud and Extortion: Manual Account Hijacking in the Wild," in *ACM Internet Measurement Conference (IMC)*, 2014.

[3] J. DeBlasio, S. Savage, G. M. Voelker, and A. C. Snoeren, "Tripwire: Inferring Internet Site Compromise," in *ACM Internet Measurement Conference (IMC)*, 2017.

[4] X. Han, N. Kheir, and D. Balzarotti, "PhishEye: Live Monitoring of Sandboxed Phishing Kits," in *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[5] A. Haslebacher, J. Onaolapo, and G. Stringhini, "All your cards are belong to us: Understanding online carding forums," in *APWG Symposium on Electronic Crime Research (eCrime)*, 2017.

[6] A. Kapravelos, C. Grier, N. Chachra, C. Kruegel, G. Vigna, and V. Paxson, "Hulk: Eliciting malicious behavior in browser extensions," in *USENIX Security Symposium*, 2014.

[7] A. Kedrowitsch, D. D. Yao, G. Wang, and K. Cameron, "A First Look: Using Linux Containers for Deceptive Honeypots," in *Workshop on Automated Decision Making for Active Cyber Defense*, 2017.

[8] M. Lazarov, J. Onaolapo, and G. Stringhini, "Honey Sheets: What Happens to Leaked Google Spreadsheets?" in *USENIX Workshop on Cyber Security Experimentation and Test (CSET)*, 2016.

[9] K. Lee, J. Caverlee, and S. Webb, "The social honeypot project: protecting online communities from spammers," in *World Wide Web Conference (WWW)*, 2010.

[10] J. Onaolapo, E. Mariconti, and G. Stringhini, "What Happens After You Are Pwnd: Understanding the Use of Leaked Webmail Credentials in the Wild," in *ACM Internet Measurement Conference (IMC)*, 2016.

[11] 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 *ACM Conference on Computer and Communications Security (CCS)*, 2009.

[12] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna, "The underground economy of spam: A botmaster's perspective of coordinating large-scale spam campaigns," in *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2011.

[13] G. Stringhini, O. Hohlfeld, C. Kruegel, and G. Vigna, "The harvester, the botmaster, and the spammer: on the relations between the different actors in the spam landscape," in *ACM Symposium on Information, Computer and Communications Security*, 2014.

[14] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting Spammers on Social Networks," in *Annual Computer Security Applications Conference (ACSAC)*, 2010.

[15] Y.-M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King, "Automated web patrol with strider honeymonkeys," in *Symposium on Network and Distributed System Security (NDSS)*, 2006.

[16] S. Webb, J. Caverlee, and C. Pu, "Social Honeypots: Making Friends With A Spammer Near You," in *Conference on Email and Anti-Spam (CEAS)*, 2008.