The Cause of All Evils: Assessing Causality Between User Actions and Malware Activity

Enrico Mariconti, Jeremiah Onaolapo, Gordon Ross, and Gianluca Stringhini

University College London
{e.mariconti,j.onaolapo,g.stringhini}@cs.ucl.ac.uk,g.ross@ucl.ac.uk

Abstract

Malware samples are created at a pace that makes it difficult for analysis to keep up. When analyzing an unknown malware sample, it is important to assess its capabilities to determine how much damage it can make to its victims, and perform prioritization decisions on which threats should be dealt with first. In a corporate environment, for example, a malware infection that is able to steal financial information is much more critical than one that is sending email spam, and should be dealt with the highest priority. In this paper we present a statistical approach able to determine *causality relations* between a specific trigger action (e.g., a user visiting a certain website in the browser) and a malware sample. We show that we can learn the typology of a malware sample by presenting it with a number of trigger actions commonly performed by users, and studying to which events the malware reacts. We show that our approach is able to correctly infer causality relations between information stealing malware and login events on websites, as well as between adware and websites containing advertisements.

1 Introduction

The malware problem is becoming more difficult to tackle as time passes: already in 2013, a new malware sample was released in the wild each second [4]. Malware can span from threats to the critical infrastructure of a country [12] to malicious software sending unwanted content to Internet users, such as email spam [18]. The huge amount of malware threats observed every day requires an effective risk assessment and appropriate prioritization, in which the potential damage that each piece of malware can cause to companies and their customers needs to be efficiently determined. For example, the spam email malware constitutes an annoyance to any infected network, but is much less critical than a banking malware that would steal financial information from the company. For this reason, certain types of malware must be dealt with higher priority than others.

The first step that is needed to tackle each malware threat is analyzing the malware sample to understand its capabilities and the potential damage that the sample can cause to the victim network. In particular, it is important to determine the purpose of the malware infection. Knowing the type of infection can guide mitigation efforts, and allows to prioritize cleanup of an infection over an other. This prioritization process is called *triaging*. In the past, malware triaging approaches used binary analysis to make their decisions [5, 9]. Although effective, these approaches have limitations due to the capability of cybercriminals to heavily harden their binary code [8, 15].

In this paper we tackle the problem of malware triaging from a different angle. Our basic insight is the following: *many categories of malware require victim activities to be triggered*. For example, information stealing malware sends stolen credentials to its controller only after it observes the user inputing these credentials into a web login form. If we can simulate different types of activity commonly performed by users and we then study how different malware samples react to such activities, we can then infer the typology of these malware samples.

In our approach, we first set up virtual machines to automatically perform activity that is typical of human users. We call the different types of activity *user triggers*. We then infect these virtual machines with malware and study how these samples react to each user trigger. Following the previous example, we expect that an information stealing malware sample will be triggered after the victim will log on a website and upload the user's credentials to a Command and Control (C&C) server. It will not present any particular behavior, however, if presented with other user activity, such as browsing on public websites. Our approach then applies Bayesian inference to assess *causality relations* between user triggers and malware samples. These relations can then be used to assess

the type of malware samples (e.g., if a certain sample belongs to the category of information stealer malware). Note that we are able to infer causality, and not simple correlation, because we are in full control of the user triggers that are provided (or not) to the malware sample.

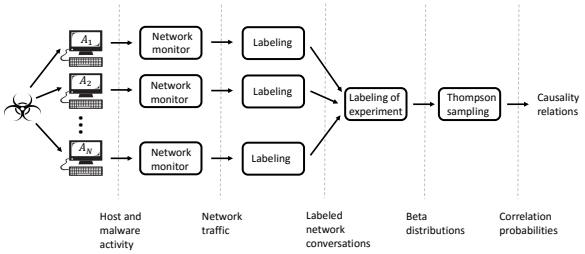

Figure 1: Overview of our approach.

We test our approach on multiple types of malware, namely information stealing malware, triggered by login events, adware, triggered by users navigating to webpages containing web advertisements, and spamming bots which perform their nefarious activity regardless of the activity performed by the victim. In summary, this paper makes the following contributions:

- We propose a methodology to assess causality relations between a user action and malware activity. We run malware samples in virtual machines performing user triggers, and infer the typology of the malware sample under analysis by studying how it reacts to the different triggers and using Bayesian inference.
- We present an instantiation of our approach in which we consider two types of user triggers (login events and website navigation) and three types of malware (information stealing, adware, and spambots).
- We evaluate our system showing that it is able to correctly infer causality between the trigger events and malware types.

2 Background: causality

This section will formalize the definition of causality that we use and how it is related to our work. The sources of this definition are in Lewis's work ([13]) where there is the definition of causality through "counterfactual analysis". The idea is to make a minimal modification to the variables set and observe if the outcome changes or not; if it happens, it is possible to determine whether there is a dependency relation between the changing variable and the outcome.

In our case, the outcome is the presence of new connections operated by the malware and the only thing that is changed is the trigger. If, among the used triggers, there is one trigger that makes the malware generate new connections, then in that test, there is a condition of causality between the trigger and the malware network operations. As explained in the next section, being in an experimental environment, there are issues related to the sensitivity of the system and to the noise due to unexpected network connections. These issues will be addressed through the use of Beta distributions for the Thompson sampling used in the Bayesian Inference tests.

Other causality models can be found in the literature. The most known is probably Pearl's causality model [14]; Although the importance of Pearl's work, we decided to rely on a simpler model, as the system did not require formalizations as complicated as Pearl's model.

3 Methodology

The goal of our approach is to infer the typology of a malware sample by learning causality relations between user actions (e.g., logging into a website) and the activity performed by the malware sample. To this end, we observe the network activity generated by infected Virtual Machines (VMs) and we apply statistical tests to assess causality.

An overview of our approach is displayed in Figure 1. We run a malware sample in a VM in which we execute a simulated user activity, called trigger. We then record network traffic generated by the VM and separate it between traffic that is relative to the user trigger (e.g., the traffic related to shopping websites), traffic that is generated by the malware sample before the trigger happens, and traffic that is generated by the malware sample after the trigger happens. We then extract the occurrences frequency of a certain activity related to a specific trigger, and perform Bayesian inference to determine correlation between this activity and the corresponding trigger. As we will explain, our Bayesian inference process involves extracting Beta distributions from the data and performing Thompson sampling to assess the causation probabilities. The decision of using a more complicated method such as Bayesian inference instead of a simpler chi-square test is because chi square only takes into account the proportion between quantities while Bayesian inference also considers the uncertainty in the measurements by using randomic sampling [20].

3.1 Setting up user triggers

In this section we explain and formalize the approach setup. Our approach takes into account a set of malware samples and a set of user triggers, and studies how each malware sample reacts to a specific user trigger. More formally, we define a set of malware samples $M_1, ..., M_i, ..., M_K$ and a set of possible trigger events $A_1, ..., A_j, ...A_N$. Each experiment runs a malware sample M_i in the presence of each trigger A_j, one at a time. This formalization is extremely scalable, in fact, the structure is still valid by increasing the number of malware types or the possible trigger events.

Instantiation of the experiments. We set up our experiment to take into account malware samples from three different types and study their relation to two user trigger events. The malware families that we study are *information stealer* malware (identified as *Inf* in the rest of the paper), *adware* (*Ad*), and *other* malware (*Ot*), where "other" includes malware samples that we typically do not expect to be triggered by user activity (e.g., spambots that send emails regardless of what the owner of the infected computer does). The trigger events that we used are the navigation to popular shopping websites (*Nav*) and the log in event into the Gmail webmail provider (*Log*). To correctly label network conversations at the next step, we also need to run an infected VM in which no user trigger is executed. We also identify network traffic generated by the operating system regardless of user activity and the malware sample infecting the machine; to this end, we run a not infected VM. We call this test *Idle*. The combinations of malware types M_i and user triggers A_j used in this paper are summarized in Table 1.

	Doing nothing	**Navigation**	**Logging in**
Not infected	Idle	Nav	Log
Info Stealer	Inf	InfNav	InfLog
Adware	Ad	AdNav	AdLog
Other	Ot	OtNav	OtLog

Table 1: Summary of our test cases.

Experimental environment. We set up a virtual environment in which different VMs are configured to run. The structure is similar to Botlab, created by John et al. [10].

A webserver manages the download of malware by the VMs and the additional content needed for the experiments. A mailserver is a sinkhole that receives all the SMTP packets the VMs generate. This design avoids our VMs from sending spam to the Internet. To allow the connectivity of the virtual network, a router implements rules of network address translation, SMTP packets redirection, and bandwidth restrictions, as described and suggested in [16]. To avoid the detection of the virtual environment by the malware samples, we used the Pafish tool [1] checks.

3.2 Extraction and labeling network conversations

The network dump files collected during our experiments contain information on the IP addresses contacted by the VMs during each of the tests. We define a *conversation* as the exchange of packets that have in common the tuple formed as source IP address, source port (TCP or UDP), destination IP address, and destination port. Conversations are then used for labeling. This way can be agnostic to the network payloads themselves.

This phase aims at assigning a "label" to the network conversations observed by a certain experiment. The goal is to identify the conversations that compose the user trigger first, and we can then label accordingly the malware activity that happens before and after the trigger.

For each test, we extract the list of network conversations, resolve the DNS domain associated with the destination IP address, and proceed with labeling them. More specifically, we assign four different labels to network conversations:

Common: operating systems such as Linux and Windows perform network traffic as part of their behavior, regardless of any user activity or program running on the machine. Examples of this include automated software updates and synchronization with network shares. To avoid considering this traffic as part of other labels, we run our VM without any malware sample or user trigger ("Idle" test in Table 1) and label any observed traffic as common, filtering it out when elaborating other labels.

Trigger: these conversations are those generated by the VM as part of a user trigger activity, for example the set of connections generated by visiting websites. We label conversations as Trigger if they are observed in the tests when the VM is not infected (marked as "Nav" and "Log" in Table 1) and were not marked as Common in the test "Idle."

Untriggered: these conversations are performed by a malware sample independently from the user trigger action. We use this label for conversations that are generated by the malware when no user trigger is present ("Inf," "Ad," and "Ot" tests in Table 1)

Triggered: these are the most important conversations for this work, because they are the ones that have the potential to present a correlation with the user trigger. We mark as Triggered any conversation that happens in a test in which a user trigger is happening, and that was not previously marked with any other label.

We first perform the "Idle" tests, followed by the tests in which only malware or user triggers are present, followed by the ones that combine a trigger and a malware sample. As we will explain later, the variability of the set of IP addresses and domains contacted as part of different trigger activities and by different malware samples forced us to re-run our tests multiple times. Table 2 reports the number of performed runs.

Table 3 shows which test assigned which label to a contacted domain. Apart from the test Idle, the tests without infection were giving a different trigger label to their contacted domains, while the domains contacted from tests without trigger are Untriggered ones, indi-

cating that the samples contact those domains independently from the machine action.

Test	Runs	Test	Runs	Test	Runs
Idle	42	Nav	108	Log	30
Inf	114	InfNav	40	InfLog	60
Ad	87	AdNav	73	AdLog	71
Ot	401	OtNav	159	OtLog	157

Table 2: Number of repetitions per test.

	Doing nothing	**Navigation**	**Logging in**
Not infected	Common	Navigation Trigger	Login Trigger
Info Stealer	Info Stealer Untriggered	Triggered	Triggered
Adware	Adware Untriggered	Triggered	Triggered
Other	Other Untriggered	Triggered	Triggered

Table 3: Labels encoding per each test.

Labeling settings. The labeling phase is the most delicate: we continuously performed an accurate tuning of the translation of IP addresses to the contacted domains because stealthy malware may be undetected if it uses the same domains as legitimate traffic or too many "Triggered" labels were assigned to contacted domains when the network identification was too fine grained.

As mentioned, we map IP addresses to domains when labeling network conversations. This works in most cases, because domains used by malware are not the ones used by legitimate applications. However, in some cases a domain can be used by both malware and legitimate traffic. One example of this is the use of Content Delivery Networks (CDNs). The biggest issue for our experiments were Amazon and Akamai servers: those address spaces are extremely wide and are used by a large variety of clients, from Amazon itself for advertisements on its website to malware samples hosting content to their domains. It is not possible nor to simply assign `amazonaws.com` a specific label, nor to assign one to the exact IP. Therefore we found a good balance in using the first two octets of the IP addresses and dividing in eight groups the third one, giving the corresponding label to each of these subnetworks.

Another problem occurred when a malware sample was contacting many IP addresses on the same network but not all of them: it happened that the sample contacted different IP addresses in different test runs. A similar issue is given by advertisements used by Amazon: it asks to several addresses the required information and every time a different address can be contacted. For this reason we ran some of the tests more times than others, increasing the labeling reliability.

3.3 Labeling of experiments

As we discussed, our approach assigns a label to each network conversation, whether it happens independently of a user trigger (untriggered), it is part of the trigger itself, or it happens as a consequence of the user trigger (triggered). We run each experiment as a combination of user trigger and malware sample, however, it is composed of multiple activities that generate a multitude of network conversations. To assess whether the malware running inside a certain VM as part of an experiment was triggered or not by a user action, we must "label" the whole experiment as triggered or not. For example, if we observe a new connection after the VM has logged into a website we can mark the experiment as "triggered." Otherwise, if no new activity is generated after the user trigger, we can mark it as "untriggered."

To label an entire experiment, we look at the labels assigned to the single conversations as explained in the previous sections. If any of the conversations is marked as "triggered" then we label the entire experiment as such. Otherwise we label the experiment as untriggered. A single sample is not sufficient to assess the relation between the malware sample and the trigger; for example, we might be mistakenly considering an experiment as triggered because the malware sample starts contacting a previously-unseen domain — however this can easily be the consequence of a change of C&C server (for example due to fast flux) rather than a reaction to the user trigger. For this reason we repeat each test several times and apply Bayesian inference to the set of results. Zand et al. [21] already applied statistical analysis and showed its validity in repeated tests for the study of causality in network traffic; they applied Chi-Square tests to assess causality between services and they repeated the tests varying some of the parameters to validate the test. In this work we applied Bayesian Inference on malware traffic instead of Chi Square tests on services on the network.

3.4 Statistical analysis

We use statistical analysis to assess whether there is a connection between what is a relation between the user activity performed by the VM and the network activity by the malware.

After the labeling procedure described in the previous section has been carried out, we have a set of frequencies at which different labels (i.e., triggered and untriggered) have occurred in the tests. The fraction of triggered and untriggered tests is then used to estimate the proportion parameter θ of a Binomial(θ) distribution based on the

sequence of binary observations where the observations are 1 in triggered experiments, and 0 in untriggered ones.

We estimate the proportion parameter using Bayesian inference to capture all uncertainty about its value. When performing Bayesian inference for the Binomial distribution, it is common to use the conjugate Beta(α, β) distribution as a prior. In this case, the posterior distribution is Beta($\alpha + N, \beta + M$) where N denotes the triggered label occurences during the test, and M denotes the occurences of the untriggered labelst [17]. The α and β parameters in the prior are chosen to take prior information into account, and we use the non-informative setting $\alpha = \beta = 0.5$

Once the posterior distribution has been obtained, we detect increases in the proportion parameter θ. This detection would make us understand whether there is a stronger relation between the trigger and the triggered communications among the different malware types; it can be done by integrating the joint posterior distribution over the relevant region of space. We use an approach based on Thompson sampling [19] for this purpose. We sample a random value from each of the Beta distributions and note which distribution produced the highest observed value. We repeat this procedure many times and divide the counts of the highest values by the number of repetitions. After the normalization we have a correlation probability of each test for the analyzed sequence and, as said in [23], because our environment is fully controlled, we can assess causality between the test with the highest probability and the sequence. In case of this strong relation it is possible to affirm that the malware samples that are part of a certain family are triggered by a certain action in the real world and operate different actions on the network because of the user trigger.

3.5 Limitations

The actions that can be detected by the presented system are a large variety and, because the system is content agnostic, this approach may also detect attacks through covert channels. The system is limited in detecting those samples that contact always the same C&C server during different phases of the attack: an Info Stealer sample that communicates the credentials to the same C&C server used in the first phase would not result as Triggered and can be misclassified in a detection system based on this work. At this stage, we cannot use unknown samples because we cannot infer causality through unknown samples; with the development of a detection system based on the causality inference it will be possible to use unknown malware samples.

4 Dataset

In Section 3.1 (Table 1) we illustrated which malware samples we used in this work. Although our approach can be used to assess relations between any user trigger and malware type, we decided to work with three malware types and two user triggers. More precisely, we ran 20 Zeus samples [3] as Info-stealers, 10 Shopper-pro and 3 CloudGuard samples as Adware [2], and 20 samples of other families. The use of a limited quantity of samples is due to different reasons, the most important being that we need active communication between the C&C server and the malware sample for our experiments. To collect the malware samples we periodically downloaded the most recent samples from VirusTotal.

As mentioned, we performed our tests multiple times (Table 2). The main reason for these repetitions was the need to establish strong statistical evidence to allow us assessing causality links between user actions and malware activity. Another reason for the runs is that both malware samples and legitimate services vary their servers frequently, especially by using fast flux and CDN. The Nav test had to be repeated multiple times because the advertisements pulled by Amazon change constantly, due to the use of CDNs, and we must label these connections correctly, to avoid noisy unreliable tests.

4.1 Behavior of malware samples by type

Before applying all the procedure described in Section 3, we manually analyzed the network traffic originated by the tests to understand the typical behavior of the different malware types. These are the characterizing traits:

Info stealers. These samples typically try to contact a certain number of C&C Servers to receive instructions about what to do in case of relevant data to steal (i.e., where to upload the stolen data). When relevant data is stolen, the malware communicates with different C&C Servers to upload the stolen data.

Adware. This type of malware operates a few connections to C&C servers to receive instructions about the hosts to contact when a website containing advertisements is visualized by the user. When the user navigates to a website containing advertisements, these are substituted by malicious ones. The sample's goal is immediately reached: the visualization of the malicious ads generates money to the malware operator.

Other. This group of malware samples were mainly Spambot samples. The Spambot samples are operating several different actions: they are contacting different C&C servers by using HTTP, HTTPS, and proprietary protocols; after these communication they start sending emails to victims by using the SMTP protocol. As we mentioned, our mailserver worked as a sinkhole for these emails. The large amount of communications caused mis-labeling and noise in the results.

5 Evaluation

In this section we evaluate our system. We present the labeling results on how many tests were triggered by which user triggers. We then describe how we extracted Beta

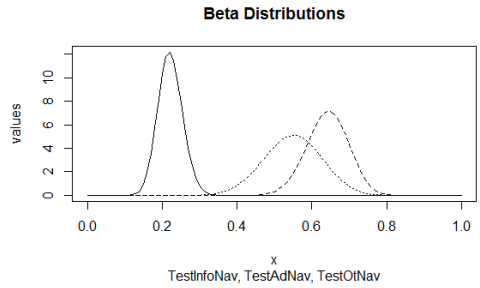

Figure 2: The Beta distributions related to the **Nav tests.

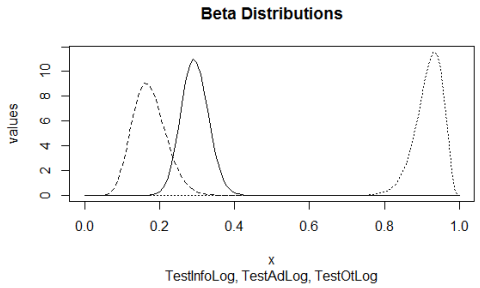

Figure 3: The Beta distributions related to the **Log tests.

distributions from the experiments and how we assessed causality, providing evidence on the validity of our work.

5.1 Labeling results

In Tables 4 and 5 we show what fraction of tests presented the "Triggered" or "Untriggered" label. Table 4 shows quite high values of triggered Adware samples (64.4%) while info stealers present a lower value (55%). These tests use the VM that navigates to shopping websites, loading the related advertisements, and runs the malware sample. Because of the adware modus operandi, we expect many triggered activities from the adware samples, rather then from the other malware types. Most of the adware samples are triggered by the navigation user trigger, however, a relevant number of info stealer samples seems triggered as well; these labeling errors are ruled out by the statistical tests. In other words, the statistical tests are able to determine that there is no causal link between a user navigating to a website and activity by information stealing malware. On the other hand, it is able to assess that adware is likely triggered by navigation.

Table 5 shows the result of the experiments for the tests in which the user trigger is a login event on the Gmail website. There is a high fraction of triggered Info stealers samples (92%), while only a small quantity of triggered Adware samples are triggered; the Other type reports 29% of its tests as "Triggered," but these triggers are ruled out by the statistical tests.

5.2 Beta distributions

To infer causality through the use of Bayesian inference, the first step is the creation of the Beta distributions from the results presented in the previous section. These results are used to draw the a posterior Beta distributions for each test as $(\beta(NumberOfTriggered + 0.5, NumberOfNotTriggered + 0.5)$. The Beta distributions that we used to model the variables are shown in Figure 2 and Figure 3. With the Test**Nav (all possible malware types, VM that is doing navigation) we observe a certain similarity between the curves in shape, height and position; these similarities are stronger observing only the Test AdNav and Test OtNav distributions. The distribution for the Test**Log tests show more differences between distributions; in fact the distributions are not close and the curve related to the InfLog tests (Info stealer) is much higher than the others. We can expect that the statistical tests will show a quite balanced situation between Tests**Nav while Tests**Log will give a preference in their results.

5.3 Statistical evaluation of causality and experimental validity

We ran Thompson sampling on the Beta distribution 200 times and calculated the average of the results over 10 repetitions; each result is a value between 0 and 1 that represents the probability of a causal relationship between a test and its label ("Triggered" or "Untriggered"). This probability takes into account the uncertainty given by mis-labeling due to the previously-explained issues therefore a big gap between the highest probability and the second one allows to assess causality.

In **Nav tests, Test AdNav is dominant. Test InfNav has 0.157 probability of being the cause of the triggered event among these tests while the probability of AdNav tests is 0.843. In OtNav tests, the triggered cases are not relevant. The mis-labeled OtNav tests did not affect the results of the statistical tests (probability equal to zero). The difference between the highest probability (the AdNav case) and the second one (InfNav) allows to indicate that the navigation user trigger caused the Adware network traffic. The statistical tests for **Log tests have a very clear outcome: the triggered info stealer actions are caused by the login to Gmail user trigger because the probability given by the statistical test is 1, while there is no relation between the login into gmail and the actions of the other malware types (the probability related to the other tests is zero).

Validity of the experiments. For this work we empirically decided the number of samples to use, how many times the tests were repeated and how many observations with random sampling were necessary. We evaluated if

Test	Triggered percentage	Untriggered percentage
InfNav	55%	45%
AdNav	64.4%	35.6%
OtNav	22%	78%

Table 4: Labels for the tests in which the VM is navigating to *amazon.com*.

Test	Triggered percentage	Untriggered percentage
InfLog	92.6%	7.4%
AdLog	16.9%	83.1%
OtLog	29.2%	70.8%

Table 5: Percentages of the different labels for the tests with Log VMs.

the number of repetitions operated for each test can be considered sufficient. We repeated the statistical test using different portions of the operated runs. When the **Nav tests were operated, at least 80% of the repetitions was needed for results to achieve enough confidence as with using the full set. On the other hand, **Log tests were derived from beta distributions extremely different, in fact the tests were giving reliable and stable results already with a small percentage of the runs.

Similarly to the procedure used for the test repetitions, we empirically validate the number of observations used during the Thompson sampling phase: starting from two observations, arriving to 200, we observed that more than 50 observations are needed to have stable results on **Nav tests, while even a minimum amount of observations is enough with **Log tests because the Beta distributions in this case are extremely different and indicate that InfLog tests have a clear correlation with the user trigger.

6 Discussion

In this section we will discuss the results of our framework in assessing the causality between a user-trigger and network activities performed by malware samples.

6.1 Labeling Results

As we already discussed, the labeling phase presented many challenges due to the several domains contacted by some malware samples and by the VM navigating to shopping websites. Despite our best attempts, whitelisting websites confirmed to be benign and the granularity tuning cannot completely remove mislabeling mistakes when large server domains (e.g. `amazonaws`) were contacted. For example, tests with Spambots are considered triggered because of two advertisements domains contacted only during OtNav and OtLog tests.

6.2 Results and Validity

We ran statistical tests to assess the relation between the user trigger and one of the tests. We expected a strong relation between Adware triggered traffic and the navigation trigger (Test AdNav) and between Info Stealer triggered tests and the login trigger (Test InfLog). Both relations were clearly assessed, even if some mislabeling affected the navigation case. To better understand if this noise influenced the experiments we observed that Test OtNav and Test OtLog do not present different results (Tables 4 and 5), while the tables show different behaviors with Test InfNav and Test InfLog or with Test AdNav and Test AdLog. These differences indicate that the malware samples are influenced by the user trigger events; in the case of the login, the significance of the result was not affected by the noise when we applied Bayesian Inference while the noise has been more effective in the navigation case, although our statistical analysis was still able to rule out these mislabeling.

The validity paragraphs (Section 5.3) show that the experiments are not biased by an incomplete dataset or a non sufficient number of observations. Because the validity criteria is respected we can argue that both the statistical tests indicated a causal dependency between the navigation user trigger and adware as well as between the login trigger and information stealing malware.

In its current form, the framework does not take into account information on *when* a network flow appears within a test, but only if it appears or not. Mislabeling could be reduced in the future by using this information.

This work to be improved into a fully-fledged detection system. Malware samples could be run against different user triggers and an alarm could be raised when a type of malware that is considered of particular risk will be generated (e.g., an information stealing malware sample that has the capability of damaging to the company).

7 Related Work

Causality is a delicate topic: although we can use tests to infer correlation, we cannot infer causality because it does not only establishes the presence of a relation between two variables, but its direction. According to [23] "correlation does not necessarily indicate causality and ideally, a controlled experiment would allow firm causal inference". As the tests and the environment (Section 3.1) are fully controlled, we can assess causality where the statistical tests are giving evidence of correlation between our incoming variable (the operated test) and the outcome (the conversations labels).

Correlation and causality topics are widely explored in some fields of computer actions and network communications like services dependency, the most recent papers on this topic are Orion [7] and Rippler [21].

However, the topic we tackle is not the causality between services but between events and malware activ-

ity. Another important point the previous work analyzed is the malware samples similarity. The number of malware samples is dramatically increasing every day, but new samples are often using part of the code of previous ones and, as consequence, the behavior is similar too. Chakradeo et al. [6] analyzed the phenomenon on mobile malware while Bitshred [9] speeds up malware identification by taking advantage of malware triaging. Malware triaging is important with respect to our work because of the similarities between samples' network behavior (shown in Sigmal [11]); moreover, some samples need a trigger event to do certain operations, a problem tackled by Brumley et al. [5] by automatically analyzing the binaries, while we are looking to the network level of the phenomenon. These researches analyzed the binaries similarities, but not the network operations caused by network events.

A more recent work [22] is more similar to ours because it looks to malware communications, but limiting itself to HTTP and DNS events to detect malicious triggered events; they assessed causality because of the detection of the malicious events, but without any test as evidence. Moreover the mentioned work used its own proof-of-concept malicious samples that were written by the researchers to act in a predefined way while a real sample may act differently every time depending on many factors that can affect its reactions to trigger events.

8 Conclusions

In this paper we assessed the causality relation between user actions and malware activities on the network. User actions were identified as the triggers of malware network traffic even in case of strong noise affecting the statistical tests. The causal relation between a specific trigger event and a certain type of malware can bring to the identification of the type of malware that is infecting the network. The malware type identification may lead to prioritization choices when a network administrator has many alarms to manage and has to understand which ones are the highest network threat.

9 Acknowledgments

We wish to thank the anonymous reviewers for their comments. This work was funded by the EPSRC grant number EP/N008448/1. Enrico Mariconti was funded by the EPSRC under grant 1490017, while Jeremiah Onaolapo was supported by the Petroleum Technology Development Fund (PTDF), Nigeria.

References

[1] Alberto Orgega. Pafish (Paranoid Fish). https://github.com/a0rtega/pafish.

[2] Aycock, J. *Spyware and Adware*. Springer Science & Business Media, 2010.

[3] Binsalleeh, H., Ormerod, T., Boukhtouta, A., Sinha, P., Youssef, A., Debbabi, M., and Wang, L. On the analysis of the zeus botnet crimeware toolkit. In *International Conference on Privacy Security and Trust* (2010).

[4] Bradley, Tony. Report: Average of 82,000 new malware threats per day in 2013. http://www.pcworld.com/article/2109210/report-average-of-82-000-new-malware-threats-per-day-in-2013.html, 2014.

[5] Brumley, D., Hartwig, C., Liang, Z., Newsome, J., Song, D., and Yin, H. Automatically identifying trigger-based behavior in malware. In *Botnet Detection* (2008).

[6] Chakradeo, S., Reaves, B., Traynor, P., and Enck, W. Mast: triage for market-scale mobile malware analysis. In *ACM conference on Security and privacy in wireless and mobile networks* (2013).

[7] Chen, X., Zhang, M., Mao, Z. M., and Bahl, P. Automating network application dependency discovery: experiences, limitations, and new solutions. In *OSDI* (2008).

[8] Christodorescu, M., Jha, S., Kinder, J., Katzenbeisser, S., and Veith, H. Software transformations to improve malware detection. *Journal in Computer Virology* (2007).

[9] Jang, J., Brumley, D., and Venkataraman, S. Bitshred: feature hashing malware for scalable triage and semantic analysis. In *ACM Conference on Computer and Communications Security (CCS)* (2011).

[10] John, J. P., Moshchuk, A., Gribble, S. D., and Krishnamurthy, A. Studying Spamming Botnets Using Botlab. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (2009).

[11] Kirat, D., Nataraj, L., Vigna, G., and Manjunath, B. S. Sigmal: A static signal processing based malware triage. In *Annual Computer Security Applications Conference (ACSAC)* (2013).

[12] Langner, Ralph. To Kill a Centrifuge. http://www.langner.com/en/wp-content/uploads/2013/11/To-kill-a-centrifuge.pdf, 2013.

[13] Lewis, D. Counterfactuals and comparative possibility. *Journal of Philosophical Logic 2* (1973), 2161–2173.

[14] Pearl, J. *Causality*. Cambridge university press, 2009.

[15] Rad, B. B., Masrom, M., and Ibrahim, S. Camouflage in malware: from encryption to metamorphism. *International Journal of Computer Science and Network Security* (2012).

[16] Rossow, C., Dietrich, C. J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., Bos, H., and van Steen, M. Prudent practices for designing malware experiments: Status quo and outlook. In *IEEE Symposium on Security and Privacy* (2012).

[17] Scott, S. L. A modern Bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry 26* (2010), 22–35.

[18] Stone-Gross, B., Holz, T., Stringhini, G., and Vigna, G. 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).

[19] Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika 25* (1933), 285–294.

[20] Trotta, R. Bayes in the sky: Bayesian inference and model selection in cosmology. *Contemporary Physics 49* (2008), 71–104.

[21] Zand, A., Vigna, G., Kemmerer, R., and Kruegel, C. Rippler: Delay Injection for Service Dependency Detection. In *IEEE Conference on Computer Communications (INFOCOM)* (2014).

[22] Zhang, H., Yao, D., and Ramakrishnan, N. Detection of stealthy malware activities with traffic causality and scalable triggering relation discovery. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)* (2014).

[23] Zhang, J., Durumeric, Z., Bailey, M., Liu, M., and Karir, M. On the Mismanagement and Maliciousness of Networks. In *Symposium on Network and Distributed System Security (NDSS)* (2014).