# A Methodology To Assess Malware Causality In Network Activities

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

University College London
e.mariconti@cs.ucl.ac.uk, j.onaolapo@cs.ucl.ac.uk,
gordon@gordonjross.co.uk, g.stringhini@ucl.ac.uk

**Abstract.** The current malware analysis methods cannot stand the pace the creation of new malware samples has. When analyzing an unknown malware sample, it is important to determine its capabilities of damaging its victims. In a company, for example, a malware infection from an information stealer sample is much more critical than one from a spambot sample, and have to be dealt with the highest priority. In this paper, we present a methodology and some initial results about learning the typology of a malware sample by presenting it with a number of user trigger actions, and studying if the sample reacts to the events. We present a statistical approach able to determine causality relations between a specific trigger action (e.g., a user visiting a certain website) and a malware sample. The initial results show that our approach can correctly infer the causality relations between malware types and trigger events.

## 1 Introduction

The malware problem is an increasing threat as time passes: [1] measured that in 2013 attackers released a new malware sample in the wild every second. It is possible to create malware to attack critical infrastructures of a country [7] as well as samples that is sending unwanted content to Internet users, such as email spam [6, 11]. It is clear that the huge amounts of malware threats that are observed every day requires an effective risk assessment, in which the potential damage that each piece of malware can cause to companies and their customers needs to be efficiently determined.

The first step to tackle malware threats is to analyze the samples to understand their capabilities and the potential damage that they can cause to the victim network. It is particularly important to figure out the purpose of the malware infections. Knowing the infection type allows focusing mitigation efforts, for instance by prioritizing the cleanup of an infection over an other. *triaging* is how we refer to this prioritization process. In the past, malware triaging approaches used binary analysis to choose the strategy [2, 4]. When cybercriminals heavily harden their code [3, 8] these approaches are less effective.

In this paper we tackle malware triaging from a different perspective. The rationale behind the work is the following: many categories of malware are activated from the victim actions. By running malware samples in the presence of different types of simulated user actions (triggers), we can assess which user activity is triggering the malware sample, and detect the type of that malware sample. For example, we expect an information stealing malware sample to be triggered when the victim will input login credentials on a website and the sample will upload this data to a C&C server. However, it will not react to other types of user activity, such as browsing on public websites. We propose a framework to study the malware sample network behavior in the presence of different user triggers and inferring which triggers activate the malware. Based on these observations, we can infer the typology of this malware sample. Since we control the user triggers provided to the malware sample, we can infer **causality** between the user triggers and the malware activity.

In summary, this paper proposes a methodology to assess causality relations between a user action and malware activity. Our statistical approach is based on Bayesian inference.

## 2 Methodology

We want to infer the type of a malware sample by learning causality relations between user actions and the activity performed by the malware sample. For this reason, we observe the network packets generated by infected Virtual Machines (VMs) and apply statistical tests to assess causality. In this section, we describe our approach in detail.

Each test regarding a trigger and a malware sample follows this procedure: we record the test's network activity and extract the conversations from the dump file to label them depending on what generated them. From the conversation labels we create a chain of labels every time; we repeat the test to apply Bayesian inference on the chains frequencies of labels assessing if there are relations between labels and tests.

### 2.1 Formalization of our approach

In a nutshell, our approach takes into account a set of malicious samples and a set of triggers, and studies if the samples react to the user triggers. More formally, we define a set of malware samples $M_1, ..., M_i, ..., M_K$ and a set of possible trigger events $N_1, ..., N_j, ...N_L$. An experiment consists in running one of the samples $M_i$ in the presence of each trigger $N_j$, one trigger at a time. This formalization is extremely scalable, in fact, the approach is valid when changing how many malware families or possible triggers we use.

### 2.2 Experimental environment

We set up a virtual environment in which different VMs are configured to run. The structure is similar to the one created by John et al. [5].

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, the router redirects those packets. 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, redirection of SMTP packets, and bandwidth restrictions, that will mitigate denial of service attacks performed against public servers from the VMs. These are only some of the security measures that we applied by following the guidelines from [9]. To avoid the detection of the virtual environment by the malware samples, we followed the suggestions of the Pafish tool.

## 2.3 Extraction and labeling network conversations

The network dump files collected during our experiments recorded information on the tests. Packets that have in common the tuple (sourceip, sourceport, destinationip, destinationport) are a conversation. The conversations is the key for labeling: we do not need to analyze each packet to identify the malware actions but we still have all the needed details.

For each test, we extract the conversations and then we label them: we determine the protocol and which domain is contacted by the VM for each conversation. Table 1 shows the label assigned to the contacted IP addresses depending on the test we are taking into account. For example, domains that are always contacted by the VM that runs trigger event 1, regardless of the specific malware sample that is being tested, are labeled as "event1 Trigger". These conversations are independent from malware traffic and we will filter them out when we will be looking for traffic generated by malware as a possible response to a user trigger.

We define four possible conversations:

1. Common: those are the conversations the VM performs independently from malware samples or user trigger actions.
2. Pre-Trigger: those communications are performed by a malware sample independently from the user trigger action performed by the VM. The word pre indicates the independence from events happening after a specific user trigger is issued.
3. Trigger: those conversations are part of a user trigger activity, for example the connections generated by visiting a website.
4. Triggered: those conversations that did not appear in any of the previous labels. In this case, we consider the malware sample operating a connection as a consequence to a user trigger event.

We first have to learn which connections belong to the "Common" label: this is done by test Idle, no malware and no trigger. Those conversations are the ones the operative system has to operate to establish the communications and monitor which machine is in the same network as the one of the records. Those contacted domains and IP addresses will always be contacted in every test and will not be important for the analysis; for this reason the label is used only as a filter.

The "Trigger" ones are the labels assigned to the domains contacted by the VMs when no malware is infecting them: the domains that are not excluded by the "Common" label are those related to the trigger event. Using the same logic for "Pre-Trigger", we can label the domains contacted by the malware samples when the VM does not operate any particular action.

The “Triggered” label is given to the domains that are contacted during tests where there are a trigger operation and a malware: this label is given to the domains that are not already in the previous labels. The label “Triggered” will be given to those domains (if there is any) that are not already in “Trigger” and in “Pre-Trigger” labels.

| | **Doing nothing** | **Trigger 1** | ... | **Trigger L** |
|---|---|---|---|---|
| **Not infected** | Common | Trigger | ... | Trigger |
| **Malware type 1** | Pre-Trigger | Triggered | ... | Triggered |
| **...** | ... | ... | ... | ... |
| **Other** | Pre-Trigger | Triggered | ... | Triggered |

Table 1: Encoding of the labels. Domains contacted during tests are labeled following this table. Running VMs without any malware infecting them allows to find the conversations labeled as “Trigger”, while running an infected VM in idle is how we assign to the conversations the label“Pre-Trigger”. When the label to be assigned is “Triggered”, it can be assigned only if that domain is not already in previous ones.

The labeling phase is the most delicate of this work: we continuously performed an accurate tuning of the translation of the IP addresses to the contacted domains because stealthy malware may be unobserved if they were using the same domains as legitimate traffic or too many “Triggered” labels were assigned to contacted domains when the identification of the network is too precise. This is due to the presence of large IP spaces and the use of Content Delivery Networks.

## 2.4 Chains of events

In Figure 1 we explained which test was assigning which label to a certain 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 labeled with a pre-trigger label indicating that the samples contact those domains independently from the machine actions.

The tests where a malware sample runs in a VM that operates a trigger event label the domains that are not part of pre-trigger and trigger lists as triggered. This means that each of these tests may have different labels every time they are repeated, depending also on which samples and the current settings of the websites that are visited. The main concept of the work is in this point: every repetition of the test will create a chain of labels given the events (i.e. the connections) that the host machine will record. Each repetition will have a correspondent labels chain, therefore every test will have a number of times where a certain chain is the result while another one related to another chain.

Every test where there is an infection and an action by the VM can have two possible outcomes: “PreTrigger-Trigger-Triggered” in case of some new actions generated by the malware sample and “PreTrigger-Trigger” in case of a sample that does not react to the trigger. In the next section we explain how to study the correlation between the test type and the resulting sequences and why, in case of correlation, we can assess causality.

### 2.5 Statistical analysis

We use statistical analysis to assess whether there is a connection between what happens in the VM and what is observed on the network and, as consequence, if there is a connection between the VM actions and the malware ones.

After the above labeling procedure has been carried out, the results consist of the frequencies at which the chains of labels have occurred in the tests. For each chain, our goal is to estimate the proportion of times it occurs during the test. This is essentially the task of estimating the proportion parameter $\theta$ of a Binomial($\theta$) distribution based on a sequence of binary observations where the observations are 1 if the sequence occurred, and 0 if it did not occur.

We estimate the proportion parameter using Bayesian inference to allow all uncertainty about its value to be captured. When performing Bayesian inference for the Binomial distribution, it is usual to use the conjugate Beta($\alpha, \beta$) distribution as a prior. In this case, the posterior distribution is Beta($\alpha + N, \beta + M$) where $N$ denotes the number of times the analysed sequence occurs during the test, and M denotes the number of other sequences that occurred during repetitions of the test [10]. The $\alpha$ and $\beta$ 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, it is possible to detect increases in the proportion parameter $\theta$. This can be done by integrating the joint posterior distribution over the relevant region of space. We use an approach based on Thompson sampling [12] 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 reputations. After the normalization we have a probability of correlation of each test for the sequence analyzed and, as said in [13], because our environment is fully controlled and managed, it is possible to 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 trigger.

## 3 Conclusions and future work

In this paper we presented a methodology to assess causality between user actions and malware activities on the network. The causal relation identifying a specific trigger event as root of a certain type of malware activity can bring to the identification of the type of malware that is infecting the network. This identification may lead to prioritization choices when a network administrator has to manage the interventions due to many security alarms.

This work shows the opportunity to relate as cause and effect an action typically made by a user and the actions of a malware. By following different paths, future work can lead to an effective defensive tool:

– The use of a high number of types of malware and of user triggers to have a complete spectrum of the causality relations.
– The implementation of a detection system that uses the techniques explained in this work. The different behavior with or without the possible trigger events would allow the identification of the malware sample.

## 4 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. 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.
2. Brumley, D., Hartwig, C., Liang, Z., Newsome, J., Song, D., and Yin, H. Automatically identifying trigger-based behavior in malware. In *Botnet Detection* (2008).
3. Christodorescu, M., Jha, S., Kinder, J., Katzenbeisser, S., and Veith, H. Software transformations to improve malware detection. *Journal in Computer Virology* (2007).
4. 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).
5. 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).
6. Kreibich, C., Kanich, C., Levchenko, K., Enright, B., Voelker, G. M., Paxson, V., and Savage, S. Spamcraft: An inside look at spam campaign orchestration. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)* (2009).
7. Langner, Ralph. To Kill a Centrifuge. http://www.langner.com/en/wp-content/uploads/2013/11/To-kill-a-centrifuge.pdf, 2013.
8. Rad, B. B., Masrom, M., and Ibrahim, S. Camouflage in malware: from encryption to metamorphism. *International Journal of Computer Science and Network Security* (2012).
9. 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).
10. Scott, S. L. A modern Bayesian look at the multiarmed bandit. *Applied Stochastic Models in Business and Industry 26* (2010), 22–35.
11. 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).
12. 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.
13. 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).