HOnions: Detection and Identification of Snooping Tor HSDirs

Amirali Sanatinia, Guevara Noubir
College of Computer and Information Science
Northeastern University, Boston, USA
{amirali,noubir}@ccs.neu.edu

***Abstract*—In the last decade, Tor proved to be a very successful and widely popular system to protect users' anonymity. However, Tor remains a practical system with a variety of limitations, some of which were indeed exploited in the recent past [1]. Tor's security relies on the fact that a substantial number of its nodes do not misbehave. In this work, we introduce, the concept of honey onions, a framework to detect misbehaving Tor relays with HSDir capability. This allows to obtain lower bounds on misbehavior among relays. We propose algorithms to both estimate the number of snooping HSDirs and identify the most likely snoopers. Our experimental results indicate that during the period of the study (72 days) at least 110 such nodes were snooping information about hidden services they host. We reveal that more than half of them were hosted on cloud infrastructure and delayed the use of the learned information to prevent easy traceback. The preliminary results of this work have been presented [2].**

I. HOnion Generation & Detection

We introduce the concept of *honey onions* (`honions`), to expose when a Tor relay with HSDir capability has been modified to snoop into the hidden services that it currently hosts. In order to automate the process of generating and deploying honions in a way that they cover a significant fraction of HSDirs, we developed several tools. A key constraint in this process was to minimize the number of deployed honions. This derives primarily from our desire to not impact the Tor statistics about hidden services (specially given the recent surge anomaly). By considering the number of HSDirs (approximately 3000), we could infer that we need to generate around 1500 honions to cover all HSDirs with 0.95 probability. We used 1500 honions per batch (daily, weekly, or monthly) and could verify that 95% of the HSDirs were systematically covered.

HOnion back end servers: Each honion corresponds to a process that is running locally. The server behind hidden services, should not be running on a public IP address, to avoid de-anonymization. We also log all the requests that are made to the server programs and the time of each visit. Recording the content of the requests allows us to investigate the snoopers' behavior and intent.

HOnions generation and deployment schedule: To keep the total number of honions small, we decided on three schedules for their generation and placement, *daily*, *weekly*, and *monthly*. The three schedules allow us to detect the malicious HSDirs who visit the honions shortly (less than 24 hours) after hosting them. Since the HSDirs for hidden services change periodically, more sophisticated snoopers may wait for a longer duration of time, so they can evade detection and frame other HSDirs.

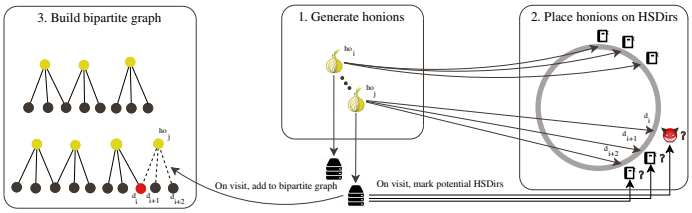
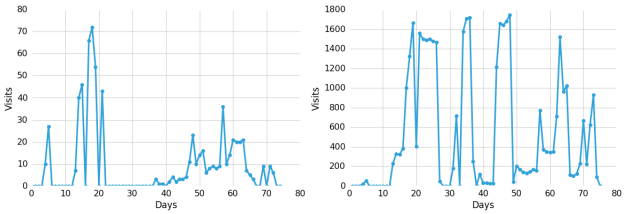


Fig. 1: Flow diagram of the honion system.

Identifying snooping HSDirs: Based on the visited hidden service, the time of the visit, and the HSDir that have been hosting the specific onion address prior to the visit, we can mark the potential malicious and misbehaving HSDirs. Then, we add the candidates to a bipartite graph, which consists of edges between HSDirs and the visited honions. The analysis of this graph allows us to infer a lower bound on the number of malicious HSDirs as well as specific snoopers. Figure 1 depicts the architecture of the system.

HOnion Visit Graph Formation: In the following we first introduce a formal model and notation for the Honey Onions system. First, HO denotes the set of honey onions generated by the system that were visited, and HSD the set of Tor relays with the `HSDir` flag (so far referred to as HSDir relays). The visits of honions allow us to build a graph $G = (V, E)$ whose vertices are the union of HO and HSD and edges connect a honion ho_j and HSDir d_i iff ho_j was placed on d_i and subsequently experienced a visit. G is by construction a bipartite graph. We also note that each honion periodically changes descriptors and therefore HSDirs (approximately once a day). However, a HSDir currently a honion ho cannot explain visits during past days. Therefore, each time a honion changes HSDirs we clone its vertex ho to ho' and only add edges between ho' and the HSDirs who know about its existence when the visit happened.

Estimation & Set Cover: Since each honion is simultaneously placed on multiple HSDirs, the problem of identifying which ones are malicious is not trivial. We first formulate the problem of deriving a lower-bound on their number by finding the smallest subset S of HSD that can explain all the visits. The size s of the minimal set tells us that there cannot be less than s malicious HSDirs who would explain the visits.



(a) Daily Visits (b) All Visits

Fig. 2: Plot of the visits to the honions.

Fig. 3: The global map of detected misbehaving HSDirs and their geographic origin.

$$\underset{S \subseteq HSD}{\operatorname{argmin}} |S : \forall (ho_j, d_i) \in E \exists d_i' \in S \wedge (ho_j, d_i') \in E| \quad (1)$$

Finding the smallest set S as defined by Equation 1, is not trivial as one can easily see that it is equivalent to the hitting set problem, which itself is equivalent to the set cover problem, which is well known to be NP-Complete. However, it can also be formulated as an Integer Linear Program. Let $x_{1 \leq j \leq |HSD|}$ be binary variables taking values 0 or 1. Solving Equation 1, consists of finding integer assignments to the x_j such that:

$$\begin{array}{ll} \min_{(x_1,\ldots,x_{HSD})} & \sum_{j=1}^{|HSD|} x_j \\ \text{subject to } \forall ho_i \in HO & \sum_{\forall j:(ho_i,d_j)\in E} x_j \geq 1 \end{array}$$

II. Results & Discussion

We started the daily honions on Feb 12, 2016; the weekly and monthly experiments on February 21, 2016, which lasted until April 24, 2016. During this period there were three spikes in the number of hidden services, with one spike more than tripling the average number of hidden services. There are some theories suggesting that this was due to botnets, ransomware, or the success of the anonymous chat service, called Ricochet. However, none of these explanations can definitely justify the current number of hidden services. Our daily honions spotted snooping behavior before the spike in the hidden services, this gives us a level of confidence that the snoopings are not only a result of the anomaly (Figure 2). Rather, there are entities that actively investigate hidden services.

Snooping HSDirs Nature and Location: In total we detected at least 110 malicious HSDir using the ILP algorithm, and about 40000 visits. More than 70% of these HSDirs are hosted on Cloud infrastructure. Around 25% are exit nodes as compared to the average, 15% of all relays in 2016, that have both the HSDir and the Exit flags. This can be interesting for further investigation, since it is known that some Exit nodes are malicious and actively interfere with users' traffic and perform active MITM attacks [3]. Furthermore, 20% of the misbehaving HSDirs are, both exit nodes and are hosted on Cloud systems, hosted in Europe and Northern America. The top 5 countries are, USA, Germany, France, UK, and Netherlands. Figure 3 depicts the spread and the geolocation of the malicious HSDirs.

HSDirs Behavior and Intensity of the Visits: Most of the visits were just querying the root path of the server and were automated. However, we identified less than 20 possible manual probing, because of a query for favicon.ico, the little icon that is shown in the browser, which the Tor browser requests. Some snoopers kept probing for more information even when we returned an empty page. For example, we had queries for `description.json`, which is a proposal to all HTTP servers inside Tor network to allow hidden services search engines such as Ahmia, to index websites. One of the snooping HSDirs (5.*.*.*:9011) was actively querying the server every 1 hour asking for a server-status page of Apache. It is part of the functionality provided by mod_status in Apache, which provides information on server activity and performance. Additionally, we detected other attack vectors, such as SQL injection, targeting `the information_schema.tables`, username enumeration in Drupal, cross-site scripting (XSS), path traversal (looking for `boot.ini` and `/etc/passwd`), targeting Ruby on Rails framework (rails/info/properties), and PHP Easter Eggs (?=PHP*-*-*-*-*).

III. Conclusion & Future Work

In this work, we introduced honey onions (HOnions), a framework for methodically estimating and identifying Tor HSDir nodes that are snooping on hidden services they are hosting. We propose algorithms to both estimate the number of snooping HSDirs and identify them. Our experimental results indicate that during the period of the study (72 days) at least 110 such nodes were snooping information about hidden services they host. Based on our observations not all snooping HSDirs operate with the same level of sophistication. For example, some do not visit the hosted honions immediately to avoid detection by daily honions, our weekly and monthly honions can detect them. We believe that behavior of the snoopers can be modeled and studied in more detail. Furthermore, we reveal that more than half of them were hosted on cloud infrastructure making it difficult to detect malicious Tor nodes. Furthermore, cloud providers such as Vultr, even accepts payments in the form of bitcoins, which prevents the traceback and identification of misbehaving entities.

References

[1] A. Sanatinia and G. Noubir. Onionbots: Subverting privacy infrastructure for cyber attacks. In *DSN*, 2015.

[2] A. Sanatinia and G. Noubir. Honey onions: a framework for characterizing and identifying misbehaving tor hsdirs. In *IEEE CNS*, 2016.

[3] P. Winter, R. Köwer, M. Mulazzani, M. Huber, S. Schrittwieser, S. Lindskog, and E. Weippl. Spoiled onions: Exposing malicious tor exit relays. In *PETs*, 2014.