IPFS and IPNS protocols as a means of botnet control: proof of concept

Bruno Macabeus M. de Aquino1**, Marcus Vinicius L. de** Lima1**, Joa˜o Paolo Cavalcante M. de** Oliveira1**, Cidcley Teixeira de** Souza1

[1]Federal Institute of Education, Science and Technology of Ceara' (IFCE)
Fortaleza, CE - Brazil

bruno.macabeus@gmail.com, marcsvll@acad.ifce.edu.br,
{paolo.oliveira,*cidcley}@ifce*.edu.br

Abstract: *To make the internet safer, a fundamental step is to avoid the usage of a remotely-controlled infected computer network (Botnet) by a malicious user (Botmaster) who may use it for, among other purposes, DDoS attacks. To fight against this problem, a challenge is the evolution of command and control services (C&C) used by the Botmaster for Botnet management, which are increasingly more sophisticated and hard to detect. A natural evolution for C&C is the usage of new distributed computing protocols. This paper outlines a new proof of concept C&C using two of these protocols, IPFS and IPNS, which would allow the Botmaster to acquire safer and more anonymous communication.*

Summary. *To make the Internet more secure, a fundamental step is to combat the use of a network of infected and remotely controlled computers (Botnet) by a malicious user (Botmaster), who can use it for, among other purposes, DDoS attacks. To combat this problem, one challenge is the evolution of the command and control (C&C) services used by the Botmaster to manage his Botnet, which are increasingly sophisticated and difficult to detect. A natural evolution in C&C would be the use of new distributed computing protocols. This paper presents a proof-of-concept of C&C with two of these protocols, IPFS and IPNS, which would enable the Botmaster to achieve a more secure and anonymous communication.*

1. Introduction

Malware exists whose purpose is to make the victims' computers remotely controllable by the attacker. A network of computers controlled in this way is commonly known as a *botnet*, with each infected machine being called a *bot*, and the controller being called the "*bot*". of this network called the *botmaster*. The network is accessible to the *botmaster* by means of
a communication service known as *Command-and-Control* (C&C). Using its *botnet*, the *botmaster* acquires a large computational power, which it can use to, for example, perform a *Distributed Denial-of-Service* (DDoS) attack [Upadhyaya et al. 2011].

The C&C is the fundamental part of a *botnet*. It needs to ensure the anonymity of the *botmaster* and be a secure means of communication to prevent the *malware from* being easily detected, as well as prevent *Sybil* attacks [Wang et al. 2009]. Several protocols, such as IRC, HTTP and DNS [Dietrich et al. 2011], as well as different topologies, such as centralized and distributed, are used to develop different C&C [Bailey et al. 2009].

In this research a *botnet* whose C&C uses two recently proposed protocols: *InterPlanetary File System* (IPFS) and *InterPla- netary NameSpace (*IPNS), was developed as a proof of concept, verifying their viability as a C&C channel.

This paper is organized as follows. Section 2.1 provides a technical discussion of C&C. Section 2.2 presents the IPFS and IPNS protocols. Sections 3, 3.1, and
3.2 describe and discuss the results obtained in the implementation of a *bot* using these protocols in its C&C. Finally, section 4 concludes and describes future work.

2. Theoretical Background

This section explains the concepts related to *Command-and-Control* (C&C) and the *InterPlanetary File System (*IPFS) and *InterPlanetary NameSpace* (IPNS) protocols.

2.1. Command-and-Control

Several C&C models have been proposed and are used in *bots*. The oldest is the centralized topology model, in which a single point is responsible for the exchange of messages between the *bots* and the *botmaster*. The major advantages of the centralized model are low latency in communication and simplicity of design. However, there is the disadvantage of the C&C server being a critical point, since all communication is performed through it. If discovered, the whole system can be compromised. Furthermore, monitoring by third parties, such as researchers and security agents, becomes easier [Zeidanloo and Manaf 2009, Bailey et al. 2009].

Due to this fragility of the centralized model, *botnets* with distributed topology, also called *Peer-to-Peer* (P2P), began to be developed [Zeidanloo and Manaf 2009]. With this model, the *botmaster* can enjoy a higher degree of anonymity [Upadhyaya et al. 2011] and provide resilience to the *botnet*, since there is not only a single point of failure [Zeidanloo and Manaf 2009].

One of the ways to detect a *bot* is by analyzing the traffic of its C&C. Because of this, attackers are always looking for new protocols and ways to communicate with the C&C, to evade security agents [Bailey et al. 2009]. Thus, it is relevant to study new techniques for the development of C&C, in order to understand and measure future risks.

2.2. ***InterPlanetary File System* (IPFS) and *InterPlanetary NameSpace* protocols (IPNS)**

IPFS is a *peer-to-peer* protocol for distributed file systems [Benet 2014]. The simplest way to access IPFS is through the public *gateway*[1] . It is also possible to run a *daemon* locally or host your own *gateway* to gain access to IPFS.

On the IPFS network one can add data, such as files, which are called *objects*. Each *object is* referenced by a *hash* calculated from

1https://gitbook.com/book/flyingzumwalt/decentralized-web-primer its [illegible]Every *object* contained in IPFS is immutable, so if it is necessary to add a new version of a given file, a new *object* must be created, totally independent from the previous one. In this way, a new *hash is* obtained for the new version of the file, but the previous one is also accessible.

This concept is called *content-addressing* and aims to ensure greater integrity, regardless of where or who is offering the [illegible]due to this concept, the *links* are permanent, always pointing to the same *object* [Benet 2014]. H[illegible] this approach becomes inconvenient if the data is frequently updated, since it would be [illegible]to obtain the *hash of the* latest version of the *object* from another channel. To solve this problem, the IPNS protocol was developed, which allows to provide a redirection to a given *object* from a *peerID* (*hash of a* [illegible] public key) [Benet 2014]. In this way, a unique *hash* can be referenced that always points to the *hash of* the newest *object.*

3. Proof of Concept

As a proof of concept, we developed[2] a *bot* that uses the IPFS and IPNS protocols as C&C. Among the different ways to access the files contained in the IPFS, we chose to use the [illegible]*ateway*, so as not to need to instantiate in the *malware* itself a *daemon* to gain access to the IPFS, nor to execute a public *gateway* in another server. The public *gateway* has some limitations, such as not being able to add *objects* from the *bot*. However, it is sufficient for the proof of concept, since it allows receiving data - in this case, the commands issued by the *botmaster* to his *botnet*.

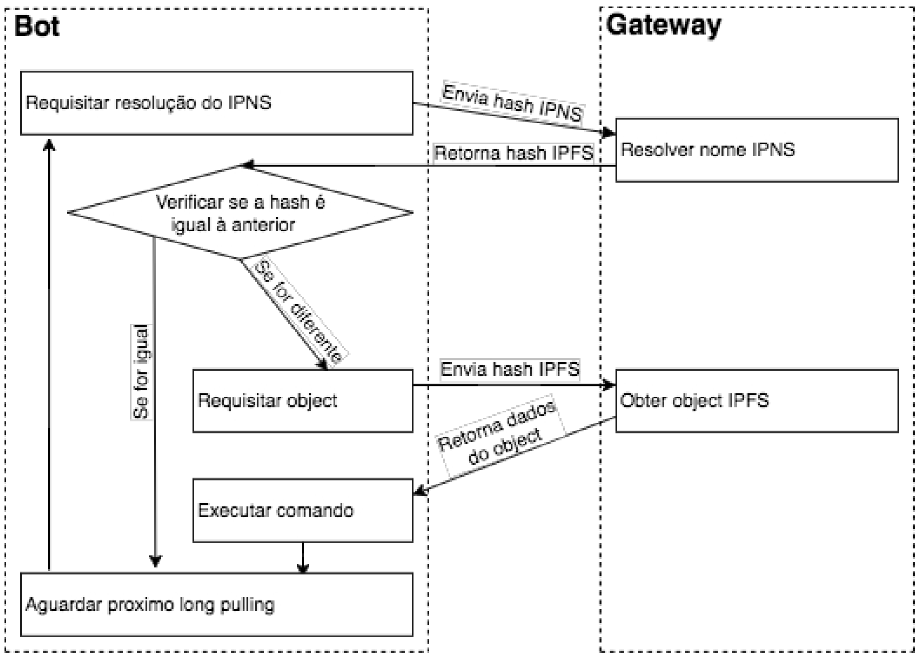

Figure 1. Flow diagram of the proof-of-concept execution.

Figure 1 shows the execution flow of the *bot* proposed in this paper. Periodically (*long polling*), the *bot* requests a *hash* addressed by IPNS to the [illegible]*ateway*,

2https://github.com/macabeus/ipfsbot

which responds by sending the *hash of* the *object* pointed to. The *bot* then checks, if the *hash* received is different from the one obtained in previous requests and, if it is different, requests the contents of the *object* associated with that *hash*. Finally, the *gateway* responds with the content of the object associated with the *hash*, which consists of the command issued by the *botmaster*. We decided to develop a single type of command, simple but useful for a *botnet*: *download* the file from a URL, save it in the temporary file directory, and execute it.

3.1. Results

In our experiments we were successful in sending commands to the *botnet* via the IPFS and IPNS protocols.

One of the characteristics analyzed was the propagation time of commands from the *botmaster* to the *bot*. There are two main factors that influence this time: the *long polling* check interval and the IPNS name resolution. While the first factor can be configured by the *botmaster*, the second is beyond his control. We noticed that IPNS name resolution usually takes a few seconds. In our tests, the resolution took at least 1 second and, on average, around 10 seconds. However, sometimes the resolution took longer than 30 seconds, a fact that led to a refactoring of the *bot to*, in these cases, give up the resolution and try again.

Another fact observed was that after inserting a new *object* in the IPFS and redirecting it to an IPNS *hash*, even if the *botmaster* was disconnected from the network, the IPFS noes (including the public *gateway*) were still able to access the IPNS address. This fact occurs due to the caching system, which can store data for up to 36 hours[3] . A *botmaster* could take advantage of this to increase his anonymity, as long as he republished his IPNS address from time to time so that the command issued would remain accessible to the *botnet*.

3.2. Discussion

One of the advantages of IPNS for the *botmaster* is related to registering domains for the C&C. *Botmasters* usually register domains to deal more easily with migrations or changes in the IP address of servers. However, they run the risk of being de-accredited by the registrar, completely losing control of their *botnet* [Bailey et al. 2009]. With IPNS, there is no de-accreditation: the closest would be for a *gateway* maintainer to add the *botmaster*'s IPNS *hash to* a *blacklist,* however, this could still be accessed from another *gateway*, or via a local *daemon. In* addition, creating multiple *peerIDs does* not incur any additional cost to the *botmaster,* allowing him to have multiple IPNS addresses to use as a channel.

IPNS naturally functions as a signature mechanism, ensuring that the *botnet* will execute only legitimate commands issued by the *botmaster*. An adversary would need to generate a valid signature in possession of only the public key (*peerID*) to be able to redirect the *botmaster*'s IPNS address to another *object on* the IPFS.

The IPFS and IPNS protocols do not by themselves provide a greater gain of anonymity, since it is possible to obtain the IP of whoever is distributing or receiving a given *object*. However, since these protocols are agnostic at the transport layer,

3https://github.com/ipfs/faq/issues/154

it is possible to use them in conjunction with other protocols that aim at anonymity, such as I2P and Tor[4] . Furthermore, the *botmaster* could use some tricks to issue commands with IPFS and remain hidden, such as using several *peerID* keys, as well as copying the *peerID* on different computers in different locations, using them to add new *objects* and redirect its IPNS address to them. In this way, IPFS would be responsible for maintaining and distributing the file for a period of time, without needing to maintain constant communication with the *botmaster* in this process.

3.3. Conclusion

This paper presented a proof of concept of a *botnet* whose C&C uses the IPFS and IPNS protocols. In the presented architecture, we [illegible] a command propagation time corresponding to the *long polling* interval added to the IPNS resolution time.

Future work could explore the IPFS and IPNS protocols more directly. For example, the *gateway* could be integrated with some of the *botnet bots*, avoiding the use of the public *gateway*. Another option would be to run the protocol *daemon* on all the *bots*. Both alternatives would allow the *bots to* send back commands through their own IPFS and IPNS protocols.

Due to the potential that the IPFS and IPNS protocols present for use in *botnets*, it is important that further studies and analysis be performed, allowing detection methods to be formulated and broadening the understanding of the risks involved.

References

Bailey, M., Cooke, E., Jahanian, F., Xu, Y., and Karir, M. (2009). A survey of botnet technology and defenses. In *2009 Cybersecurity Applications Technology Conference for Homeland Security*, pages 299-304.

Benet, J. (2014). IPFS - content addressed, versioned, P2P file system. http://arxiv.org/abs/1407.3561.

Dietrich, C. J., Rossow, C., Freiling, F. C., Bos, H., v. Steen, M., and Pohlmann, N. (2011). On botnets that use dns for command and control. In *2011 Seventh European Conference on Computer Network Defense*, pages 9-16.

Upadhyaya, A., Jayaswal, D., and Yadav, S. (2011). Botnet: A new network terminology. In *2011 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*, pages 424-428.

Wang, P., Wu, L., Aslam, B., and Zou, C. C. (2009). A systematic study on peer-to-peer botnets. In *2009 Proceedings of 18th International Conference on Computer Communications and Networks*, pages 1-8.

Zeidanloo, H. R. and Manaf, A. A. (2009). Botnet command and control mechanisms. In *2009 Second International Conference on Computer and Electrical Engineering*, volume 1, pages 564-568.

[4] https://discuss.ipfs.io/t/privacy-and-anonymity-in-ipfs-ipns/1068/4