Poster: Botnet Spoofing: Fighting Botnet with Itself

Cui Xiang*, Jin Shuyuan#, Liao Peng#, and Hao Zhiyu#
* Student # Faculty
Institute of Computing Technology, Chinese Academy of Sciences
{cuixiang,jinshuyuan,liaopeng,hzy}@ict.ac.cn

I. INTRODUCTION

A botnet consists of a network of compromised computers connected to the Internet that is controlled by a remote attacker (botmaster) via command and control (C&C) channels. Botnets are the root cause of many Internet attacks such as Email spam, extortion through DDoS, seeding malware, and online identity theft etc.

Recently, the arms race between botmasters and defenders has become increasingly common. Defenders have successfully shut down many well-known botnets such as Rustock, Mariposa, Waledac, Stuxnet, Coreflood, and Kelihos by exploiting their C&C design flaws. However, these countermeasures also stimulate the botnets to be more resilient. For security-conscious Internet users, the host-based security software (i.e., anti-virus and firewall) could provide effective protection against the botnet attacks; however, the remaining security-unconscious users will suffer from the botnet attacks and be compromised easily. Consequently, how to protect both security-conscious and security-unconscious users against advanced botnets (without any C&C vulnerability) has posed a great challenge to this day.

In this paper, we propose the idea of *botnet spoofing* which aims at addressing the above challenge to some degree. We define botnet spoofing as a technique which could trick a malicious bot to spread BotSpoofer instead of spreading itself. *BotSpoofer* is defined as a computer program which implements botnet spoofing technique.

II. OVERVIEW OF BOTNET SPOOFING

A. Basic Idea

The initial idea of botnet spoofing is inspired by cockroach killer. Cockroach killer could lead to a contagion inside the cockroach group. In other words, the infected cockroach is not expected to die immediately; on the contrary, it is misled to use its own addressing method to back infect other cockroaches. BotSpoofer works similar to cockroach killer to some extent. It tricks a malicious bot, using its own propagation mechanisms, to spread BotSpoofer instead of spreading itself to other victims. After BotSpoofer is delivered and executed on the victims, the victims will not only avoid an originally successful attack but also obtain extra protection provided by BotSpoofer.

B. Architecture

In the framework of botnet spoofing, Internet hosts can be classified into three groups; they are *Infected, Vulnerable and BotSpoofer* groups respectively. The Infected Group contains hosts which have already been compromised by a bot; the Vulnerable Group contains vulnerable hosts which could be located and then infected by a bot; and the BotSpoofer Group contains hosts which have been equipped with BotSpoofer. There are a total of three kinds of actions to trigger a host to transfer from one group into another group, as shown in Fig.1.

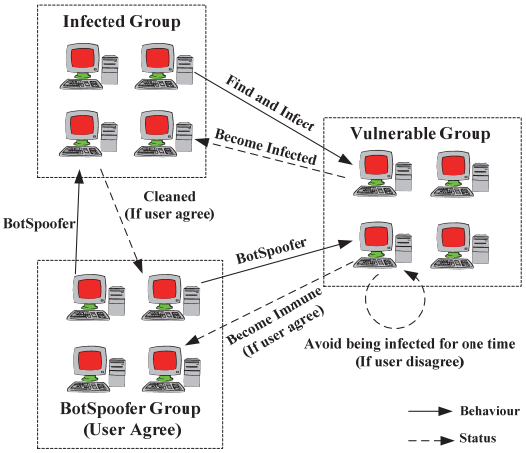

Figure 1. Architecture of Botnet Spoofing

Action 1 — triggers hosts to transfer from Vulnerable Group into Infected Group. When an infected host compromises a vulnerable host, the latter will become infected.

Action 2 — triggers hosts to transfer from Vulnerable Group into BotSpoofer Group. When BotSpoofer "compromises" a vulnerable host, the latter will equipped with BotSpoofer.

Action 3 — triggers hosts to transfer from Infected Group into BotSpoofer Group. When BotSpoofer "compromises" an infected host where a specific bot is resident, BotSpoofer will clean the infected host under user's grants.

With the increase in the size of BotSpoofer group and the decrease in the size of infected and vulnerable group, the specific bots will decline continuously. At the same time, the

vulnerable hosts will be protected automatically by the BotSpoofer.

C. *Spoofing Methodology*

Based on reading source code and reverse analyzing binary file of a good deal of self-propagating worms and bots (including but not limited to MSBlaster, Welchia, SDBot family, Rbot family, Agobot family, Slapper, Mariposa, Rustock, Conficker, Waledac, Bobax/Kraren, Storm etc), we have a fundamental observation that a persistent bot (both EXE and DLL) MUST obtain its file path for its subsequent auto-start registration and self-propagation. Therefore, the key issue of spoofing a bot lies in tricking it to retrieve a fake path.

API Hooking. We can hook some APIs(i.e., GetModuleFileName, GetModuleFileNameEx, and QueryFullProcessImageName) to cheat a bot to retrieve a fake file path. Specially, whenever the hooked API is called in the process space of a bot, the file path of BotSpoofer will be returned instead of the real path of the bot. Note that the hook operation can only be done in malicious bot's process space; therefore other processes will not be affected by API hooking.

Address Substitution. Bots relying on remote vulnerability exploitation must send Shellcode to remote potential victims. The Shellcode generally contains the IP and Port of the original bot host or a publicly available server which will be used for the subsequent reverse connection from the victim. Hence BotSpoofer could replace this address with a new one, by which way the remote Shellcode on victim will connect back to retrieve BotSpoofer instead of the real bot.

III. PROTOTYPE AND EXPERIMENT

A. *Prototype of BotSpoofer*

To prove the feasibility of botnet spoofing, we create a prototype named ConSpoofer to fight against Conficker botnet. Conficker botnet [1, 2] is one of the most widespread and sophisticated botnet till now. It constructs an extremely resilient C&C channel by combining Domain-Flux and unstructured random-scan based P2P protocol, which makes the traditional countermeasures inefficacious.

PE Information. We randomly select a Conficker.B sample. Its MD5 is 3aff8601a8a6fc1dccb836ae3e971e3e, file name is atsjshck.dll, and file size is 158967 bytes.

Locating Method. Conficker, like majority of botnets, employs GetModuleFileName API to locate its file path. Thus we could adopt API hooking method to spoof it. Note that other methods such as address substitution are also feasible. When propagating via MS08-067, Conficker needs to send a Shellcode containing the IP address of infection source and a HTTP PORT opened by it. ConSpoofer can replace the PORT to make the remote victim connect a new HTTP server opened by ConSpoofer and then download ConSpoofer.

Loading Method. When arriving at a new victim, Conficker always uses rundll32.exe as its loader. More specially, when Conficker tries to propagate itself via MS08-067, removable devices, and network share, it always relies on rundll32.exe to execute itself for the first time.

Self-Verification Method. Similar to most of current bots, Conficker doesn't verify the authenticity of its executable file, so we don't need to patch it.

Architecture. The file and image of ConSpoofer are shown in Fig.2.

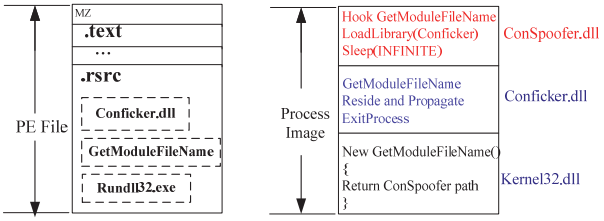

Figure 2. ConSpoofer File and Image

B. *Experiment*

Conficker employs three propagation vectors, so we set up an experiment environment with four computers, where computer V has MS08-067 vulnerability, computer W has weak password vulnerability, computer U has autorun feature, and computer S is an infection source which equips with ConSpoofer.

After ConSpoofer is activated on S for several minutes, S infects V and W successfully. On V, ConSpoofer is executed by the Shellcode sent from S to V. On W, ConSpoofer is added to Task Scheduler and waiting to be executed. In addition, after an USB device is inserted into S, Autorun.inf and ConSpoofer files are created in the USB device immediately. When the USB device is connected to U, ConSpoofer is activated. Furthermore, ConSpoofer is registered as an auto-start service automatically by the released Conficker in all computers.

IV. CONCLUSIONS

In this paper, we propose the idea of botnet spoofing. In comparison to traditional mitigation strategies, the distinctive advantage of botnet spoofing lies in its capability of protecting a large-scale of network space, including both security-conscious and security-unconscious hosts. Although the legal and ethical issues are no doubt controversial, we hope the idea of botnet spoofing will inspire more research on how to defend against advanced botnets in the future.

REFERENCES

[1] Seungwon Shin, Guofei Gu, Narasimha Reddy, Christopher Lee. "A Large-Scale Empirical Study of Conficker." To appear in IEEE Transactions on Information Forensics and Security, 2012.

[2] P. Porras, H. Saidi, and V. Yegneswaran. A Foray into Conficker's Logic and Rendezvous Points. In USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'09), 2009.