

Moving Target Defense for Cloud Infrastructures: Lessons from Botnets

Wei Peng, Feng Li, and Xukai Zou

Abstract While providing elasticity to clients through on-demand service and cost-effectiveness to service providers through efficient resource allocation, current cloud infrastructures are largely homogeneously and statically configured for ease of administration. This leaves ample opportunities for attackers to reconnoiter and penetrate the security perimeter of cloud services. This chapter (1) explores the evolution in botnet technologies from the early static architectures to the recent dynamic and resilient architectures that employ various moving target defense (MTD) techniques to circumvent crackdowns, and (2) draws lessons from botnets in identifying cloud security challenges and proposed solutions to MTD for cloud infrastructures, in which the cloud infrastructure configuration constantly evolves to confuse attackers without significantly degrading the quality of service. Proposed solutions may increase the cost for potential attackers by complicating the attack process and limiting the exposure of network vulnerability in order to make the network more resilient against novel and persistent attacks.

1 Introduction

Cloud computing has emerged as a mainstream computing and storage service model for personal, business, and government affairs out of its roots on autonomic computing [50, 59, 67], grid computing [15] and utility computing [10]. One characteristic of cloud computing is multi-tenancy. While providing elasticity to clients through on-demand allocation and cost-effectiveness to cloud service

W. Peng • F. Li (✉) • X. Zou

Indiana University-Purdue University Indianapolis, Indianapolis, IN, USA

e-mail: pengw@iupui.edu; fengli@iupui.edu; xkzou@iupui.edu

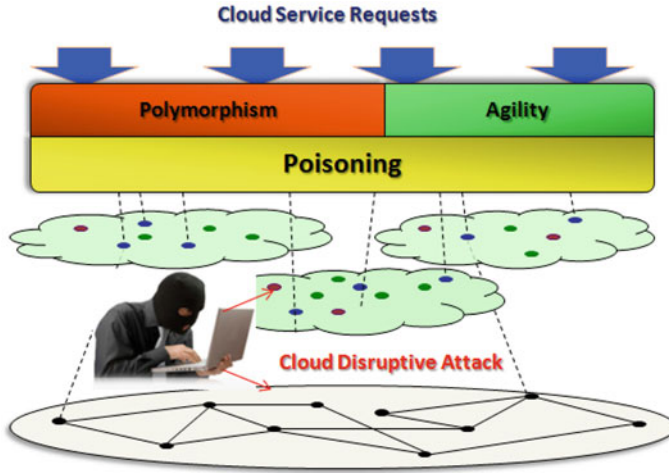


Fig. 1 Botnet moving target attack technology

providers (CSPs) through efficient resource allocation, multi-tenancy has its own security implications. For example, client virtual machine (VM) instances running on the same physical machines are susceptible to side-channel attacks [32].

The automated process of allocating resources on client demands creates many server instances with identical or very similar configurations, in which a single VM-level compromise may quickly scale up to a service-level breach due to the high homogeneity. Homogeneity and static configurations provide attack opportunities to botmasters. Botnets [24, 60, 63, 107] have plagued the Internet for over a decade. Studies on discovered botnets put the number of bots on the order of hundreds of thousands [46] to millions [34]. Though only a fraction of the whole bot population may be online at the same time [93] due to diverse geographical distribution and diurnal pattern of the bots [23], the cumulative bandwidth and computational capacity of the bots at the disposal of botmasters enable numerous nefarious activities, including email spam campaigns [86, 88, 114], distributed denial of service attacks [9, 105], key logging [48], and identity theft [73]. Despite intensive cyber security research efforts to mitigate botnets, botnets are still active [88, 108]. Dainotti et al. [24] reported the botnet's scanning behavior, including general methods to correlate, visualize, and extrapolate botnet behavior across the global Internet. "Botnets rival the power of today's most powerful cloud computing platforms. These *dark clouds*, controlled by cybercriminals, are designed to silently infect your network." [19]

Botnets are fast-moving targets (see Fig. 1) that are difficult to detect with conventional security tools. Therefore, *moving target defense (MTD)* has become a major theme in cyber-security researches since mid-2010 [55, 56, 101].

According to the U.S. Homeland Security Cyber Security R&D Center [30]:

Moving target defense (MTD) is the concept of controlling change across multiple system dimensions in order to: increase uncertainty and apparent complexity for attackers, reduce their window of opportunity and increase the costs of their probing and attack efforts. MTD assumes that perfect security is unattainable. Given that starting point, and the assumption that all systems are compromised, research in MTD will focus on enabling the continued safe operation in a compromised environment and to have systems that are defensible rather than perfectly secure.

MTD enables us to create, analyze, evaluate, and deploy mechanisms and strategies that are diverse and that continually shift and change over time to increase complexity and cost for attackers, limit the exposure of vulnerabilities and opportunities for attack, and increase system resiliency.

In the context of cloud infrastructures, MTD is motivated by the *asymmetric* [55] costs borne by defenders and attackers. While the defenders need to secure the entire system against potential attacks, a single vulnerability is enough for the attackers to break into the system [55, 56]. This is exacerbated by the growing complexity of modern systems. MTD tries to tilt the balance towards defenders over attackers by dynamically and proactively changing configurations of the cloud infrastructures.

Research and development of MTD in cloud infrastructures are still in an early stage. How to interpret and implement MTD in the context of cloud infrastructure security is still an open challenge. The objectives of this chapter are to provide a comprehensive botnet survey and an introduction to MTD cloud infrastructures for cloud researchers, administrators and developers.

This chapter is organized as follows: Section 2 provides the evolution in botnet design from the early static architectures to the recent dynamic and resilient ones through various MTD techniques. Section 3 identifies cloud security challenges, describes botnets (dark clouds) versus clouds, and provides illustrative MTD techniques for secure clouds. Section 4 presents the proposed solutions to MTD for cloud infrastructures by drawing lessons from botnets. Conclusions are presented in Section 5.

2 Moving Target Defense: Lessons from Botnets

Research on botnet detection/mitigation evolves over time. Literatures on botnet research appeared in academia around 2003 [71]. Early signature-based techniques [64, 122], which were and still are employed in many Honeynet projects [54], fail to detect both polymorphic [119] variants of old botnets and completely new botnets. Later development of botnet detection methodology includes anomaly-based [42], DNS-based [123], mining-based [36] and heuristic-based [72] techniques, and the techniques specially designed to cope with fast IP/domain flux used by botnets [4, 129].

The following subsections provide a few of the aspects of botnets that have been investigated.

2.1 Botnet Formation and Communication

Consensus on the life cycle of a botnet [44] consists the following stages [120]:

- *Botnets Attack Vectors*: Botnets share the attack vectors with other forms of malware, including server exploits, trojan/rootkit piggyback, social engineering through spamming [90, 94], and other advanced forms of attack vectors [31]. Studies on captured bot samples indicate that modern botnets employ multiple attack vectors to maximize the chance of propagation [1]. A common trend in this stage is the increasing emphasis on *social network* vulnerabilities [12].
- *Rallying*: Methods for a newly infected bot to join the existing botnet include hard-coded IP/domain/IRC-channel names (e.g., Akbot [85]), external configuration file (e.g., Trojan.Peacomm [41, 46]), and dynamically generated rendezvous (e.g., Torpig [107]). This stage is called *rallying*. A common trend in this stage is the transition from *random* sampling, such as consecutive scanning of a whole IP block, to *targeted* probing, such as hard-coded rendezvous.
- *Command and Control (C&C)*: Existing botnet C&C channels include public Internet services such as IRC, HTTP, DNS, and various P2P protocols [132]. A common trend in this stage is the migration from traditional *centralized* IRC or web-based channels [22] to more robust *distributed* P2P or *tiered hybrid* channels.

2.2 Botnet Population Measurements

The size of a botnet is characterized by two metrics such as *footprint* and *live population* [93]. Footprint measures the cumulative number of bots over the entire lifetime of a botnet; live population measures the dynamics of online bot population reachable from the botmaster. In addition, the temporal/spatial distribution of the bots in a botnet is also of interest to both the botmaster and the defender. Measurement techniques include both *passive* detection [18] and *proactive* infiltration [1, 107]. Measurement results indicate that a significant portion of the bot population are behind Network Address Translation (NAT) firewall, possibly in home or small-office/home-office (SOHO) settings [31, 57].

2.3 Botnet Technologies

Recent developments in botnet design, especially the rallying and C&C stages, embody the principles of MTD. Detailed discussions are in Section 2.5. The other stage, initial infection of bot, is usually accomplished by a combination of drive-by download [91, 103], software vulnerabilities [131], and social networks [12, 124]. The following is a list of a few real cases.

2.3.1 Drive-by Download

Tidserv rootkit [113], used by the *TDL4* botnet [100], is bundled with rogue security software and infects low-level system printer and filesystem drivers. It also blocks system update and disables some anti-malware programs. Then, *Tidserv* modifies the boot record, i.e., Master Boot Record (MBR), on the hard disk so that it is loaded and executed prior to the operating system every time the system reboots. Through this technique, *Tidserv* circumvents the system's mandatory driver signing mechanism. *Mebroot* rootkit [43], used by the *Torpig* botnet [107], infects systems through injecting malicious HTML and JavaScript scripts that exploit vulnerabilities of Web browser plugins. If any such exploit is successful, a copy of the *Mebroot* rootkit is downloaded and executed on the victim's computer. Like *Tidserv*, *Mebroot* also modifies the MBR to circumvent detection by anti-malware programs. Early variants of the *Zeus* botnet [2, 11] also adopt drive-by download for initial infection by redirecting victims to a webpage that contains a malicious PDF file that exploits known vulnerabilities in the Adobe Reader software [77]. Similar vulnerabilities on Adobe Reader are also exploited by the *Gumblar* botnet [112]. *Asprox* botnet [92] launches SQL injection attacks [13] against vulnerable pages based on Microsoft Active Server Page (MSASP) to inject malicious scripts for propagating malware.

2.3.2 Software Vulnerabilities

In some variants of the *Conficker* botnet [21, 89], new victim computers are infected by existing bots through specially crafted Remote Procedure Call (RPC) requests. The requests will trigger buffer overflows, which allow the existing bots to send and install the malware on the victim computers without victim users' knowledge. Other variants of *Conficker* launch dictionary attacks against default shared resources.

2.3.3 Social Networks

Email spamming has been the most popular malicious activity of botnets. Botnet owners make money through renting their botnets for spamming, and compromise computers or web sites for botnet expansions. For examples, the *Srizbi/Reactor* botnet [111] was behind the Ron Paul spam campaign [87]. *Zeus* uses Facebook phishing [27] and fakes billing emails from Verizon Wireless [51] to initiate drive-by download. The *Nugache* botnet [31, 109] lures victims into downloading and installing the malware (packaged into a popular video editing application) by using existing bot population to boost the visibility of the malware on popular software download aggregation sites through fake downloading [31]. A wide variety of social network-based spam campaign measurement studies and detection techniques are in [39, 86, 88].

Next, we will first discuss a few moving target defense techniques observed in real-world botnets (Section 2.5), and then sample a few botnet technologies from

proactive botnet-mitigation research efforts (Section 2.6). We will notice a common trend of adopting dynamic and resilient moving-target defense mechanisms in both real-world and research botnet technologies. We conclude by relating these recent developments of moving-target defense in botnets to the rise and fall of early static IRC-based botnets (Section 2.4).

2.4 Rise and Fall of IRC-Based Botnets

According to Ferguson [37], the *Sub7* torjan and *Pretty Park* worm, which both surfaced in 1999, first introduced the concept of *malicious* bots: Victim machines connect to an IRC channel, waiting for commands issued by a remote attacker. Prior to that, bots were used on IRC channels for benevolent purposes such as automating channel administration and providing help to new users. Some notable developments in early IRC bots include Gtbot, Agobot and Spybot. In 2000, the Gtbot, which was based on the mIRC client, could initiate rudimentary DoS attacks due to the possibility of running IRC-event-triggered scripts, and having raw access to Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) socket programming interface provided by the mIRC client. In 2002, the commercialization of *Sdbot* allowed development of a new botnet to be based on previous ones. In the same year, *Agobot* introduced the concept of a modular staged attack packaged as additional malicious payloads over an underlying backdoor. The initial attack is to set up the backdoor, which paves the way for later attacks that are packaged into small modules. In 2003, *Spybot* introduced key-logging, data mining, and instant messaging spam into botnet. Later that year, *Rbot* introduced SOCKS proxy and the use of compression and encryption to evade detection. Despite expansion in functionality, these early botnets all use the IRC protocol as the C&C channel. The former open nature and wide adoption of the IRC protocol proved to be vital for the success of early botnets. However, as more IRC-based botnets surfaced, system administrators became alerted to unauthorized IRC traffic and, hence, IRC ports were disabled by default and IRC traffic served as a sign for malware infection.

A real-world example is the “Operation: Bot Roast”, an international investigation against botnets led by the United States Federal Bureau of Investigation (FBI), in which five botnet authors and operators were arrested and charged, including the 18-year old New Zealand author of *Akbot* [70], an IRC-based botnet [35]. In *Akbot*, after a victim is infected through remote exploitation of the Windows operating system [74], the malware downloads and installs itself through an FTP server, attempts to join three fixed IRC servers on TCP port 6584 and waits for the botmaster’s instructions. *Akbot* was detected because of its constant IRC traffic and static IRC server/port. This is a vivid example of how static configurations in early IRC-based botnets expose the botnets and compromise their operation; this is why later botnets move on to more dynamic settings.

In late 2007, Zhuge et al. [133] reported their 1-year measurement study of IRC-based botnets. In their study, a honeypot-based, distributed, and fully-automated

measurement system, consisting of 50 malware sensors deployed on 17 nodes, was used to track botnet activities on their C&C IRC channels. During the study, an average of 2,800 samples were captured every day, and 3,290 unique (in terms of DNS name, port number, and channel name) IRC-based botnets were identified from the samples. IP-address-based geographic analysis indicated that almost 38.8 % of the C&C channels were hosted in the United States, followed by China, Korea, Germany, Netherlands, Canada, Sweden, Great Britain, and other countries, and almost 15.1 % of the bots were located in Brazil, followed by China, Malaysia, Taiwan, Korea, Mexico, Russia, Argentina, India, and other countries. Only about 36.1 % of the discovered botnets used the standard IRC port 6667 to host the C&C channel; the majority other used non-standard ports for C&C. This confirmed earlier observations that IRC-based botnets started to use non-standard configuration to evade simple port-based detection. For example, 1.3 % of the discovered botnets used TCP port 135, which is commonly used by Windows for file sharing. The study found an average lifetime of 54 days for the C&C servers and, among the 3,290 botnets they monitored, 378 were still alive in mid-June, 2007. This indicates the resilience of the botnets. During the study, a total of 1,520,000 distinct bot IDs were observed for 1,904 (57.9 %) out of the 3,290 botnets. For 1,110 (33.7 % of the total) of the 1,904 botnets, 700,700 distinct IP addresses were observed, and the largest observed botnets consisted of more than 50,000 IP addresses. Though unique IP addresses do not translate to unique bot due to churn effect and dynamic IP assignments, this still shows the prevalence of botnets.

The reason that the researchers could measure the size of the 1,904 botnets is because these botnets did not disable the user listing IRC command on their C&C channel. While taking the advantage offered by the IRC infrastructure, these botnets did not customize the protocol to suit their needs and thus suffered the consequences of being easily detectable. Had they changed the default configuration and customized the underlying IRC protocol, and thus moved away from being *static* to being *dynamic*, it would have been much harder to infiltrate them.

2.5 Moving-Target Design Examples in Real-World Botnets

After the fall of IRC-based botnets (Section 2.4) due to the easy-to-detect C&C traffic on IRC channels, botnets adapted by migrating away from IRC to other HTTP-based or customized C&C channels. Therefore, botnet network topologies are hard to be identified. At first, a single C&C server, corresponding to one IP address or domain name, was used. For example, initially, Gamblar connected to a fixed domain gumblar.cn, and the taking down of this domain in May 2009 apparently shut down Gamblar [20]. Gamblar reappeared later, and used multiple domain names for rallying, making it harder to detect and stop. *Fast flux* [49] was devised by botnet creators in response to the crackdown of individual C&C servers. With fast flux, a bot queries a known domain name, which is associated with a DNS record with short time to live (TTL) value and thus would be translated into a

series of IP addresses in a quick and round-robin fashion. This reduces the impact of a single C&C server being taken down on the operation of the whole botnet. Some traditional system defense mechanisms, like IP-based access control, become ineffective against fast flux. Storm [28, 38, 40] and WarezoV/Stration [110] were among the first botnets that adopted fast flux, with Wibimo [45] being a more recent example.

Although fast flux solves the single-point-of-failure problem (for the botnet) with regard to C&C server's IP address, the unique domain name still leaves a single point of failure for authorities to take down the botnet. New generations of botnets solve this problem with a technique commonly known as *domain flux*. With domain flux, each bot independently and periodically computes a list of domain names with a customized *domain generation algorithm* (DGA). The bot proceeds to contact the hosts with these domain names one by one until a host responds and validates itself as a C&C server under the botnet protocol. Due to dynamic assignment and decentralized management of the rendezvous, even if one of the C&C domain names is blocked by ISPs or taken down by authorities, it is likely that another C&C domain name is still valid, and thus could be used by the bots to locate the C&C server. DGAs were previously used as the primary network evasion technique in many highly publicized and well studied botnets [26], including Conficker [21, 89], Murofet [16], and Torpig [53, 107]. Recent variants of Zeus use DGAs as backup strategies for rallying should their primary rallying mechanisms, such as peer-to-peer channels or hard-coded IP addresses, fail [25]. In a case study on a previously unreported botnet sample conducted by Antonakakis et al. [3], the botnet uses the date as the seed to its pseudo-random-number generator and generates approximately 1,000 domains, employing purely alphanumeric characters and a particular top-level domain (TLD) for a given day. Another example is Torpig. Torpig uses a two-tier DGA. Each bot first generates a weekly domain name, which only depends on the current week of the year, and appends a few common top-level domains (such as ".com" and ".net") to the weekly domain name. The bot tries to contact each of these domain names until an attempt succeeds. If all fail, the bot generates another daily domain name, appends the TLDs, and contacts the resulting domain names in turn. If all fail, the bot resorts to a number of hard-coded domain names in its configuration file. The DGA used in Torpig is completely deterministic. This, along with the weak obfuscation over the C&C communication channel, allowed researchers to take over the botnet for 10 days between 25 January 2009 and 4 February 2009 [107].

The network topology used by botnets to organize the bots and C&C servers also evolve, from the centralized architectures (early variants of Zeus, Torpig, and Grum [79]) to more dynamic and robust P2P (used in TDL-4 [100], later variants of Zeus [61], Nugache [31, 109], and Storm [28, 38, 40]) or hybrid (Waledac [81, 102] and Sality [34]) ones. Interestingly, a partial reversion to hard-coded IP lists for initial rallying in some recent P2P botnets (e.g., recent variants of Zeus [61]) was observed. This is, perhaps, due to botmasters' aversion to inherent latency in coordination and control introduced by the P2P architecture. This indicates the need to strike a balance in real applications between the security gain and the performance

hit, both introduced by the adoption of moving-target defense. In the rest of this section, we will use Nugache and Waledac to illustrate the operations of P2P and hybrid botnets, respectively.

Nugache has evolved over time. The original *Nugache*, first documented in late April 2006 [80], was largely dismissed by the security community as trivial to detect due to a few distinctive *invariants* such as connections to certain TCP ports. After the malware was updated to use random high-numbered port for communication, *Nugache* stayed largely undiscovered until the arrest of its author in September 2007 [78]. Unlike the *inside* study used on Torpig [107] for infiltration and takeover, which exploited weak obfuscation in the communication protocol and deterministic domain-name probing in the domain-name generation algorithm used by Torpig, the study on *Nugache* was more difficult, due to its strongly encrypted communication channel. Traditional honeypots [47], which worked well in detecting centralized botnet and collecting information (e.g., captured packet, intrusion detection signatures, and passive operating system fingerprints), fell short in detecting P2P botnets. In their study on *Nugache* [31], Dittrich and Dietrich specifically customized the honeypots to deal with the peculiarities of *Nugache*. Unable to infiltrate the C&C structure and examine the encrypted message exchanged by peering bots, the analyses on *Nugache* were largely based on external traffic analysis which included P2P connections, probing associated with remote vulnerability exploitation, and DDoS attacks through the P2P network [31]. Traffic analysis on the *Nugache* botnet trapped in the honeypots indicates that *Nugache* does not have a centralized C&C structure. Each bot makes infrequent inbound/outbound connections at the rate of dozens per day. The P2P structure with strong encryption allows the botmaster to control a significant number of hosts with only a small percentage of them actively probing and forwarding commands at the same time. Also, each bot only maintains a few peers to minimize exposure, in case that the bot is captured by the defender. IRC logs recorded for an early IRC-based variant of *Nugache*, which contain systematic probing in the non-routable IP address blocks reserved for intranet by RFC 1918 [95], indicate that the small-office/home-office (SOHO) networks were targeted by Torpig.

Waledac emerged in late 2008 after the infamous Storm botnet [28], which gained its notoriety through its large infection base (various sources put the number of bots in Storm from 250,000 up to a few million [38]), through its deliberate counter-attack against investigation (Storm was known to launch DDoS attacks against security vendors and researchers who participated in its investigation [40]), and through its enormous cumulative computing resources and bandwidth (some claimed that the cumulative computing capability was greater than some supercomputers [115]). An early variant of *Waledac* was delivered through the same backdoor used for carrying Storm; thus, *Waledac* is considered a descendant of Storm. In *Waledac*, the bots are divided into two layers, *spammers* and *repeaters*, based on whether a bot is behind network address translations (NATs): Those behind NATs, which do not have a public-accessible IP address, are the spammers and the other publicly accessible bots serve as repeaters. Within the botnet, each spammer communicates exclusively with an upper-tier repeater; repeaters, besides

serving a few spammers, communicate among themselves. This structure is similar to the tiered architecture used in the Skype VoIP system [6]. P2P communication is restricted to the repeaters, which are responsible for collecting data from spammers and distribute commands from the botmaster. Unlike Storm, Waledac does not communicate through the decentralized *Overnet P2P networks using Kademlia* [69], but exclusively through encrypted HTTP with fast flux. The rallying for Waledac is much like Storm. Each newly infected bot finds a neighboring repeater through probing a hard-coded list of IP addresses. If the probing fails, the bot will download an IP-address list through a hard-coded URL, which is fortified with fast flux to reduce the chance of being taken down. Repeaters are solely responsible for coordinating communication between the spammers and upper-tier nodes, which are under close control of the botmaster. Early research speculated that the upper tier was also tiered [102], which was later verified when the researchers Nunnery et al. [81], collaborating with two of the affected hosting providers in the Netherlands, were able to obtain the file-system images and network traces of the servers serving as upper-tier botnet nodes. Sinclair et al. [102] defined that “TSL is the name of the Windows registry entry that the Waledac binary uses to store a list of servers for this tier. As such, we named the list of these servers as the TSL layer.” Note that the meaning of this acronym is unclear. The upper tier consists of two additional layers: several TSL servers and a single Upper Tier Server (UTS). The UTS server is the ultimate C&C server directly controlled by the botmaster. The TSL servers are responsible for coordinating communication between the UTS server and the repeaters, and take their name from an entry in the repeaters’ local configuration listing their corresponding TSL servers. The TSL servers are set up with pre-packaged customized software stack (which includes the operating system) and are hosted on third-party hosting services. They insulate the UTS server, which is the C&C center, from lower-tier bots. In case a TSL server is taken down, the botmaster can set up a new one and relegate the repeaters corresponding to the compromised TSL server to the new server through the repeater-layer P2P channel. Waledac is considerably more resilient against crackdown than earlier centralized versions, due to its dynamically tiered hybrid C&C structure.

2.6 Towards More Resilient Botnets

The ongoing battle between the botmasters and the security professionals prompts some researchers to take a more proactive approach. The motto for this approach is “forewarned is forearmed”: Rather than conducting *postmortem* analysis when the distress has been caused, the best way to defend against an unknown botnet is to explore advanced botnet design and mitigation before they have been seen in the real world. In this section, we sample two such proposals.

Wang et al. [121] propose a hybrid botnet design. Their design is motivated by the following challenges faced by botmasters [121]:

- How to reduce the chance of being detected by the defenders via communication traffic analysis?
- How to minimize the exposure of the whole network to the defender, if some bots are captured?
- How to maintain a robust network for the rest, if a substantial number of bots are taken down?
- How to monitor the botnet given the constraints implied by the above challenges?

The key ideas are [31, 121]:

- The differentiation between two types of bots, *slaves* and *servants*, based on whether the IP address is publicly accessible. Only the servants, which have publicly accessible IP addresses, will appear in the peer-list of a bot. This is similar to the spammer-repeater distinction in Waledac.
- Infection is through a worm-like channel, in which the infector and the infectee can directly communicate. The infector shares with its infectee its peer-list; if the infector is a servant itself, the infectee adds the infector to its peer-list. This eliminates the bootstrap phase, which is often the Achilles' heel of a botnet due to the staticity in this phase.
- The number of peers in a peer-list does not exceed a system parameter. Thus, each bot only knows a small portion of the whole botnet population. This reduces the chances of exposing the whole botnet in case one bot is compromised.
- A botmaster could monitor the entire botnet by issuing a special report command, which instructs the bots to report to a compromised machine called the *sensor* host. The sensor host changes every time to avoid being compromised by the defender.
- For each botnet, the service port for incoming connections is randomly chosen, and every connection is encrypted by a locally negotiated symmetric key to prevent sniffing. The increased dynamicity reduces the impact of infiltration and poisoning attacks.

Dittrich and Dietrich [31] report that most of these ideas, perhaps with the exception of the monitoring mechanism using sensor hosts, have already been used by Nugache. Wang et al.'s design was partially inspired by the deficiencies in an earlier variant of the Nugache botnet with a IRC C&C channel.

Inspired by the works of Wang et al. and others, Liu et al. [66] proposed a recoverable hybrid botnet design. Their design is motivated by two challenges: (1) to recover from the event of C&C being taken down and (2) to reduce the impact of P2P routing table poisoning. The key ideas are [66]:

- The C&C structure consists of two independent but coordinated mechanisms: *decentralized hybrid P2P-based C&C (HPCC)* and *centralized domain-flux-based C&C*, i.e., URL Flux-based C&C (UFCC).
- The HPCC uses the servant/slave distinction proposed by Wang et al. [121].
- The UFCC is hosted on a few robust Web 2.0 services, which are used to publish command and links to malicious payloads.

- A propagation-based reputation system is used in peer-list exchange to avoid Sybil attacks [28] and P2P routing table poisoning against the botnet.

Unlike real-world botnets, these hypothetical designs lack convincing verification on their effectiveness, which, paradoxically, is available only if they have been implemented, released, and tested in the real world. This is a dilemma faced by researchers taking the proactive approach to botnet mitigation. This echoes the call for action from Aviv and Haeberlen [5] for a PlanetLab-like botnet-research testbed, where researchers can test and verify their ideas without disrupting the whole Internet. Nevertheless, these designs, along with the real-world botnets before them, clearly indicate moving-target defense as the unifying theme in future research on botnet design and mitigation.

3 Moving Target Defense: Towards Secure Clouds

Elasticity has attracted many organizations to migrate their computing and storage services to clouds, which provide on-demand provision of computing resources such as processor time, memory, and mass storage. The ever increasing information assets at stake have raised security concerns over clouds. In this section, we will discuss cloud infrastructure security challenges based on lessons from Botnets for cloud security.

3.1 Cloud Infrastructure Security Challenges

The transfer of management responsibility from customers to CSPs, while providing elasticity to clients through on-demand allocation and cost-effectiveness to service providers through efficient resource allocation, introduces numerous security challenges that permeate the whole cloud infrastructure, from the bottom hardware layer, through the medium VM layer, up to the operating system layer that supports customer applications. We briefly discuss a few such challenges below.

Virtual local area networks (VLANs) are often used for traffic isolation and for providing layer-2 QoS in the data center Ethernets. Rouiller [97] summarizes some cloud infrastructure security challenges in layer-2 VLANs as follows.

- *MAC Flooding Attack*: Attackers send numerous fake Media Access Control (MAC) address queries to the switches. This will saturate the MAC table associated with each VLAN port, after which the Ethernet switch will essentially degenerate to a hub by broadcasting every MAC message, and thus the frames could be sniffed.
- *Layer-2 Routing Manipulation Attack*: Some versions of the Spanning Tree Protocol (STP) are used by Ethernet switches to implement loop-free layer-2 routing. It is possible to corrupt the configuration data such that the Bridge

Protocol Data Unit (BPDU) messages used in the STP cause a switch to be elected as the root of the spanning tree and thereby have the traffic directed as desired by the attacker.

Besides the layer-2 Ethernet switches used for internal interconnection, layer-3 [84] routers are used to connect cloud data centers to the external world. These routers become more important in infrastructures with distributed data centers, to support a seamless cloud computing environment. Cloud infrastructure challenges in layer-3 routing include the following:

- *QoS Misconfigurations*: Improperly configured IP-flow and QoS related settings, such as Multi-Protocol Label Switching (MPLS) signaling [83, 98, 128] and differentiated services [62].
- *Layer-3 Routing Poisoning Attack*: Forged or tampered routing messages, which would lead to catastrophic consequences if the messages are accepted without authentication.

The complexity associated with the rich features provided by inter-domain routing protocol such as Border Gateway Protocol (BGP), coupled with the reluctance of some Internet service providers (ISPs) to share their configuration, leads to plenty of chances of misconfigurations [7], manifested as misconfigured address-prefix advertisements, alternative routes, or packet filtering rules. For example, in *BGP prefix hijacking* [83, 98, 128], an autonomous system address space is incorrectly announced without the owner's permission, perhaps due to misconfigurations or deliberate attacks. This will negatively affect the availability of cloud-based resources. Studies indicate that BGP prefix hijacking occurs several 100 times per month due to misconfigurations and less than a 100 times per month due to deliberate attacks. A real-world instance of BGP prefix hijacking is that in 2008, Pakistan Telecom attempted to block YouTube in the country, due to suspected blasphemous video being hosted there, by announcing an incorrect route to the service; YouTube became unavailable world-wide for 2 h as a result [106].

The increased external DNS querying due to outsourcing services to clouds makes DNS-based attacks particularly challenging in cloud environments. For example, vulnerabilities in many deployed DNS servers were discovered that allowed attackers to direct legitimate DNS queries from users to malicious domains under attackers' control by poisoning the DNS caches [99]. The vulnerabilities were rooted in the lack of both strong authentication (that prevents attackers from manipulating queries) and sufficient protocol randomness (that prevents attackers from being able to fake query responses with correct IDs) in deployed DNS software implementations.

In clouds, the abstraction of infrastructures and services also means that customers usually do not have the ability to precisely control the visibility and lifetime of some underlying resources, such as IP or physical address caches. There are lags between changing resource IP address and updating the new address in DNS caches, and similarly, between changing physical (MAC) address and the clearing of old entries in Address Resolution Protocol (ARP) caches. This means that some

resources that have migrated or are being removed might still be accessible in the caches. According to Mather et al. [68], earlier reports on IP-address aging problems were likely the impetus behind the announcement of Elastic IP services from Amazon Web Services (AWS) in 2008 [14].

Currently, virtualization is implemented by one of several models, including OS-level virtualization (e.g., Solaris containers and Linux/BSD jails), paravirtualization (a combination of hardware/software virtualization), and hardware virtualization (Xen, VMware, and Microsoft Hyper-V). Fortifying virtualization is critical to securing cloud infrastructures. In a real-world incident in 2009, attackers erased over 10^6 websites hosted by UK based web hosting service provider VAserv, by exploiting a zero-day vulnerability in the hypervisor manager used by the company [117].

Client VM instances running on the same physical machines are susceptible to side channel attacks. Using the Amazon Elastic Compute Cloud (Amazon EC2) service, Ristenpart et al. [96] demonstrate that it is possible to map the internal cloud infrastructure, to identify where a particular target VM is likely to reside, and then to instantiate new VMs until one is placed co-resident with the target. Such placement can be used to mount cross-VM side-channel attacks to extract information from a target VM on the same machine.

Similar to centralized C&C center in the botnets, cloud infrastructures maintain configuration information in centralized databases. When new servers are allocated, such databases are automatically duplicated for ease of management and allocation efficiency. The automated process of allocating resources on client demands creates many server instances with identical or very similar configurations, in which a single VM-level compromise may quickly scale up to a service-level breach due to the high homogeneity.

3.2 Similarities and Dissimilarities: Botnets and Clouds

Security is a hidden feature, where “no news is good news”, and is often shadowed by more visible features such as usability and performance. The explosive growth of cloud services has outpaced the security measures used to protect their infrastructures. This situation bears resemblance to the early golden age for the IRC botnets, in which botnet authors pushed the boundary of what an IRC botnet could do without concern for the vulnerabilities in the underlying IRC infrastructure. For example, almost a third of all the botnets studied by Zhuge et al. [133] used the default IRC port 6667, and over one half of the botnets allowed unauthorized query for channel membership provided by the IRC protocol; this directly led to the exposure of whole botnets. When the IRC C&C channel became the single point of failure of a botnet, botnets began to adopt MTD techniques like fast IP/domain fluxes and more resilient hybrid or purely distributed C&C architectures. Conceivably, something similar will happen for clouds.

In the rest of this subsection, we will identify a few similarities and differences between clouds and botnets, and suggest a few lessons that can be learned from botnets for securing cloud infrastructures. In both clouds and botnets, we have adversaries, albeit of different characters. In clouds, adversaries are attackers who disrupt and compromise cloud services for fame or for gain. In botnets, adversaries are usually security professionals with the intention of protecting the Internet community at large from the botnet. Intentions and nature aside, similar techniques could be used by malicious attackers for evil as well as by security professionals for good.

Both clouds and botnets need to maintain a known entry. In a botnet, this entry is used in rally for newly infected bots to join the botnet. This could be an IRC channel in an IRC-based botnet, the IP address in a web-service-based botnet, the domain name in a fast-IP-flux-based botnet, the DGA in a fast-domain-flux-based botnet, the repeater/servant/super-node in a hierarchical hybrid botnet [130], and the hard-coded rendezvous or peer-exchanged peer-list in a P2P-based botnet. In a cloud infrastructure, the entry is usually the web-service-based management gateway. Fast-flux-like techniques could be adapted to cloud services to increase the uncertainty for malicious attackers without compromising usability for regular users.

The large number of computational elements (i.e., bots in a botnet, and service instances in a cloud infrastructure) brings forth another similarity. The similarity in configuration between the elements allows an attacker to quickly scale up the attack once a vulnerability is identified and exploited. In modern botnets, polymorphic programming techniques, such as memory randomization, code obfuscation, and encryption, which are traditionally associated with advanced computer viruses, are all used to frustrate security analysts' attempts to understand the malware. On the network level, traditional fixed C&C communication ports are being replaced by dynamically assigned ones. Also, the hard-coded rendezvous points in traditional centralized botnets are supplemented by localized peer-list in recent P2P botnets. Similar measures should be taken in cloud infrastructures for the distributed, static, duplicated, and largely identical configurations to thwart scalable attacks. As long as the agreed resources are guaranteed, the service provider is free to deploy MTD mechanisms, which thwart incremental reconnaissance and vulnerability exploitation by making user-invisible changes to the infrastructure.

Communication protocols and channels are usually the Achilles' heel in both clouds and botnets. In botnets, many security analyses begin with profiling them in terms of communication dynamics. The C&C protocol and structure could be easily identified if such communication is not properly protected, like in early IRC-based botnets and some early MTD botnets such as Torpig. A strong encryption goes a long way in protecting the integrity of the botnet, as demonstrated in the case of Nugache, in which the botnet stays under the radar for a long time, and security researchers have to resort to speculation in its analysis. A lesson for securing cloud infrastructures is that it pays off to encrypt communication, both externally and internally, at every possible level.

A difference between cloud infrastructures and botnets is the ownership. In botnets, with the possible exception of the C&C center, most hosts, including all the bots and most C&C proxies, usually do not belong to the botmasters. Thus, the botmasters adopt techniques to hide the botnet malware from the owners of these hosts. In typical clouds, the numerous computational, storage, and communication elements in the infrastructures all belong to the cloud service providers. The security mechanisms on these elements can be more rich in functionality and aggressive in resource appropriation.

This difference in ownership means that cloud infrastructures could afford better security mechanisms at the cost of increased resource consumption. This also means that the best security practices for botnet designs are not necessarily that for cloud infrastructures. Though we have witnessed, in botnets, a steady transition from centralized architectures to distributed architectures for resilience against crackdowns, the lack of real-time control over the whole infrastructure, inherent in the distributed architectures, is often not acceptable for clouds. Thus, when mapping MTD designs from botnets to cloud infrastructures, we shall carefully consider the impact of these differences on the mapping.

In the next section, we are going to present a few botnet-inspired MTD techniques in detail to illustrate the application of the above lessons to securing cloud infrastructures.

3.3 *Illustrative MTD Techniques for Secure Cloud*

In this section, we present four complementary techniques to adapt lessons from botnets for securing cloud infrastructures. Specifically, these techniques are:

- *Heterogeneous VM Replication* to deliberately introduce diversity to the multiple replications of a client VM instance, to reduce the chances of the same vulnerability being exploited across the replications.
- *Proactive VM Deployment Evolution* to thwart reconnaissance and penetration from attackers by constantly monitoring active client instances for security exposure, and migrating high-risk active VM instances to heterogeneous but compatible low-risk replications to neutralize attackers' reconnaissance and penetration attempts.
- *Agile, or Security-Context-Aware, Opportunistic Migration* to minimize the chances of the migration process being exploited at a lower data/control plane level through randomization.
- *Dynamic Authentication* to seamlessly protect the other techniques from unauthorized access, tampering, theft and poisoning.

4 Secure Cloud Infrastructures

Homogeneity and static configurations have been used for simple cloud administration and management tasks, but attackers have ample opportunities to reconnoiter and penetrate the security perimeter of cloud services. Figures 1 and 2 indicate that security design objectives of cloud and *dark clouds (botnets)* are similar because both clouds face constant threats. Bot creators sometimes use polymorphism to increase computational complexities to identify and remove bots. Polymorphic malicious codes or files are functionally identical but differ from one another in file size, content, or other respects [75]. There are two general types of malicious polymorphism: “(1) Server-side polymorphism, in which a server is configured to serve a slightly different version of a file every time it is accessed, possibly by

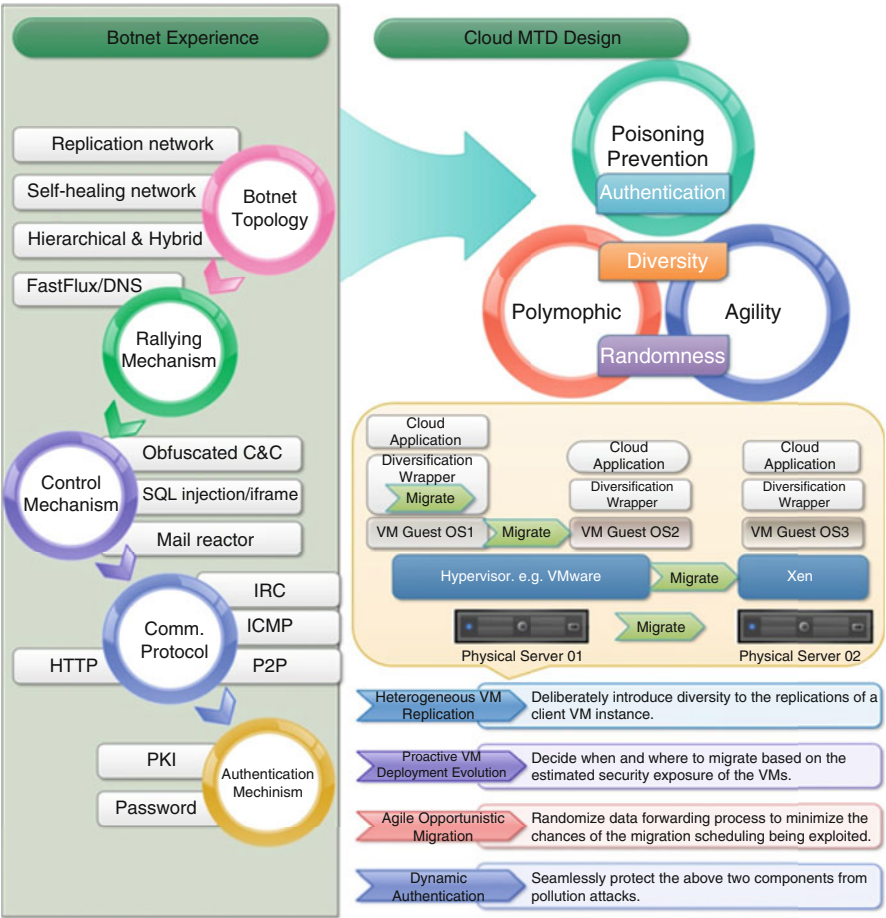


Fig. 2 Cloud moving target defense (MTD)

changing the file name of a component to a new random value, or by encrypting or compressing it in a slightly different way. (2) Malware polymorphism, in which the malware itself is designed to change slightly every time it replicates [75].” Cloud defense mechanisms have not been very effective because botnet technologies have been evolving rapidly using novel moving-target attack methods such as polymorphism and agility to protect botnet operations (see Fig. 1).

Our MTD research aims to develop moving-target defense technologies for secure cloud infrastructures as follows:

- *Polymorphism*: Explore botnet polymorphism techniques, i.e., server-side polymorphism and malware polymorphism [8, 52, 116]. Develop the novel cloud defense polymorphism techniques to protect cloud infrastructures from attackers.
- *Agility*: Investigate botnet agility behaviors. Develop the rapid provisioning technologies of cloud resources to provide high resource availability to cloud customers.
- *Poisoning Prevention*: Probe botnet poisoning mechanisms. Develop the tamper-evident technologies that make unauthorized access to the protected cloud resources easily detected.

In this section, our on-going secure cloud infrastructures research areas, i.e., *heterogeneous VM replication*, *proactive VM deployment evolution*, *Agile opportunistic migration* and *dynamic authentication*, are briefly described to embody three MTD design principles, i.e., diversity, randomization, and authentication in secure cloud infrastructures. Figure 2 depicts the illustrative cloud MTD.

4.1 Heterogeneous VM Replication

To support continuous availability, a logical client VM instance maps to multiple physical VM replications [33, 126]. Only a few replications need be active at a particular time to support the service; the rest are standing by, ready to replace faulty ones when needed. Only active replications will interact with external sources and hence are susceptible to attacks. For ease of administration, the status quo in cloud infrastructures is that replications are homogeneous in terms of the software stack (including the operating system) running on the VM, and the underlying VM hypervisor. The lack of diversity allows the same vulnerability in either the software stack or the underlying hypervisor to be exploited across replications.

Heterogeneous VM replication (HVMR) [65, 126], which allows user application instances to move between heterogeneous VM images, alleviates the problem by deliberately introducing diversity in the replications at four different levels: guest OS configuration, guest OS, hypervisor, and physical machines. For example, user applications can migrate between VM images with different guest OSs managed by the same hypervisor (e.g., Xen, QEMU, Solaris Container, and HP Virtual Partitions), or same guest OSs managed by different hypervisors. HVMRs, when

coupled with *proactive infrastructure deployment evolution* and *agile opportunistic migration* to be discussed later, thwart attacks by changing the profile of the targeted client instance to the disadvantage of attackers.

A major challenge is to support transparent migrations between the heterogeneous replications. Ideally, users should not notice any differences in the heterogeneous physical instances. In reality, it is challenging due to the fact that subtle differences in implementations may result in an application behaving differently. Yet another challenge is to decide on the amount of diversity sufficient to thwart attackers without disturbing clients.

4.2 Proactive VM Deployment Evolution Strategies

Inspired by the dynamic C&C structures in emerging botnets, to mitigate reconnaissance and penetration, *proactive VM deployment evolution* (PVMDE) constantly monitors individual client instances for security exposure, and proactively migrates online instances to a heterogeneous but compatible replication.

4.2.1 Globally Coordinated Evolution Strategy

Similar to centralized C&C structure in a botnet, a central scheduler in the cloud collects information from all active replications. Due to the redundancy of physical replications in supporting a logical client instance, the active replications, called the *backbone*, at a particular time are a subset of all the partitions. All the other replications are standing by, and are beyond the reach of attackers.

The reduced security exposure of the stand-by replications comes at the cost of the active ones. Thus, it is necessary to rotate the backbones to amortize the risks. A straightforward solution is to construct multiple disjointed backbones and let them work alternatively. This can be implemented by, for example, Connected Dominating Set (CDS) [76, 125, 127]. With centralized coordination, backbones can be enumerated and alternately activated to amortize security exposure. However, global coordination requires a control channel to collect status information from replications and send activation/deactivation commands to them. As in botnets with a centralized C&C channel, the control channel is susceptible to attacks.

4.2.2 Locally Coordinated and Attack-Surface-Based Probabilistic Evolution Strategy

The heterogeneous VM replications created for logical client instances have diverse attack surfaces. The *locally coordinated, attack-surface-based, probabilistic evolution strategy* makes use of this in scheduling migrations. Intuitively, a replication

with a small attack surface can be in service longer before being replaced by (i.e., migrating to) other replications.

In this strategy, an active replication will be in service until its accumulated risks, characterized by security exposure, exceed a predefined value. Then, it will make a probabilistic migration decision. In making the migration decision, each potential replacement will be chosen by a probability based on its attack surface: Potential replacements with smaller attack surfaces are more likely to be chosen over those with larger attack surfaces. Quantitatively, for two potential replacement replications a and b , if a has an attack surface half the size of b , a will be twice as likely to be chosen as the replacement than b .

4.2.3 Locally Coordinated and Switching-Based Probabilistic Evolution Strategy

In this strategy, an active replication j (corresponding to a logical client instance i) makes a probabilistic migration decision based on a *switching* probability P_s .

$$P_s = \begin{cases} 1 - \frac{\overline{E_{i,R[j]}}}{E_{i,j}} & \text{if } \overline{E_{i,R[j]}} \leq E_{i,j}, \\ 0 & \text{otherwise} \end{cases},$$

in which $E_{i,j}$ is replication i 's security exposure when the active application is j , and $\overline{E_{i,R[j]}}$ is the average security exposure of all the replacements $R[j]$ of replication j .

4.3 Agile Opportunistic Migration

The aforementioned locally coordinated evolution strategies introduce diversity and randomness in the *logical* migration process. However, since mutually replaceable replications may be managed by different hypervisors or reside in different machines or even geographically distant data centers, the *physical* migration process over the underlying data communication network needs to be protected from disruptive attacks that are prevalent in static networks. For example, an attacker may launch man-in-the-middle attacks on the migration process by logically positioning himself on the migration path using a number of techniques such as ARP spoofing, DNS poisoning, and route hijacking [82, 118]. Another possibility is jamming attacks.

Agile opportunistic migration (AOM) provides the desired protection by randomizing the data forwarding process over the underlying network during migration. By opportunistic, we mean that, instead of setting up a deterministic route from the source replication to the target one (i.e., source routing), the data forwarding decision is made en route. By agile, we mean that the data forwarding process is security-context and replication-status aware. More specifically, based on the

instantaneous disruption level on the link, and replication status (whether the replication is active or standing by), priorities will be assigned to candidate next-hop forwarders; those replications with high priorities are favored as the next-hop data forwarders in a probabilistic forwarding process.

4.3.1 Locally Coordinated Opportunistic Data Forwarding

When a link (i, j) is subject to disruption, the probability $p_{i,j}$ that a migration data packet will successfully pass through that link is a number between 0 and 1. Suppose the candidate next-hop forwarders of i are $(i+1, i+2, \dots, i+k)$. With a probability $p_{i,i+1}$, $i+1$ will receive and hence forward the packet. Otherwise, with a probability $p_{i,i+2} \cdot (1 - p_{i,i+1})$, $i+2$ will receive and hence forward the packet, and so on.

Since more than one candidate next-hop forwarders may receive the data packet, the opportunistic data forwarding process is subject to the following local coordination.

- A node assigns a priority to each candidate next-hop forwarder based on agility.
- The node appends the IDs of the candidate next-hop forwarders to the packet header, and sends the packet out.
- Upon receiving the packet, the candidate next-hop forwarder sets a timer, based on its priority.
- Upon timeout, a candidate next-hop forwarder will forward the packet *if and only if* no other nodes have done so.

A candidate next-hop forwarder that indeed forwards the packet upon timeout will notify all other candidate next-hop forwarders of the forwarding. By the coordination rule, this will reduce the chances of the same packet being transmitted over the network, and hence decreases the chances of the packet being intercepted by attackers.

4.3.2 Adaptive Next-Hop Forwarder Selection

A possible next-hop forwarder selection strategy is to enlist every candidate. A more sophisticated, and potentially more efficient and secure, strategy is to consider:

- Candidates' active/stand-by status at the time of forwarding.
- Current link disruption conditions.
- Progressiveness towards the intended destination.

Only those candidates that will be active at the time of forwarding shall be selected. Whether a candidate will be active at the time of forwarding depends on the chosen PVMDE strategy.

After estimating the candidates' active/stand-by status, a challenge is to choose the number of candidates to forward the data packet, which, ideally, should adapt to the current security context: If the links to the candidates are severely jammed, more

candidates should be enlisted to increase the likelihood of successfully forwarding the data packet; otherwise, fewer candidates shall be enlisted to reduce redundancy and local coordination overhead.

“Progressiveness towards the intended destination” means the chosen next-hop forwarders should be topologically closer to the destination. If not so, a data packet may be stuck in a loop due to opportunistic routing. Existing measures of an intermediate forwarder’s distance to the destination include hop count and the expected transmission count metric (ETX) [29].

4.3.3 Lottery-Like Priority Assignment

Priorities that are assigned to the candidate next-hop forwarders determine the preference of the candidates. To make the forwarding process less susceptible to disruption by increasing attackers’ uncertainty, we can assign priorities by *drawing lotteries*. The winning probability of a candidate i is commensurate with its potential contribution/utility u_i to the migration data forwarding process.

Suppose the set of candidate next-hop forwarders is C . The winning probability of a candidate i is $u_i / \sum_{c \in C} u_c$. After the first candidate is chosen, and if more candidates are needed, the lottery will determine other candidates.

4.4 Dynamic Authentication

Besides disruption, an even more serious attack against replication migration (the *data/control plane*) is unauthorized access (theft) and tampering (poisoning). Theft violates confidentiality and poisoning violates integrity. For example, a confidential document can be stolen if an attacker can read the migrating VM’s file system image, or a backdoor can be planted if the attacker can modify the VM’s memory image. In addition, if an attacker can inject or tamper with control signals, the whole VM migration [104] process can be hijacked. VM migration is also susceptible to theft at the data plane (e.g., passive snooping) and poisoning at the control plane (e.g., incoming/outgoing migration control attack, false resource advertising) [82].

A comparable case comes from botnet design. Although new botnets, especially those with peer-to-peer (P2P) C&C structures, are more dynamic than their predecessors to circumvent a single point of failure at the central C&C host, many of them are susceptible to poisoning, due to inherent design decisions of the underlying P2P protocols on which they are based. Examples: (1) Some P2P botnets allow bots to access or modify the peer index without any authentication; (2) in some botnets, keying materials used to encrypt data and authenticate peers are stored in plaintext on the bots, and the whole mechanism is compromised if the keying materials are obtained by security professionals from a captured bot sample.

Unlike botnets, in clouds, infrastructures are under the control of administrators. Cloud administrators have some unique advantages over botmasters in defending

against theft and poisoning. For example, cloud administrators can implement traditional secret-key-based encryption and authentication mechanisms [17] to protect against outside theft and poisoning threats. However, an insider attack [58] requires more secure key management; mechanisms for doing this are currently being developed by cloud security researchers.

5 Conclusion

Homogeneity and static configurations in cloud infrastructures leave attackers ample opportunities to reconnoiter and penetrate the security perimeter of current cloud services. This chapter has discussed cloud infrastructure security through learning from botnets. Lessons were drawn by studying the evolution of botnets from early static designs to the latest MTD designs that circumvent ever intensifying crackdowns. Challenges to cloud infrastructure security were identified. Illustrative MTD techniques were presented to inspire further research on improving availability, resiliency and data integrity of clouds. Then, secure cloud infrastructures were briefly discussed. Heterogeneous VM replication, proactive VM deployment, agile opportunistic migration, and dynamic authentication technologies will be further developed to shift secure cloud infrastructures, which reduce attackers' understanding of the systems and their ability to launch attacks, while maintaining satisfactory cloud service performance.

Acknowledgements This material is based upon work partially supported by the Northrop Grumman Cybersecurity Research Consortium grant, the Air Force Office of Scientific Research (AFOSR) and the Air Force Research Laboratory (AFRL) Visiting Faculty Research Program (VFRP) extension grant LRIR 11RI01COR.

References

1. Abu Rajab, M., Zarfoss, J., Monrose, F., Terzis, A.: A multifaceted approach to understanding the botnet phenomenon. In: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, IMC'06, New York, pp. 41–52. ACM, New York (2006). doi: 10.1145/1177080.1177086
2. abuse.ch, ZeuS gets more sophisticated using P2P techniques. <http://goo.gl/ugThA> (2011)
3. Antonakakis, M., Demar, J., Elisan, C., Jerrim, J.: damballa.com, DGAs and cyber-criminals: a case study. <http://goo.gl/yDG2C> (2012)
4. Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., Dagon, D.: From throw-away traffic to bots: detecting the rise of dga-based malware. In: Proceedings of the 21st USENIX Conference on Security Symposium, Security'12, Bellevue, pp. 24–24. USENIX Association, Berkeley (2012)
5. Aviv, A.J., Haeberlen, A.: Challenges in experimenting with botnet detection systems. In: Proceedings of the 4th Conference on Cyber Security Experimentation and Test, CSET'11, San Francisco, pp. 6–6. USENIX Association, Berkeley (2011)

6. Baset, S.A., Schulzrinne, H.: An analysis of the skype peer-to-peer internet telephony protocol. In: Proceedings the 25th IEEE International Conference on Computer Communications, INFOCOM'06, Barcelona, pp. 134–146. IEEE, Washington, DC (2006).doi:10.1109/INFOCOM.2006.312
7. Bauer, L., Garriss, S., Reiter, M.K.: Detecting and resolving policy misconfigurations in access-control systems. *ACM Trans. Inf. Syst. Secur.* **14**(1), 2:1–2:28 (2011). doi:10.1145/1952982.1952984
8. Bayoglu, B., Sogukpinar, I.: Polymorphic worm detection using token-pair signatures. In: Proceedings of the 4th International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, SecPerU'08, Sorrento, pp. 7–12. ACM, New York (2008). doi:10.1145/1387329.1387331
9. Beitollahi, H., Deconinck, G.: Review: analyzing well-known countermeasures against distributed denial of service attacks. *Comput. Commun.* **35**(11), 1312–1332 (2012). doi:10.1016/j.comcom.2012.04.008
10. Bhattacharya, J., Vashistha, S.: Utility computing-based framework for e-governance. In: Proceedings of the 2nd International Conference on Theory and Practice of Electronic Governance, ICEGOV'08, Cairo, pp. 303–309. ACM, New York (2008). doi:10.1145/1509096.1509160
11. Binsalleeh, H., Ormerod, T., Boukhtouta, A., Sinha, P., Youssef, A., Debbabi, M., Wang, L.: On the analysis of the Zeus botnet crimeware toolkit. In: Proceedings of 8th Annual International Conference on Privacy Security and Trust, PST'10, Ottawa (2010). doi:10.1109/PST.2010.5593240
12. Boshmaf, Y., Musluhkov, I., Beznosov, K., Ripeanu, M.: The socialbot network: when bots socialize for fame and money. In: Proceedings of the 27th Annual Computer Security Applications Conference, ACSAC'11, Orlando, pp. 93–102. ACM, New York (2011). doi:10.1145/2076732.2076746
13. Boyd, S., Keromytis, A.: SQLrand: preventing SQL injection attacks. In: Proceedings of the 2nd Applied Cryptography and Network Security, ACNS'04, Yellow Mountain, pp. 292–302 (2004)
14. businesswire.com, Amazon Web Services launches “Elastic IPs” – static IPs for dynamic cloud computing
15. Caracas, A., Altmann, J.: A pricing information service for grid computing. In: Proceedings of the 8th ACM/IFIP/USENIX International Middleware Conference: 5th International Workshop on Middleware for Grid Computing, MGC'07, Newport Beach, pp. 4:1–4:6. ACM, New York (2007). doi:10.1145/1376849.1376853
16. Cepe, J.: trendmicro.com, The plot thickens for Zeus-LICAT. <http://goo.gl/roa3j> (2010)
17. Cheng, Y., Agrawal, D.: An improved key distribution mechanism for large-scale hierarchical wireless sensor networks. *Ad Hoc Netw.* **5**(1), 35–48 (2007)
18. Choi, H., Lee, H., Lee, H., Kim, H.: Botnet detection by monitoring group activities in DNS traffic. In: Proceedings of the 7th IEEE International Conference on Computer and Information Technology, CIT'07, Fukushima, University of Aizu, pp. 715–720. IEEE Computer Society, Washington, DC (2007)
19. Comazzetto, A.: sophos.com, Botnets: the dark side of cloud computing. <http://goo.gl/AOaoB>
20. computerweekly.com, Reports of Gumblar's death greatly exaggerated. <http://goo.gl/n41HQ> (2009)
21. confickerworkinggroup.org, Conficker Working Group: lessons learned. <http://goo.gl/bfsPZ> (2011)
22. Cova, M., Kruegel, C., Vigna, G.: Detection and analysis of drive-by-download attacks and malicious javascript code. In: Proceedings of the 19th International Conference on World Wide Web, WWW'10, Raleigh, pp. 281–290. ACM, New York (2010). doi:10.1145/1772690.1772720
23. Dagon, D., Zou, C., Lee, W.: Modeling botnet propagation using time zones. In: Proceedings of the 13th Network and Distributed System Security, NDSS'06, San Diego. USENIX Association, Berkeley (2006)

24. Dainotti, A., King, A., Claffy, K., Papale, F., Pescapè, A.: Analysis of a “/0” stealth scan from a botnet. In: Proceedings of the 2012 ACM Conference on Internet Measurement Conference, IMC’12, Boston, pp. 1–14. ACM, New York (2012). doi:10.1145/2398776.2398778
25. damballa.com, Zeus gets more sophisticated using P2P techniques. <http://goo.gl/MseB7> (2011)
26. damballa.com, DGAs in the hands of cyber-criminals. <http://goo.gl/MseB7> (2012)
27. Danchev, D.: zdnet.com, Facebook phishing campaign serving Zeus crimeware. <http://goo.gl/dn4cb> (2010)
28. Davis, C., Fernandez, J., Neville, S., McHugh, J.: Sybil attacks as a mitigation strategy against the storm botnet. In: Proceedings of the 3rd International Conference on Malicious and Unwanted Software, MALWARE’08, Fairfax. IEEE Computer Society, Washington, DC (2008). doi:10.1109/MALWARE.2008.4690855
29. De Couto, D.S.J., Aguayo, D., Bicket, J., Morris, R.: A high-throughput path metric for multi-hop wireless routing. *Wirel. Netw.* **11**(4), 419–434 (2005). doi:10.1007/s11276-005-1766-z
30. dhs.gov, U.S. Homeland Security Cyber Security R&D Center: Moving Target Defense (MTD) program. <http://goo.gl/XuIUx> (2012)
31. Dittrich, D., Dietrich, S.: P2P as botnet command and control: a deeper insight. In: Proceedings of the 3rd International Conference On Malicious and Unwanted Software, MALWARE’08, Fairfax, pp. 46–63. IEEE, Piscataway (2008)
32. Domnitser, L., Jaleel, A., Loew, J., Abu-Ghazaleh, N., Ponomarev, D.: Non-monopolizable caches: low-complexity mitigation of cache side channel attacks. *ACM Trans. Archit. Code Optim.* **8**(4), 35:1–35:21 (2012). doi:10.1145/2086696.2086714
33. Dong, Y., Chen, Y., Pan, Z., Dai, J., Jiang, Y.: ReNIC: architectural extension to SR-IOV I/O virtualization for efficient replication. *ACM Trans. Archit. Code Optim.* **8**(4), 40:1–40:22 (2012). doi:10.1145/2086696.2086719
34. Falliere, N.: symantec.com, Salty: story of a peer-to-peer viral network. <http://goo.gl/kCfm5> (2011)
35. fbi.gov, Operation: bot roast. <http://goo.gl/FnHZK> (2007)
36. Feily, M., Shahrestani, A., Ramadass, S.: A survey of botnet and botnet detection. In: Proceedings of the 3rd International Conference on Emerging Security Information, Systems and Technologies, SECURWARE’09, Athens, pp. 268–273. IEEE Computer Society, Washington, DC (2009). doi:10.1109/SECURWARE.2009.48
37. Ferguson, R.: trendmicro.eu, The history of the botnet—Part I. <http://goo.gl/nfDHI> (2010)
38. Francia, R.: blorge.com, Storm worm network shrinks to about one-tenth of its former size. <http://goo.gl/Jw8j7> (2007)
39. Gao, H., Hu, J., Wilson, C., Li, Z., Chen, Y., Zhao, B.Y.: Detecting and characterizing social spam campaigns. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC’10, Melbourne, pp. 35–47. ACM, New York (2010). doi:10.1145/1879141.1879147
40. Gaudin, S.: informationweek.com, Storm worm botnet attacks anti-spam firms. <http://goo.gl/OPtVa> (2007)
41. Grizzard, J.B., Sharma, V., Nunnery, C., Kang, B.B., Dagon, D.: Peer-to-peer botnets: overview and case study. In: Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets, HotBots’07, Cambridge, pp. 1–1. USENIX Association, Berkeley (2007)
42. Gu, G., Perdisci, R., Zhang, J., Lee, W.: BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: Proceedings of the 17th Conference on Security Symposium, SS’08, San Jose, pp. 139–154. USENIX Association, Berkeley (2008)
43. Gutmann, P.: The commercial malware industry. In: Proceedings of the 2007 DEFCON Conference, DEFCON’07, Las Vegas (2007)
44. Hachem, N., Mustapha, Y.B., Granadillo, G.G., Debar, H.: Botnets: lifecycle and taxonomy. In: Proceedings of the 2011 Conference on Network and Information Systems Security, SAR-SSI’11, La Rochelle, pp. 1–8. IEEE Computer Society, Washington, DC (2011). doi:10.1109/SAR-SSI.2011.5931395

45. Higgins, K.J.: darkreading.com, New fast-flux botnet unmasked. <http://goo.gl/5CpCu> (2011)
46. Holz, T., Steiner, M., Dahl, F., Biersack, E., Freiling, F.: Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, LEET'08, San Francisco, pp. 9:1–9:9. USENIX Association, Berkeley (2008)
47. honeynet.org, Honeywall. <http://goo.gl/TU4vi>
48. Howard, A., Hu, Y.: An approach for detecting malicious keyloggers. In: Proceedings of the 2012 Information Security Curriculum Development Conference, InfoSecCD'12, Kennesaw, pp. 53–56. ACM, New York (2012). doi:10.1145/2390317.2390326
49. Huang, S.Y., Mao, C.H., Lee, H.M.: Fast-flux service network detection based on spatial snapshot mechanism for delay-free detection. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS'10, Beijing, pp. 101–111. ACM, New York (2010). doi:10.1145/1755688.1755702
50. Huebscher, M.C., McCann, J.A.: A survey of autonomic computing: degrees, models, and applications. *ACM Comput. Surv.* **40**(3), 7:1–7:28 (2008). doi:10.1145/1380584.1380585
51. hyphenet.com, Fake Verizon Wireless bill notification emails lead to malware. <http://goo.gl/PrkaX> (2012)
52. Jabrooth, A.U., Parvathavarthini, B.: Polymorphic worms detection using extended PolyTree. In: Proceedings of the 2nd International Conference on Computational Science, Engineering and Information Technology, CCSEIT'12, Coimbatore, pp. 532–538. ACM, New York (2012). doi:10.1145/2393216.2393305
53. Jackson, D.: secureworks.com, Untorpig. <http://goo.gl/RCfvl> (2008)
54. Jain, P., Sardana, A.: Defending against internet worms using honeyfarm. In: Proceedings of the CUBE International Information Technology Conference, CUBE'12, Pune, pp. 795–800. ACM, New York (2012). doi:10.1145/2381716.2381867
55. Jajodia, S., Ghosh, A.K., Swarup, V., Wang, C., Wang, X.S. (eds.): Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats. *Advances in Information Security*, vol. 54. Springer, New York (2011). doi:10.1007/978-1-4614-0977-9
56. Jajodia, S., Ghosh, A.K., Subrahmanian, V.S., Swarup, V., Wang, C., Wang, X.S. (eds.): Moving Target Defense II: Application of Game Theory and Adversarial Modeling. *Advances in Information Security*, vol. 100. Springer, New York (2012)
57. Kang, B.B., Chan-Tin, E., Lee, C.P., Tyra, J., Kang, H.J., Nunnery, C., Wadler, Z., Sinclair, G., Hopper, N., Dagon, D., Kim, Y.: Towards complete node enumeration in a peer-to-peer botnet. In: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, ASIACCS'09, Sydney, pp. 23–34. ACM, New York (2009). doi:10.1145/1533057.1533064
58. Katz, J., Shin, J.S.: Modeling insider attacks on group key-exchange protocols. In: Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS'05, Alexandria, pp. 180–189. ACM, New York (2005). doi:10.1145/1102120.1102146
59. Kephart, J.O.: Autonomic computing: the first decade. In: Proceedings of the 8th ACM International Conference on Autonomic Computing, ICAC'11, Huddersfield, pp. 1–2. ACM, New York (2011). doi:10.1145/1998582.1998584
60. Lee, S., Kim, J.: Fluxing botnet command and control channels with URL shortening services. *Comput. Commun.* **36**(3), 320–332 (2013). doi:10.1016/j.comcom.2012.10.003
61. Lemos, R.: eweek.com, 'Gameover' financial botnet compromises nearly 700,000 victims. <http://goo.gl/izm6t> (2012)
62. Li, Z., Mohapatra, P.: QoS-aware multicasting in DiffServ domains. *Comput. Commun. Rev.* **34**(5), 47–57 (2004). doi:10.1145/1039111.1039112
63. Li, Z., Goyal, A., Chen, Y., Paxson, V.: Automating analysis of large-scale botnet probing events. In: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, ASIACCS'09, Sydney, pp. 11–22. ACM, New York (2009). doi:10.1145/1533057.1533063

64. Liang, Z., Sekar, R.: Fast and automated generation of attack signatures: a basis for building self-protecting servers. In: Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS'05, Alexandria, pp. 213–222. ACM, New York (2005). doi:10.1145/1102120.1102150
65. Liu, P., Yang, Z., Song, X., Zhou, Y., Chen, H., Zang, B.: Heterogeneous live migration of virtual machines. In: Proceedings of the International Workshop on Virtualization Technology (IWVT), Beijing (2008)
66. Liu, C., Lu, W., Zhang, Z., Liao, P., Cui, X.: A recoverable hybrid C&C botnet. In: Proceedings of the 6th International Conference on Malicious and Unwanted Software, MALWARE'11, Fajardo, pp. 110–118. IEEE Computer Society, Washington, DC (2011). doi:10.1109/MALWARE.2011.6112334
67. Maggio, M., Hoffmann, H., Santambrogio, M.D., Agarwal, A., Leva, A.: Decision making in autonomic computing systems: comparison of approaches and techniques. In: Proceedings of the 8th ACM International Conference on Autonomic Computing, ICAC'11, Karlsruhe, pp. 201–204. ACM, New York (2011). doi:10.1145/1998582.1998629
68. Mather, T., Kumaraswamy, S., Latif, S.: Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance. O'Reilly Media, Sebastopol (2009)
69. Maymounkov, P., Mazières, D.: Kademlia: a peer-to-peer information system/ based on the xor metric. In: Proceedings of the 1st International Workshop on Peer-to-Peer Systems, Cambridge, pp. 53–65 (2002)
70. mcafee.com, W32/Akbot. <http://goo.gl/cbrRC> (2006)
71. McCarty, B.: Botnets: big and bigger. IEEE Secur. Privacy **1**(4), 87–90 (2003). doi:10.1109/MSECP.2003.1219079
72. Mendonça, L., Santos, H.: Botnets: a heuristic-based detection framework. In: Proceedings of the Fifth International Conference on Security of Information and Networks, SIN'12, Jaipur, pp. 33–40. ACM, New York (2012). doi:10.1145/2388576.2388580
73. Mercuri, R.T.: Scoping identity theft. Commun. ACM **49**(5), 17–21 (2006). doi:10.1145/1125944.1125961
74. microsoft.com, Microsoft Security Bulletin MS04-011. <http://goo.gl/DP4QB> (2004)
75. microsoft.com, How Does Botnets Work? <http://goo.gl/UYGQ1> (2009)
76. Misra, R., Mandal, C.: Rotation of cds via connected domatic partition in Ad Hoc sensor networks. IEEE Trans. Mob. Comput. **8**(4), 488–499 (2009). doi:10.1109/TMC.2008.128
77. Moscaritolo, A.: scmagazine.com, Zeus spreading through drive-by download. <http://goo.gl/KJ4y8> (2009)
78. Mrozek, T.: justice.gov, Wyoming man charged with infecting thousands of computers with 'trojan' that he used to commit fraud. <http://goo.gl/G6wtW> (2008)
79. Mushtaq, A.: fireeye.com, Killing the beast – part 5. <http://goo.gl/mtDH7> (2012)
80. Nazario, J.: arbornetworks.com, Nugache: TCP port 8 bot. <http://goo.gl/FqF6D> (2006)
81. Nunnery, C., Sinclair, G., Kang, B.B.: Tumbling down the rabbit hole: exploring the idiosyncrasies of botmaster systems in a multi-tier botnet infrastructure. In: Proceedings of the 3rd USENIX Conference on Large-Scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More, LEET'10, San Jose, pp. 1–1. USENIX Association, Berkeley (2010)
82. Oberheide, J., Cooke, E., Jahanian, F.: Empirical exploitation of live virtual machine migration. In: Proceedings of the 2008 Blackhat Conference, BLACKHAT'08, Las Vegas (2008)
83. Palmieri, F., Fiore, U.: Enhanced security strategies for MPLS signaling. J. Netw. **2**(5), 1–13 (2007). doi:10.4304/jnw.2.5.1-13
84. Pang, W.L., Chieng, D., Ahmad, N.N.: A practical layer 3 admission control and adaptive scheduling (13-acas) for cots wlans. Wirel. Pers. Commun. **63**(3), 655–674 (2012). doi:10.1007/s11277-010-0157-7
85. Park, Y., Reeves, D.S.: Identification of bot commands by run-time execution monitoring. In: Proceedings of the 2009 Annual Computer Security Applications Conference, ACSAC'09, Honolulu, pp. 321–330. IEEE Computer Society, Washington, DC (2009). doi:10.1109/ACSAC.2009.37

86. Pathak, A., Qian, F., Hu, Y.C., Mao, Z.M., Ranjan, S.: Botnet spam campaigns can be long lasting: evidence, implications, and analysis. In: Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems, SIGMETRICS'09, Seattle, pp.13–24. ACM, New York (2009). doi:10.1145/1555349.1555352
87. Paul, R.: arstechnica.com, Researchers track Ron Paul spam back to Reactor botnet. <http://goo.gl/Qgk5Q> (2007)
88. Pitsillidis, A., Kanich, C., Voelker, G.M., Levchenko, K., Savage, S.: Taster's choice: a comparative analysis of spam feeds. In: Proceedings of the 2012 ACM Conference on Internet Measurement Conference, IMC'12, Boston, pp. 427–440. ACM, New York (2012). doi:10.1145/2398776.2398821
89. Porras, P., Saïdi, H., Yegneswaran, V.: A foray into Conficker's logic and rendezvous points. In: Proceedings of the 2nd USENIX Conference on Large-Scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More, LEET'09, Boston, pp. 7–7. USENIX Association, Berkeley (2009)
90. Provos, N., Holz, T.: Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison-Wesley, Boston (2007)
91. Provos, N., Mavrommatis, P., Rajab, M.A., Monrose, F.: All your iFRAMEs point to us. In: Proceedings of the 17th Conference on Security Symposium, SS'08, San Jose, pp. 1–15. USENIX Association, Berkeley (2008)
92. Provos, N., Rajab, M.A., Mavrommatis, P.: Cybercrime 2.0: when the cloud turns dark. Queue 7(2), 46–47 (2009). doi:10.1145/1515964.1517412
93. Rajab, M.A., Zarfoss, J., Monrose, F., Terzis, A.: My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging. In: Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets, HotBots'07, Cambridge, pp. 5–5. USENIX Association, Berkeley (2007)
94. Ramachandran, A., Feamster, N.: Understanding the network-level behavior of spammers. ACM SIGCOMM Comput. Commun. Rev. 36(4), 291–302 (2006). doi:10.1145/1151659.1159947
95. Rekhter, Y., Karrenberg, D., Groot, G., Moskowitz, B.: ietf.org, RFC 1918: address allocation for private internets. <http://goo.gl/qTuQN> (1996)
96. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS'09, Chicago, pp. 199–212. ACM, New York (2009). doi:10.1145/1653662.1653687
97. Rouiller, S.: askapache.com, Virtual LAN security: weaknesses and countermeasures. <http://goo.gl/wrCZf> (2006)
98. Sanchez, F., Duan, Z.: Region-based BGP announcement filtering for improved BGP security. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS'10, Beijing, pp. 89–100. ACM, New York (2010). doi:10.1145/1755688.1755701
99. Schneider, D.: Fresh phish. IEEE Spectr. 45(10), 34–38 (2008). doi:10.1109/MSPEC.2008.4635052
100. securelist.com, TDL4: top bot. <http://goo.gl/23BaA> (2011)
101. Sheldon, F.T., Vishik, C.: Moving toward trustworthy systems: R&d essentials. Computer 43(9), 31–40 (2010). doi:10.1109/MC.2010.261
102. Sinclair, G., Nunnery, C., Kang, B.: The Waledac protocol: the how and why. In: Proceedings of the 4th International Conference on Malicious and Unwanted Software, MALWARE'09, Montreal, pp. 69–77. IEEE Computer Society, Washington, DC (2009). doi:10.1109/MALWARE.2009.5403015
103. Song, C., Zhuge, J., Han, X., Ye, Z.: Preventing drive-by download via inter-module communication monitoring. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS'10, Beijing, pp. 124–134. ACM, New York (2010). doi:10.1145/1755688.1755705

104. Srinivasan, K., Yuuw, S., Adelmeyer, T.J.: Dynamic VM migration: assessing its risks & rewards using a benchmark. *ACM SIGSOFT Softw. Eng. Notes* **36**(5), 317–322 (2011). doi:10.1145/1958746.1958791
105. Srivatsa, M., Iyengar, A., Yin, J., Liu, L.: Mitigating application-level denial of service attacks on web servers: a client-transparent approach. *ACM Trans. Web* **2**(3), 15:1–15:49 (2008). doi:10.1145/1377488.1377489
106. Stone, B.: nytimes.com, Pakistan cuts access to YouTube worldwide. <http://goo.gl/qG0Hn> (2008)
107. Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., Kruegel, C., Vigna, G.: Your botnet is my botnet: analysis of a botnet takeover. In: *Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS'09*, Chicago, pp. 635–647. ACM, New York (2009). doi:10.1145/1653662.1653738
108. Stone-Gross, B., Holz, T., Stringhini, G., Vigna, G.: The underground economy of spam: a botmaster's perspective of coordinating large-scale spam campaigns. In: *Proceedings of the 4th USENIX Conference on Large-Scale Exploits and Emergent Threats, LEET'11*, Boston, pp. 4–4. USENIX Association, Berkeley (2011)
109. Stover, S., Dittrich, D., Hernandez, J., Dietrich, S.: Analysis of the storm and nugache trojans: P2p is here. *Login Issue* **32**(6), 18–27 (2007)
110. symantec.com, W32.Stration. <http://goo.gl/RZl3e> (2007)
111. symantec.com, Trojan.Srizbi. <http://goo.gl/nOExB> (2007)
112. symantec.com, Gumbler. <http://goo.gl/GV3m0> (2009)
113. symantec.com, Backdoor.Tidserv. <http://goo.gl/Z4B1Z> (2012)
114. Thonnard, O., Dacier, M.: A strategic analysis of spam botnets operations. In: *Proceedings of the 8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference, CEAS'11*, Perth, pp. 162–171. ACM, New York (2011). doi:10.1145/2030376.2030395
115. Tung, L.: zdnet.co.uk, Storm worm: more powerful than Blue Gene. <http://goo.gl/4zNr9> (2007)
116. Van Gundy, M., Balzarotti, D., Vigna, G.: Catch me, if you can: evading network signatures with web-based polymorphic worms. In: *Proceedings of the 1st USENIX Workshop on Offensive Technologies, WOOT'07*, Boston, pp. 7:1–7:9. USENIX Association, Berkeley (2007)
117. Vijayan, J.: computerworld.com, U.K. Web hoster, customers scramble after attack deletes 100,000 sites. <http://goo.gl/fMfyE> (2009)
118. Wählisch, M., Maennel, O., Schmidt, T.C.: Towards detecting bgp route hijacking using the rpki. In: *Proceedings of the 2012 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM'12*, Helsinki, pp. 103–104. ACM, New York (2012). doi:10.1145/2342356.2342381
119. Wang, L., Li, Z., Chen, Y., Fu, Z., Li, X.: Thwarting zero-day polymorphic worms with network-level length-based signature generation. *IEEE/ACM Trans. Netw.* **18**(1), 53–66 (2010). doi:10.1109/TNET.2009.2020431
120. Wang, P., Aslam, B., Zou, C.C.: Peer-to-peer botnets, Chap. 18. In: Stavroulakis, P., Stamp, M. (eds.) *Handbook of Information and Communication Security*, pp. 335–350. Springer, Heidelberg (2010)
121. Wang, P., Sparks, S., Zou, C.C.: An advanced hybrid peer-to-peer botnet. *IEEE Trans. Dependable Secure Comput.* **7**(2), 113–127 (2010). doi:10.1109/TDSC.2008.35
122. Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G., Osipkov, I.: Spamming botnets: signatures and characteristics. *SIGCOMM Comput. Commun. Rev.* **38**(4), 171–182 (2008). doi:10.1145/1402946.1402979
123. Yadav, S., Reddy, A.K.K., Reddy, A.N., Ranjan, S.: Detecting algorithmically generated malicious domain names. In: *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC'10*, Melbourne, pp. 48–61. ACM, New York (2010). doi:10.1145/1879141.1879148
124. Yan, G., Chen, G., Eidenbenz, S., Li, N.: Malware propagation in online social networks: nature, dynamics, and defense implications. In: *Proceedings of the 6th ACM Sympo-*

- sium on Information, Computer and Communications Security, ASIACCS'11, Hong Kong, pp. 196–206. ACM, New York (2011). doi:10.1145/1966913.1966939
125. Yang, S., Wu, J.: Efficient broadcasting using network coding and directional antennas in MANETs. *IEEE Trans. Parallel Distrib. Syst.* **21**(2), 148–161 (2010). doi:10.1109/TPDS.2009.44
 126. Ye, K., Jiang, X., Ma, R., Yan, F.: Vc-migration: live migration of virtual clusters in the cloud. In: *Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing, GRID'12*, Beijing, pp. 209–218. IEEE Computer Society, Washington, DC (2012). doi:10.1109/Grid.2012.27
 127. Yu, J., Wang, N., Wang, G., Yu, D.: Review: connected dominating sets in wireless ad hoc and sensor networks – a comprehensive survey. *Comput. Commun.* **36**(2), 121–134 (2013). doi:10.1016/j.comcom.2012.10.005
 128. Zhang, Z., Zhang, Y., Hu, Y.C., Mao, Z.M.: Practical defenses against BGP prefix hijacking. In: *Proceedings of the 2007 ACM CoNEXT Conference, CoNEXT'07*, New York, pp. 3:1–3:12. ACM, New York (2007). doi:10.1145/1364654.1364658
 129. Zhang, L., Yu, S., Wu, D., Watters, P.: A survey on latest botnet attack and defense. In: *Proceedings of the 10th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TRUSTCOM'11*, Changsha, pp. 53–60. IEEE Computer Society, Washington, DC (2011). doi:10.1109/TrustCom.2011.11
 130. Zhang, Z., Lu, B., Liao, P., Liu, C., Cui, X.: A hierarchical hybrid structure for botnet control and command. In: *Proceedings of the 2011 IEEE International Conference on Computer Science and Automation Engineering, CSAE'11*, Shanghai, pp. 483–489. IEEE Computer Society Press, Washington, DC (2011). doi:10.1109/CSAE.2011.5953266
 131. Zhang, R., Huang, S., Qi, Z., Guan, H.: Static program analysis assisted dynamic taint tracking for software vulnerability discovery. *Comput. Math. Appl.* **63**(2), 469–480 (2012). doi:10.1016/j.camwa.2011.08.001
 132. Zhu, Z., Lu, G., Chen, Y., Fu, Z.J., Roberts, P., Han, K.: Botnet research survey. In: *Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference, COMPSAC'08*, Turku, pp. 967–972. IEEE Computer Society, Washington, DC (2008). doi:10.1109/COMPSAC.2008.205
 133. Zhuge, J., Holz, T., Han, X., Guo, J., Zou, W.: Characterizing the IRC-based botnet phenomenon. Technical report, Universität Mannheim/Institut für Informatik (2007)

High Performance Cloud Auditing and Applications

Han, K.J.; Choi, B.-Y.; Song, S. (Eds.)

2014, XXIV, 360 p. 89 illus., 62 illus. in color., Hardcover

ISBN: 978-1-4614-3295-1