Proactive Botnet Countermeasures – An Offensive Approache

Felix LEDER, Tillmann WERNER, and Peter MARTINI
Institute of Computer Science IV, University of Bonn, Germany

Abstract. Botnets, consisting of thousands of interconnected remote-controlled computers, pose a big threat against the Internet. We have witnessed the involvement of such malicious infrastructures in politically motivated attacks in more than once recent years. Classical countermeasures are mostly reactive and conducted as part of incident response actions. This is often not sufficient. We argue that proactive measures are necessary to mitigate the threat posed by botnets and demonstrate techniques based on a formalized view of botnet infrastructures. However, while being technically feasible, such actions raise legal and ethical questions. We would like to initiate a more acute discussion and point out the important aspects.

Keywords. Cyberwar, botnets, DDoS, defence strategies, countermeasures

Introduction

A botnet is an alliance of interconnected computers infected with some malicious software (a bot). Bots are commanded by an operator and can typically be advised to send Spam mails, harvest information such as license keys or banking data on compromised machines, or launch distributed denial-of-service (DDoS) attacks against arbitrary targets. What's more, they often interfere with regular operation rendering infected machines unstable or unusable. Thousands of such botnets exists with each containing thousands to millions of infected systems. The result are major direct and indirect consequences for economy as well as for the political life [2].

In the past, the economic damage caused by botnets has been related to bandwidth and CPU resources bound by Spam, DDoS attacks, and botnet propagation. More recent reports show that the damage is largely increasing due to the number of stolen credit card information and banking credentials [17]. As more and more botnets are incorporating functionality to collect this data, the damage will increase over the next years. Besides this, distributed denial-of-service attacks originating from botnets disrupt business at attacked sites. The measures for handling these attacks, like forensic analysis, moving sites into different networks, data recovery etc., cost up to several million US dollars per incident, let alone the collateral damage which is hard to measure [23].

Recent developments show that botnets are not only critical for companies and consumers but also show political motives. Largely organized DDoS attacks conducted by botnets in 2007 and 2008 cut off major Government sites in Eastern Europe from the rest of the Internet. This drastically shows how the vast number of remotely controlled machines has the potential to be used as a powerful weapon in a cyberwar

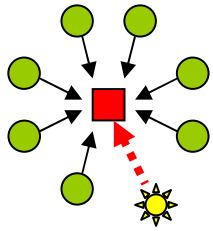
rather than just being an annoying phenomenon affecting only some individuals. The spreading of botnets is conducted actively: Remote systems are automatically attacked and exploited, mails are sent that trick the reader into opening malicious programs or web pages which actively exploit the visiting computer. On the contrary side, measures against botnets are very often passive and defensive. To date, active measures are often taken in the context of responses to an ongoing incident only. We have developed methodologies and prototypes for infiltrating botnets that can be used to tackle them from the inside. Such offensive countermeasures can be used to mitigate or extinguish existing botnets. We present our different approaches and demonstrate how they can be applied to existing botnets in case studies.

The remainder of this paper is organized as follows: The next section presents a brief overview of common botnet topologies. Section 2 reviews classical countermeasures. Proactive approaches will be explained in section 3. In section 4 some case studies will be presented. We will discuss legal and ethical aspects in section 5. Section 6 concludes the paper.

1. Botnet Topologies

The two things needed to setup a botnet are an addressing mechanism to identify and reach a command-and-control instance, and a communication protocol to distribute commands to the bots. The latter is often referred to as an *overlay network* that forms the botnet's communication channel. Different botnets are using different strategies here which is reflected in the topology used: We differentiate between *centralized*, *decentralized* and *locomotive* botnets. The kind of topology is extremely important for the selection of containment strategies.

Centralized topologies as depicted in figure ?? are the classical botnet structures. Examples are the IRC-based *Agobot*, *Rbot*, and *Sdbot* families [1]. A static command-and control server is contacted by bots via its IP address (which generally requires resolving a DNS name first). Centralized botnet infrastructures often rely on existing network protocols on top of IP that implement standard client-server architectures, like IRC or HTTP. For this reason, they are obviously completely extinguishable by taking down their C&C server.



Centralized Botnet

The communication in a centralized botnet can either follow a *push strategie* (as e.g., in IRC-based communication) where each bot stays connected to a server which then distributes commands simultaneously to all hosts in a broadcast-like manner. Or

the server has to be *polled* by the clients on a regular basis (as in HTTP-based botnets). In the latter scenario, the general method is to set up and update a central resource like a web page which can be browsed by the bots. Both approaches have their advantages, e.g., IRC botnets can be built upon an existing IRC infrastructure with multiple self-synchronizing servers, providing load-balancing and reliability. HTTP, on the other hand, is more stealthy and is better suited for bypassing security gateways and hiding amongst regular traffic patterns.

In a d*ecentralized topology*, no single command-and-control component exists. Instead, each bot seeks for a commander using some upstream query mechanism. Well-known representatives are the *Storm Worm* [3], or *Conficker*. The two-tiered approach permits the botnet owner to easily change the C&C backbone, making it much harder to take it down. As in centralized botnets, commands can be pushed to bots, which requires that they can be reached instantly, or infected machines pull commands from their individual C&C server (the latter being the most common case). Bots can be implemented to automatically re-establish a C&C session on disconnects. Most decentralized botnets seen so far were based on peer-to-peer (P2P) technology that allows for both information queries as well as host addressing, the two features needed for the communication between a bot and a command server. In a general P2P botnet some peers are controlled by the botnet owner and used to issue and propagate information (i.e. commands) to other peers. Taking advantage of the flexible self-organizing network infrastructure, these nodes are easily replaceable with other hosts.

The decentralization can be taken even further by designing fluxy registration of C&C servers at the query layer (i.e., a pool of command servers returned to queries which is kept highly dynamic through automated subscriptions). In most cases these C&C servers are also infected hosts, temporarily playing the role of a commander. Another way would be to change the query interface, e.g., by choosing time-dependent domain names. We call such botnets *locomotive* because of their constantly moving structure. One example is the HTTP-driven *Torpig* botnet [4]. *Conficker*, in addition to its P2P structure, also makes use of constantly changing DNS names [5-7]. There is no standard implementation of such botnets. In fact, the overall structure is often even more complex than outlined above. We will see different examples in section 4.

In reality, the boundaries between centralized, decentralized and locomotive botnets are blurred: A similar strategy was already commonly implemented in classical botnet infrastructures where a DNS entry was used to transparently switch between servers. However, this does not really provide more security as it only displaces the single point against which takeover attempts could be mounted.

2. Classical Countermeasures

Traditional ways of counter-measuring botnets is generally restricted to spotting a central weak point in their infrastructure that can be manipulated, disrupted or blocked. The most common way is to cooperate with an Internet service provider to gain access and shut down the central component, resulting in a loss of control for the botnet owner: The botnet cannot be commanded anymore. Such actions are often performed during emergency response to an ongoing incident like a DDoS attack. While this course of action has proven effective (e.g., shutting down an IRC-based C&C server prevents bots from receiving commands, and machines already involved in an attack

are rebooted sooner or later), it requires access to the machine, and, most notably, the willingness to cooperate at the responsible institution.

Classical countermeasures against botnets have three different points to attack:

1. The command and control (C&C) server
2. The botnet traffic
3. The infected computers

We will explain the countermeasures with their chances and difficulties in the following. Our goal is to show their differences and why we need more discussion about offensives approaches against botnets.

2.1. Taking Down the C&C Server

The most promising approach is to remove the base of a botnet, which is the C&C server. Pulling the plug of the command-and-control host allows to extinguish the whole botnet in one go. Unfortunately this is only possible if all of the following conditions are met:

1. The botnet uses a centralized structure
2. The location of the C&C server is known
3. The provider cooperates

If only one of those conditions is not met, the C&C server cannot be removed. More and more especially newer botnets are not solely relying on a centralized structure anymore. Instead they use peer-to-peer (P2P) functionality or multi-proxy structures to hide their central origin. Thus, it is hardly possible to find the location of the C&C server. If multiple, fixed servers are used, all of them must be removed. When the location is known, the provider hosting the C&C server must be cooperative. Very often, botnets are controlled from locations hosted by so-called *bullet-proof hosters*, that are not responsive to abuse requests or, even worse, move the server to affiliated partners as soon as the pressure to take the host down rises. Those providers are found in almost all countries, Germay and the U.S. being amongst the most prominent ones. Law enforcement is often one step behind when the hosted services are suddenly switched to an affiliated provider. In the lucky situation of the location of a centralized botnet to be within a cooperative provider's network, the provider must also be notified. Different organizations that track attacks in the Internet receive so many hints about possible C&C servers that they cannot handle, follow, and verify actions against each C&C individually. The number of conditions is part of the problem that such a large number of botnets exist, and it is still increasing.

Taking the C&C servers down is not always similar to removing the root of the botnet. Infected machines can also contain functionality to spread autonomously as well as other fall-back logic that gets executed in case the C&C cannot be reached. This creates additional traffic and can lead to more infected machines.

Some cases are known where a botnet takeover was performed with the goal to issue commands that make the bots stop an attack or deinstall themselves. While this approach is more delicate with respect to responsibility for effects caused on infected machines, it is extremely successful at the same time. Attacks are stopped immediately

and the botnet is eventually shut down conclusively without the chance to be brought back up by the owner. However, when rating the chance of such actions being successful, in most cases depends again on whether cooperation with the responsible infrastructure providers is possible or not.

2.2. Sinkholing Malicious Traffic

If the C&C server cannot be taken down, a second option is to redirect malicious traffic to so called *sinkholes*, a strategy that found its way into recent mitigation techniques. Sinkholes record malicious traffic, analyze it and afterwards drop it such that it cannot reach the original target it is meant for. One example of sinkholes is *DDoS null-routing*. In case traffic belonging to an ongoing DDoS attempt is observed it is dropped and sometimes counted for later analysis. DdoS null-routing at border-routers is a promising approach to mitigate DDoS attacks but comes with the challenges of reliable identification of attack related traffic and clean dissection of high-bandwith data streams at an early stage. This is generally only possible at ISP level. A collaborative worldwide initiative between providers would be another option, but is obviously beyond all question.

2.3. Cleaning Infected Systems

Th/e most sustainable countermeasure against botnets is to clean all infected systems and remove the bots installed. While this removes the full power of a botnet, it is also the most complex and most difficult to manage countermeasure To date, the owners or administrators are responsible for keeping their systems clean from infections. Only recommendations and technical advice can be given to them. As most users are not even aware of their machine being infected, let alone the ability to remove a malware, a global cleaning is impossible. The huge media campaigns about Conficker and the number of still infected systems show that even with intense warnings a client-side cleanup performed by the owners is not feasible.

The standard recommendation to keep systems safe from botnets is to use firewalls and up-to-date anti-virus (AV) software. Firewalls are a preventive feature that in many cases only block attacks from the outside. The increasing number of drive-by-exploits, using bugs in the user's browser to infect a system, and the mobility of malicious data on laptops or USB-sticks opens up a range of new infection vectors that bypass firewalls. This development has been very obvious with Conficker infections. Anti-virus software is a reactive feature. Before it is able to detect anything, signatures have to be available and the malicious data has to be on the targeted computer. If signatures do not (yet) exist at that point, the systems cannot be defended. Tests of different AV engines have shown that some detection rates are as low as 80% [18]. Once a system is infected, the bot can spread and perform malicious actions until AV signatures are available and can be used. Often AV engines are outdated and not updated on a regular basis. Furthermore, different bots disable AV scanners or hide in ways that cannot be detected by regular scanners [19].

All in all, a global cleanup, as would be required in order to effectively take power from botnets, seems to be infeasible.

2.4. Conclusion on Classical Strategies

The observations discussed in this section demonstrate that to date the success rate of botnet countermeasures depends mainly on organizational and political general conditions. Given that setting up cooperation or diplomatic agreements takes time we come to the conclusion that establishing an appropriate relationship that legitimates cooperation for collaborated actions is not suited as an ad-hoc scheme for fighting ongoing attacks. Even if such relationships exist – the affected site would generally not be involved.

The situation gets worse considering that modern botnet infrastructures do not fall under responsibility of one entity. Instead, distributed peer-to-peer networks operate globally, thus shutting down local parts (often no more than single machines) would be no effective solution. All in all, countermeasures that require close cooperation are today generally infeasible for both technical and political reasons.

There have been discussions in the past where experts stated that shutting down C&C servers has become useless as they would be replaced with new, better protected systems almost immediately. This accelerated arm's race would eventually lead to sophisticated botnet technology sooner than without mitigation. We think that this view is by no means bearable: It ignores the fact that botnets cause harm against other organizations. A hands-off approach leaves potential target sites alone with the existing threat. In the end, restricting mitigation techniques to eluding from or block ongoing attacks is an admission of powerlessness. We propose a combination of classical techniques with additional proactive strategies which we discuss in the following section.

3. Proactive Measures

The classical countermeasures are very good steps to mitigate the power of botnets but recent developments show that they are only applicable to a certain extent. Newer botnets use more sophisticated obfuscation techniques that deny the use of classical countermeasures due to the difficulties explained in the previous section. While the newer structures introduced by recent botnets complicate the applicability of classical countermeasures, they are open to more offensive counter-tactics. This section explains general principles that can be exploited to create offensive countermeasures against botnets. This section focuses on the technical possibilities.

Exploring the structure of a botnet is often the first step for finding starting points for possible countermeasures. An immanent property of all botnets is that they have to allow new machines, which run on untrusted platforms, to join the network. This is an important aspect for countermeasure approaches: We are not restricted to acting from the outside – we can join the network, perform investigations while being part of the infrastructure ourselves and might even be able to contain the botnet or take it down from the inside. Furthermore, bots are spreading to infect more systems and make the network grow. Malware samples, which are not hard to obtain, can be analyzed (i.e., reverse-engineered) to learn about their internals. With the knowledge about a bot's functionality, it is often possible to create a *fake bot* that links itself into the botnet to monitor or perturb the internal communication. This procedure is always possible, as all information about the initial bootstrapping has to be included in the malware binary and can thus be cloned. Many approaches presented in this section rely on the

infiltration of botnets, a technique that was discussed in different flavors in case studies before [3-5,9].

Offensive strategies can be split into three different categories: *Mitigation*, *manipulation*, and *exploitation*. The extent to which corresponding actions are possible depends largely on the topology used by botnet. Especially, decentralized and moving topologies offer multiple chances for countermeasures.

Strategies for *mitigation* are offensive, technical means that slow botnets down by consuming resources. Examples can be temporary DoS attempts against C&C servers, binding connections from infected machines, or blocking of malicious domains. *Manipulation* strategies make use of the command layer. The knowledge about command protocols is essential to manipulate and inject commands. The required knowledge about the protocols includes cryptography used. Even though cryptography may completely deny the inspection of botnet data exchange, our Waledac case study (c.f. section 4) shows how this can be achieved even when cryptographic methods like RSA and AES are used. Possible manipulation can be the alteration or removal of DDoS or Spam commands as well as commands to download and execute programs, which allows a remote cleanup of infected machine. Less invasive options than to execute programs on the remote machines may be to drop collected personal data, like credit card or banking details, to replace them by fake information, or to command to stop the collection. Lastly, *exploitation* is a special strategy that makes use of bugs found in bots. Like bugs in other products, these can be used to perform actions on the infected machines. Even though, this category is the most powerful, it is the one with the highest risk involved because exploits can easily crash and damage systems.

Not every strategy can be applied to every botnet. Some of them depend large on the botnet's topology. Especially non-centralized botnets offer a range of possibilities. We will explain different technical possibilities in the following.

3.1. The Addressing Layer

In this paragraph we discuss strategies targeting the routing and the addressing layer of a botnet infrastructure. It is important to understand that the routing mechanism used by a botnet is needed for addressing hosts, or C&C servers respectively. The command layer, in contrary, works on top of the addressing scheme to provide a communication overlay network to the interconnected machines.

The most common way for a bot to address a central C&C server is a DNS name that resolves to an IP address – the addressing takes place in two phases. Each phase makes a potential starting point for intervention. For instance, DNS requests are generally handled by a local DNS resolver which, in turn, forwards the request to an authoritative DNS server. This local resolver is controlled by the site administrator and can easily be instructed to return a specially crafted response to specific queries. The same holds for IP routing: Local routers can be equipped with routing table entries to sinkhole certain addresses or redirect them to different hosts (*sinkholing* is the term for redirecting connection attempts to a special purpose server to identify infected machines). As a consequence, both steps result in bots in the local network being unable to contact the original C&C server and might even be controlled by a pseudo-server. An intervention as described above always requires a man-in-the-middle position. However, it is not always necessary to change the configuration of inline devices. Approaches exist that demonstrate the live modification of relevant network traffic [11].

Modern botnets use more complex addressing schemes which are also run as an overlay network on top of the IP-based Internet. Examples are peer-to-peer networks like *Storm* or *Waledac*. They provide their own addressing scheme with the goal to increase flexibility and decentralization. Both examples will be discussed in more detail in section 4. Again, a strategic position is necessary to infiltrate the addressing layer of these botnets. A general approach is to inject a carefully monitored and controlled node which is ideally a clone of an original peer.

Even when C&C servers cannot be physically accessed, they must reachable over the Internet because bots have to contact them for receiving commands. This can be used to mitigate the botnet by creating a DoS situation to the server. A controlled allied DDoS from would make the server unreachable. Additionally, botnets often rely on technology that is prone to specific attacks by design, like the Transmission Control Protocol (TCP). For example, a C&C server's TCP backlog queue can be filled up with connection attempts to trigger denial-of-service conditions, turning the botnet's weapons against itself. This is especially useful for most HTTP based bots where new connections are established for every command request. We have evaluated different service and operating system combinations and found a temporary TCP DoS attack to be easily conductible with only very few resources. During our research we have been able to reliably decrease the probability to establish connections to TCP servers to less than 5% with only one offensive machine. A single host can keep the victim service's backlog queue filled, blocking all further connection attempts and thus hindering bots from requesting commands. Such an action can be crafted in a way that it is not possible to tell apart the connection attempts from the ones issued by bots. As a result, any counter-action intending to block the requests would also block all “legitimate” bots. Our tests showed that one single machine can keep a TCP service permanently unresponsive just by initiating and completing 3-way handshakes and keeping connections up as long as possible. Such an attack results in less bots being able to contact the C&C server and participate in malicious activity.

Flooding the link or network where the C&C server resides with packets that consume all available bandwidth is another similar attack. It obviously requires more resources, though, as more packets must be sent. A reflection attack can be used to amplify the amount of traffic sent, however, that would incorporate third-party resources and probably permission by the affected site owners which is apparently not granted.

3.2. The Command Layer

Attacking the command layer of a botnet requires knowledge of the protocol used. An easy example would be an IRC-based network where a *remove command* instructs bots to uninstall themselves from infected systems. Many classical bots implement such an instruction [1]. The injection of a command requires either control over the C&C server, or bots have to be redirected to a different server by performing an attack against the addressing layer (as described in the previous paragraph), which then distributes the removal instruction. Other bots do not have an uninstall option but offer an update functionality that can be used to replace the malware with a innocuous binary or a program that scans for and eventually removes the bot (similar to a virus scanner).

In combination with infiltrating the addressing layer, other approaches become feasible: Original commands can be monitored, intercepted and modified. A protocol

could implement checks to render such manipulations impossible, however, such measures were not seen in botnets so far.

In general, to actually conduct a botnet infiltration attack, a combination of actions on both the addressing and the command layer is necessary. Redirecting bots to a controlled server either for sinkholing or to command them to perform a self-removal is probably one of the most effective countermeasures on the infrastructure level.

3.3. Exploitation

Exploit based strategies make use of the fact, that even botnets contain bugs and programming flaws, resulting in vulnerabilities that can be exploited to gain control either over a central component (like a C&C server) or over bot-infected machines. Such vulnerabilities may range from misconfiguration, e.g., an insecure IRC server setup that allows other users to control a channel, to security holes in software, like remotely-exploitable buffer overflows.

Mitigation and manipulation strategies are mostly not invasive for the infected machines themselves. An exception are commands that download and execute programs. The exploitation of bugs is even more invasive than executing regular programs because exploit code is often required to be specifically tailored to the targeted host operating system and language. Frameworks like metasploit [21] help in developing generic exploit code. All in all, there still is a higher risk that remote systems are crashed this way. This has to be taken into consideration especially in scenarios where infected systems control critical infrastructures.

Before exploiting the bugs, infected systems have to be found. For decentralized topologies, they can be enumerated by counting connection attempts to injected bots. In locomotive topologies, this information can be extracted from sinkholing data. Other options are the use of honeypots, IDS signatures or scanners that scan network ranges for infected machines. In very rare cases, lists of other bots are available from central IRC C&C servers.

Exploitable vulnerabilities in bots have been found before [13]. Many Rbot and Sdbot variants share the same code base that contains vulnerable functions like this. A potential way to take down botnets would be to identify infected machines, exploit a vulnerability in the bot, and inject and execute code that shuts the malware down. Vulnerable code can still be found in recent malware. Conficker.B uses the MD6 cryptographic hash function for its digital signatures. The MD6 algorithm was found to contain a buffer overrun vulnerability and fixed in an update release that was immediately incorporated in Conficker.C [7]. While this particular vulnerability in Conficker.B was not exploitable, it demonstrates that even sophisticated malware is not immune to critical security holes. Actively attacking bot-infected machines raises lots of ethical and legal questions. We provide a summary of the most important of these aspects in section 5.

3.4. Conclusion on Offensive Strategies

The number of technically feasible strategies shows that there are plenty of possibilities to pro-actively act against botnets before they cause any harm. The use cases presented in the following section show that the offensive strategies are not purely theoretical but based on our practical research. While technically possible, the ethical and legal problems those strategies bring up have to be taken in consideration in practice. Before

starting to use (especially the invasive opportunities), an extensive discussion about those topics and authorities is required. The last sections of this paper are to be seen as a first step towards this.

A general challenge about many offensive approaches is that they have to be performed as stealthy as possible. Especially mitigation attempts can be countered by the botnet commanders. Manipulation possibilities can quickly be outdated with small protocol changes or the use of digital signatures. Furthermore, exploitable bugs can usually be fixed in a short time. In case a botnet is to be shut down, this must be performed globally and quickly to not leave any time to the botnet commanders for countermeasures themselves.

Experts consider prosecution of botnet constructors unlikely to have a strong impact on the global threat [20]. Instead, botnets must be fought on a technical level. Proactive measures must be taken as a joint effort of international security teams with local authority. This approach has proven successful and should be followed more consequently in the future [22].

4. Case Studies

This section contains some case studies where we present our research on the feasibility and effect of proactive countermeasures on real botnets. We focused on more sophisticated bots rather than standard IRC or HTTP based networks.

4.1. Kraken

If a botnet's communication protocol is known and messages can be forged, it is possible to inject commands that will be reacted upon by the bots. In case of the Kraken botnet, commands are requested from a server after selecting and resolving an entry from a list of domain names. By registering some of those domains and accepting connections from Kraken machines it is possible to send arbitrary commands to the bots. In [15], we have described the encryption used in the protocol. [16] have demonstrated how a remote cleanup can be conducted by issuing an update command that instructs bots to start a removal tool.

4.2. Storm Worm

The *Storm Worm* (also known as just *Storm*) is probably one of the most known bots worldwide [3,9]. While other specimens that use P2P technology were seen before, Storm was the first malware that used it in a way that the botnet could exist for more than three years. Storm is interesting for different reasons: First, spreading was almost only based on social engineering through sending Spam – people even started talking about *Spam campaigns* as the topics were linked to current news or dates like the Iran War or Christmas.

Storm uses an encrypted version of the *Edonkey* peer-to-peer protocol. We have been able to extract the 40-bytes XOR key through reverse-engineering of a storm sample and have built our own Storm P2P client to be able to infiltrate the network [9]. In P2P botnets like Storm, all nodes take part in the infrastructure and perform routing or searches for other bots. Being able to communicate with other nodes, the Storm

network routing infrastructure can be infiltrated and disrupted [3,8,9]. However, we have found a less complex yet more powerful approach [9]: We were able to extract Storm's algorithm responsible for the privilege calculation and to displace the original commanders. This makes it possible to issue own commands to all bots in the network.

Storm's command set has been reverse-engineered by us. Consequently, we would have been able to instruct Storm nodes to download and run an arbitrary binary, e.g., to remove the bot from the system. All in all, a complete take down was possible by combining attacks on the infrastructure and the command layer while exploiting a design flaw in the P2P protocol. Today, only an insignificant number of Storm machines is still existing.

4.3. Waledac

Waledac is another P2P bot that tunnels all communication through HTTP. Additionally, each message is encrypted using a hybrid encryption scheme that applies the AES and RSA implementations of OpenSSL. To be able to spy on the traffic, we conducted a man-in-the-middle attack and intercepted the RSA key exchange. One important observation was that the AES key used for further encryption was static rather than dynamically chosen, a design flaw that enables us to also decrypt Waledac messages offline, without the need for a man-in-the-middle proxy.

Being able to snoop on the traffic, we were able to manipulate the communication between two nodes and even developed a tool to construct and inject valid messages ourselves. A takeover strategy based on these findings would be to announce oneself to other Waledac hosts as a proxy node to achieve a prominent position and then drop important commands like DDoS instructions. We could also modify update commands to make bots download and execute our own binary instead of the one provided by the commander.

4.4. Conficker

The first variants of the Conficker worm implement a C&C query algorithm similar to the one used by Kraken. Every day, a list of domain names are generated. Some randomly selected names are then resolved and the corresponding hosts are contacted. The Conficker Working Group [15] has organized a collaborative effort for pre-registering and sinkholing these domains to make them unavailable to the Conficker constructors. Furthermore, vulnerabilities exist in Conficker's code that would theoretically allow for exploitation and execution of arbitrary commands on infected machines. We have developed a network scanner for reliable identification of Conficker hosts [5]. These techniques can be combined in a proactive defence strategy to take down the botnet.

5. Legal and Ethical Aspects

The technical feasibility of the presented countermeasures does not justify their use in practice. The conduction of these countermeasures may interfere with law or current ethical beliefs depending on their invasiveness and impact on third-parties. On the one hand, many people fear the debates and political consequences, and therefore the

general tendency is to stick to conservative approaches. On the other hand, the enormous damage caused by botnets cannot be simply overlooked. Since classical means have not proven to keep up with the increasing threat, discussions have to be initiated about more active strategies. The following sections discuss consequences of countermeasures against control servers and manipulation of traffic. The subsequent part discusses ethical, legal, and liability aspects of remote bot disinfections.

5.1. Targeting Control Servers

Most countermeasures that target C&C servers only can be regarded as non-critical. We assume that the commanders of botnets are criminals and cyber terrorists. Taking down their C&C server literally disarms them. The same holds for regular DoS attacks on those servers. However, DDoS attacks that use lots of bandwidth and processing power, yield to a trade-off between the large amount of resources consumed by the botnet and resources for DDoS countermeasures.

5.2. Targeting Traffic

Traffic manipulation is generally considered to be ethically and legally feasible as long as affected parties agree to it. Such alterations might be offered as a service to prevent DDoS attack, for example. Many users don't know about the threat and therefore don't take steps towards such agreements. Inspecting their traffic and modifying it is a legal problem in many countries, even though more and more countries, like the U.K., pass laws that allow traffic inspection from certain official organizations. Traffic inspection and traffic modification at ISP level would allow to remove Spam and DDoS commands as they are passed in known botnet traffic.

Such actions can also be seen as an indirect protection of the ISP's infrastructure. However, on the one hand, such courses of action raise ethical problems as users may interpret them as a kind of censorship. On the other hand, many users don't know about their infections and would really appreciate if their systems are not misused. A default policy included in contracts to allow ISP to perform those actions in conjunction with a possibility to withdraw would be a solution that is actually already evaluated at several sites.

5.3. Targeting Infected Systems

The most controversial discussion takes place about more invasive strategies, like a remote removal of bots from infected computers. This raises different issues:

1. Ethical: This bypasses the responsibility of users to keep their systems clean.

2. Legal: In most countries it is illegal to run software without the system owner's permission.

3. Liability: Who takes the consequences if the cleanup actions fail or cause problems?

A remote cleanup, in most cases, requires running a removal tool on the infected computer, which has been shown to be technically feasible for a range of botnets. This kind of cleanup has to be performed fast because otherwise new commands to kill the removal tools can be issued by the botnet commanders. Thus, asking all users is not feasible.

Up to now, users are responsible for their own systems. Remote cleanup with automated removal tools bypasses the user, his autonomy, and his responsibility. While some users interpret this as an intrusion into their privacy, a wide range of users would be very grateful for this kind of support to keep their systems clean. All in all, it keeps them a little safer from getting their banking or credit card details stolen. The typical use of AV software as an install-and-forget means supports this view.

Downloading and running software on a remote computer without the owner's permission is illegal in many countries because it is seen as an act of hacking into the system. However, some countries, like the Netherlands, require criminally motivated deeds for the applications of those laws. Since it is a general belief that botnets are run by criminals and cyber terrorists, the disinfection of hosts clearly states good will.

This may lead to the conclusion to run proactive strategies only for systems in specific countries or organizations that agreed on such actions. This selective approach is not very effective and yields at maximum in a mitigation but not removal of the considered botnet. Technically it is not always feasible to identify hosts from specific countries or organizations in the overlay network of the botnet. The cleanup of only selected systems rises the problem that the left-over partition may react, adjust to the new situation, and conduct a counter attack. Similarly, concurrent criminal organizations may observe those actions and may use the information to issue their own “updates”, which simply replaces the bot.

Cyber criminals and cyber terrorists act globally. Thus, countermeasure can only be effective when performed on a global level or at least in large parts of the Internet. A global take down would be the ideal situation. However, a global disinfection rises political questions because most country would not agree on another country's forces to remotely run software on their systems. This holds especially for infected governmental systems.
The foundation of a global organization with legitimation to perform those actions might be a solution. The discussions and consents about such an initiative have to take place on a political level.

Even the best software contains bugs. Invasive countermeasures, like removal tools, can lead to instability of the disinfected system, even with a low probability. This risk increases when bugs in a malware are exploited. In the unlikely case that this

happens, the liability is an important question. Who takes responsibility for this happening?

Closely linked to the liability question is the ethical question on the consequences of such actions on medical devices or critical infrastructures, for instance. However, the probability of malicious software causing harm is much more likely. During Conficker outbreaks in hospitals, medical devices were infected and stopped working properly. In the end, it is also a question of responsibility to leave no stone unturned – and that might even include a proactive botnet takedown to prevent further harm.

6. Conclusion

While technically possible, we argue that pro-actively fighting botnets requires immediate political and international consensus. It is a matter of the impact whether people would agree to offensive approaches or not. The affected systems' criticality have to be balanced against potential damage caused by countermeasures. This is, however, also the case for classical mitigation techniques. The portfolio of measures demonstrated in this paper range from more passive ones, like sinkholing, to offensive ones, like exploiting bot hosts to take them over and clean them. We believe that a framework for a staged approach that combines both defensive and offensive techniques should be prepared as part of an emergency response toolkit.

We have seen that cooperation is one of the most important aspects when it comes to successful and sustainable botnet mitigation. This holds for the technical and the political level likewise. Trusted forums must be strengthened and extended to be capable of reacting to botnet incidents effective and immediately. A laissez-faire policy does not lead anywhere.

References

[1] E. Stinson, and J.C. Mitchell, *Characterizing Bots' Remote Control Behavior*, Springer Verlag, Proceedings of the 4th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment, 2007

[2] S.W. Korns, J.E. Kastenberg, *Georgia's Cyber Left Hook*, 2009

[3] T. Holz et al., *Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm*, Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, 2008

[4] B. Stone-Gross et al., *Your Botnet is My Botnet: Analysis of a Botnet Takeover*, UCSB Technical Report, 2009

[5] F. Leder, T. Werner, *Know Your Enemy: Containing Conficker*, 2009

[6] P. Porras, H. Saidi, and V. Yegneswaran, *An analysis of Conficker's Logic and Rendezvous Points*, SRI International Technical Report, 2009

[7] P. Porras, H. Saidi, and V. Yegneswaran, *Conficker C Analysis*, SRI International Technical Report, 2009

[8] C. Kreibich et al., *On the Spam Campaign Trail*, First USENIX Workshop on Large-Scale Exploits and Emergent Threats, 2008

[9] G. Wicherski et al., *Stormfucker: Owning the Storm Botnet*, 25th Chaos Communication Congress, 2008

[10] F. Leder, *Waledac is wishing merry christmas*, `http://www.honeynet.org/node/325`, 2009

[11] F. Leder, *Speaking Waledac*, `https://www.honeynet.org/node/348`, 2009

[12] Team Cymru, *A Taste of HTTP Botnets*, 2008

[13] Sasser Ftpd Exploit, `http://www.securiteam.com/exploits/5AP0J0ACUM.html`

[14] The Conficker Working Group, `http://confickerworkinggroup.com`

[15] F. Leder, P. Martini, *NGBPA Next Generation BotNet Protocol Analysis*, IFIP SEC 2009

[16] P. Amini, C. Pierce, *Kraken Botnet Infiltration*, http://dvlabs.tippingpoint.com (2009)
[17] Symantec, *Symantec Global Internet Security Threat Report 2008*, 2009
[18] Malware Research Group, http://malwareresearchgroup.com (2009)
[19] J. Rutkowska, *Subverting the Vista Kernel for Fun and Profit*, Blackhat Briefings 2006
[20] R. Lemos, "*Arrests unlikely to impact bot net threat, say experts*", http://www.securityfocus.com/news/11344 (2009)
[21] The Metasploit Project, http://metasploit.com
[22] J. Stewart, Interview: "*Researcher argues for CERTs with teeth*", http://www.securityfocus.com/brief/950
[23] Ponemon Institute, *2008 Annual Study: Cost of a Data Breach*, 2009