
Fraunhofer FKIE

universitätbonn

# Current Trends in Botnet Development and Defense Expert Opinion

Study conducted for the Cooperative Cyber Defence Centre of Excellence (CCD COE)
Tallinn, Estonia
December 2010

G. Klein
F. Leder

# Copyright and Disclaimer

# Abstract

In this study, we give an overview about current trends in botnet development and defense. We start off by giving an overview of botnet technology and introducing core concepts. This is followed by a description of measures employed by botnet operators to secure their botnets against infiltration by security researchers and authorities. We discuss both the technical requirements of setting up and operating a botnet as well as the requirements with regard to skills, resources and time of infiltrating and taking down a botnet. We conclude with an outlook on future developments.

# Table of Contents

# 1 Introduction

Botnets are networks of computers infected with malicious software (malware) which are under the remote control of so-called botherders. Without the knowledge of their regular users, the infected machines within this botnet (so-called bots) are able, for example, to perform criminal activities such as sending unsolicited mass e-mail messages (spam), conducting distributed denial-of-service (DDoS) attacks or harvesting local sensitive data such as bank records. There are thousands of different active botnets, each containing thousands to several millions of infected hosts.

Beside the adverse effect of malware on local systems, sometimes rendering them unusable due to excess consumption of resources, damage is also caused elsewhere. In the past, the negative effect on available bandwidth due to DDoS attacks and spam were the biggest concern. However, with the increasing mobility of public services and administration into the Internet, economical and political disturbances are evident as well.

Financial damages are on the rise with increasing theft of credit card and online banking credentials. This is likely to escalate as more and more malware incorporates credential harvesting functionality. Extortion of money with the threat of large-scale DoS attacks against enterprises has also been seen in the recent past. Moreover, recent politically motivated DDoS attacks had the ability to immobilize public and private sectors of entire countries [16].

In this study we give an overview of current trends in botnet development and of the requirements for proactive countermeasures against botnets. The remainder of this document is organized as follows: Section 2 gives an overview of botnet technology, including different topologies and infection vectors. Section 3 describes the evolution of botnet technology, highlighting – among others – differences between generic, off-the-shelf botnets and highly specialized variants. In Section 4 we illustrate the current situation with regard to the botnet environment as well as trends in infection vectors, defensive measures on the part of botnet operators, and botnet command and control. Following that, we point out the requirements with regard to team strength, skills and time for setting up a botnet on the one hand, and infiltrating a botnet on the other hand (Sections 5 and 6). We conclude this study with an outlook on future developments (Section 7).

# 2 Overview

This section provides an overview of botnet technology. We discuss differences in topology, traditional infection vectors and classical countermeasures.

## 2.1 Botnet Topology

As with any distributed system, communication is a key aspect of botnets. Two elements need to be considered here: addressing mechanisms through which command and control (C&C) instances can be reached, and the choice of communication protocol [22].

The means through which C&C servers are reached define the *topology* of the botnet. We differentiate between distinct variants, although the boundaries between these are blurred.

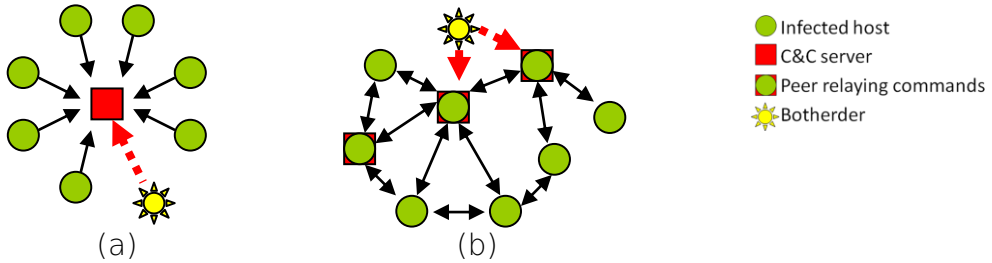


Figure 1: Example botnets with centralized (a) and decentralized (b) topologies [22].

In a *centralized* topology (which is the classic botnet structure), there is a single or fixed set of C&C servers, from which all clients receive their instructions. An example is shown in Figure 1 (a). A botherder (symbolized by a star) issues commands to a static C&C server which then instructs the individual bots. A central C&C server is reachable under a single IP address for which bots typically perform a DNS lookup. The communication process often makes use of established client/server protocols such as HTTP or IRC. Communication with the C&C server can be either push-based (as with IRC) or pull-based (as with HTTP). In the former case, bots remain connected to the server and when commands are issued, they are received by all bots simultaneously. In the latter case, bots regularly contact the server, requesting instructions. Examples of botnets using this topology are *AgoBot* [1], *RBot*, *SdBot* and the infamous *ZeuS* botnets.

In a botnet with a *decentralized* topology, no single entity acts as a C&C server. Instead, each bot knows a number of other bots in the network and relies on peer-to-peer (P2P) query mechanisms to receive current commands and lists of neighbors. This is depicted in Figure 1 (b). In this example, a botherder issues commands to three different bots that propagate the commands to other bots. Because it is a network of peers, no real C&C server exists but only

single peers that forward relevant commands. Similar to centralized botnets, the command-and-control process can take place in a pull or push fashion, where the former is the more common approach. P2P botnets are usually independent of domain names and create their own overlay structure for addressing. Examples of botnets with a decentralized topology are the *Storm worm*, *Nugache* and *Conficker*. More information on the former two can be found in [41].

While there is no single controlling instance in P2P botnets that can be infiltrated, the dynamic routes in the overlay network can be manipulated by individual nodes, thus making the entire network vulnerable. In addition to that, the creation of a suitable P2P infrastructure is much more complex. Also the longevity of P2P botnets which was observed at first is no longer present.

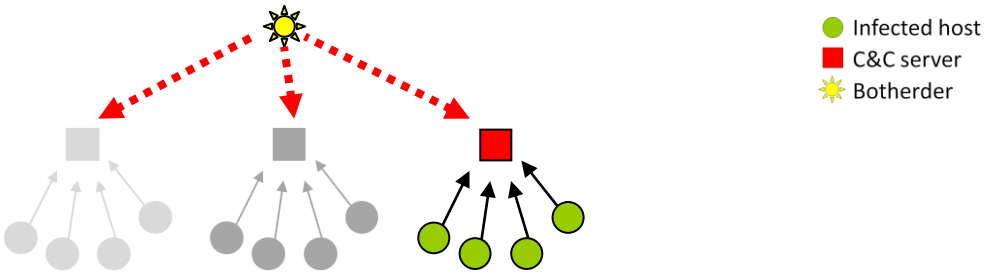


Figure 2: Example botnet with »locomotive« topology [22].

Occupying the middle ground between the two extremes (centralized and decentralized) are semiflexible topologies. These can be achieved, for example, by using *fluxy* domain registrations, i. e. returning lists of multiple IP addresses for a single DNS query. This provides obfuscation mechanisms as well as load balancing on the C&C server. Another strategy is to vary the DNS-queried domains over time. Due to its constantly changing nature, we call this highly dynamic topology »locomotive« (shown in Figure 2). Examples for botnets using this type of topology are *Torpig* and *Conficker* (in addition to its P2P components).

It was predicted in the past that future botnets would rely solely on P2P mechanisms. This has not come to pass, possibly because of the associated weaknesses. Also, more or less centralized structures with few C&C servers protected by bulletproof hosting are highly reliable and difficult to act against.

## 2.2 Infection Vectors

Because of its widespread use, the most common target for malware is Microsoft Windows. Although malware for other platforms including embedded systems [7] or mobile phones has been detected, these samples are far less widespread. Infection of victims' systems is possible by exploiting either technical flaws or design weaknesses in operating system (OS), service or application software, so-called *vulnerabilities*. Depending on the attacked component, multiple infection vectors are possible.

A hardware based method of exploiting OS vulnerabilities is the use of infected mass storage devices such as USB flash drives. The Windows mechanism of automatically executing software on these drives (*autorun*) makes it easy to install malware from an infected device. Due to increased security consciousness, autorun has been disabled on many machines. Despite this, the recently publicized *Stuxnet* botnet made use of a previously unknown vulnerability through which even machines with the autorun feature disabled could be infected simply by displaying the icon of a Windows application link (LNK) file [8].

A widespread infection method is the use of so-called drive-by-downloads. Here, weaknesses in Web browsers are exploited as a user visits a Web site. Typically this is either a regular Web site which has been hacked to include malware, or an innocuous-looking phishing Web site which has, for example, been fashioned after an online banking portal. A similar technique exploits weaknesses in viewer software for popular document exchange formats such as Flash or PDF. Here, a user is sent infected documents with contents especially relevant to the user's professional or personal background in the hope that they will be opened. This is also referred to as *social engineering* and is described in greater detail in Section 4.2.

Once a host is infected, it will typically seek to infect other machines it can reach. This can be accomplished by sending e-mails containing either links to infected Web sites (for drive-by downloads) or infected documents.

Additionally, vulnerable network services can be exploited by scanning entire domains for open ports and then sending crafted packets to these ports, resulting, possibly, in privilege escalation and malware infection. In the early days of malware, this was the original method of spreading and is still used today.

## 2.3 Classic Countermeasures and Mitigation

Once the existence of a botnet has been determined, measures can be taken to mitigate its effects. Classical countermeasures include, for example, taking down the C&C server or sinkholing malicious traffic.

Removal of the C&C server has the advantage that it renders the entire botnet useless. However, this is only possible if there is only one server (centralized topology) and the location of that server is known. Also, the cooperation of the Internet service provider hosting the server is necessary. In the case of bulletproof hosting, this is next to impossible.

If taking down the C&C server is not possible but its address is known, an option is the *sinkholing* of traffic to that address. This consists of redirecting traffic meant for the C&C server, e. g.

requests for instructions from infected machines, to other locations. Here, the traffic is then analyzed and then discarded. Sinkholing also works in the other direction where malicious traffic from infected hosts (e. g. belonging to a DDoS attack) is redirected.

In most cases, DDoS attacks can be mitigated by an increase in connection bandwidth. Internet connections via multiple providers permit measures such as the disconnection of several links to cut off certain attack source addresses. Additionally, some DDoS mitigation firms specialize on packet filtering at firewalls according to certain traffic patterns detected at border routers, e. g. distribution of TYP SYN packets [3], [33].

At the time a new malware binary begins to spread, typical anti-virus solutions unfortunately have a comparatively low detection rate. At first, signatures for new malware are not available. The length of the signature creation process can vary between several minutes and several days. In the ideal case, the sample is received and analyzed by the same vendor; the signature is generated automatically and then distributed to customers. In the much more common and lengthy case, sample collection and analysis is performed by separate vendors, automatic signature generation is not possible and needs to be done manually by an overworked analyst. Until signatures reach customers, time spans of multiple days can elapse. Even if signatures are available, AV installations are not always kept up-to-date by end users. Also, existing AV solutions can often be disabled by already installed malware.

However, once signatures are available they can be integrated into disinfection tools such as the *Malicious Software Removal Tool* (MSRT, [26]) offered by Microsoft. Often, the distribution of a specific malware sample is then noticeably reduced. Unfortunately, user awareness about tools such as MSRT is still fairly low and they are not in widespread use.

# 3 Botnet Evolution

Over the past years, botnet development techniques have undergone a certain evolution. What started off with open-source botnets (or components thereof) has developed into often commercially available botnet construction kits. A recent evolutionary step is the appearance of highly specialized, professionally developed botnets with extremely specific targets.

## 3.1 Open-Source Software

The earliest known botnets were developed as open-source software, e. g. *AgoBot*, *SdBot* or *RBot*. The entire source code is easily available via popular search engines or on criminal online forums. For users with a reasonable level of technical skill, they are easily configurable (with respect to IRC command channels or passwords, for example) via changes directly in the code. Even though the code base of such botnets is typically older, they can be extended with components for new exploits, thus remaining usable for a long time. These new exploits can either be developed individually or obtained from projects such as Metasploit [25]. The latter has the advantage that the code is very reliable and has been tested by a wide range of users.

From a botherder's perspective, this has the advantage that new botnets can be very quickly set up and put into operation. On the other hand, the easy configurability of such botnets enables the use of standard detection mechanisms for C&C server location and other parameters.

An alternative to the operation of entirely open-source botnets is the incorporation of only certain open-source components into an otherwise closed-source application. This provides security against errors made in individual implementations of standard software components. Especially the use of widespread libraries ensures preceding in-depth testing by the open-source community.

Popular open-source components are cryptographic and compression routines. For example, *Waledac* used the OpenSSL library [29], [21] for both RSA and AES, *Conficker* included the official MIT implementation of the MD6 hash algorithm [36], and *Storm* made use of the zlib [14] compression library. To facilitate the malware distribution, these components are not shipped as separate components but compiled directly into the malware binary.

## 3.2 Construction Kits

To an increasing degree, botnets are used for monetary gain, mostly through harvesting financial credentials such as online banking access data or credit card details. Thus, the technology behind them is increasingly valuable and thus a decreasing use of open-source botnets can be observed. Because of the high market value, considerable effort is put into the professional development of botnet software. Developers use state-of-the-art methods with regard to development processes and quality assurance, among others [12].

To ensure a return on this investment, the botnet software is not freely distributed but sold in the form of construction kits. These are often point-and-click applications in which users can very easily set up an entire botnet. This is described in more detail in Section 5.

We have observed that even these commercially available construction kits incorporate open-source components, although these are mostly for basic functionality such as DDoS attacks or spam e-mails. Because these components have undergone considerable development in the open-source community, there is no need to invest more effort into their development. Note that construction kits are also available for some older botnets but these are not commercially marketed because their focus was mostly on DDoS and spam for which there are open-source components.

Prominent examples of botnets that are sold as construction kits are *ZeuS* and its presumable successor, *SpyEye*, both targeting financial data. In the case of *ZeuS*, there is also an open-source component: the C&C server is written in PHP (with technical documentation in the form of code comments in Russian).

Listing 1: Excerpt from the PHP code of the *ZeuS* C&C server.

```
//Парсим данные (Сжатие данных не поддерживается).
//Поздравляю мега хакеров, этот алгоритм позволит вам спокойно читать
//данныебота. Не забудьте написать 18 парсеров и 100 бэкдоров.
$list = array();
for($i = HEADER_SIZE; $i < $data_size;)
{
  $k = @unpack('L4', @substr($data, $i, ITEM_HEADER_SIZE));
  $list[$k[1]] = @substr($data, $i + ITEM_HEADER_SIZE, $k[3]);
  $i += (ITEM_HEADER_SIZE + $k[3]);
}
unset($data);
```

Listing 1 shows a small section of the PHP source code from the *ZeuS* C&C server. The translation of the Russian comments is roughly »Parses the data (compression not supported). Congratulations mega hackers, this algorithm will allow you to

easily read from a bot. Do not forget to write a parser for 18 to 100 backdoors«.

## 3.3 Specialized Botnets

The latest step in botnet evolution is the emergence of highly specialized botnets aiming at the infection of very specific targets. This malware is highly professionally developed software combining expertise in both exploit development, e. g. 0-day exploits which are unknown to the software developers and for which no patches are available, and target domain knowledge.

Examples of such highly specialized botnets are *Ghostnet*, which aimed at political espionage in China-critical communities, *Stuxnet*, which targeted supervisory command and data acquisition architecture (SCADA) systems controlled by Windows hosts, and *Waledac*, which targeted financial information of infected systems.

More innovation regarding packing or encryption mechanisms of binaries takes place in this context than can be observed in the case of construction kit-based botnets. Because of the possible (financial) gains, developers want to create stealthy, reverse engineering-resistant software. Thus, substantial effort is put into hiding the malicious software – either in the form of individual developments or purchases of third-party routines.

# 4 Current Situation and Latest Trends

Thus far, botnet developers and operators clearly have the upper hand. If newly developed malware is released, the detection rate of up-to-date anti-virus (AV) software can vary greatly. Sometimes, less than 10 % of samples are detected within the first 24 hours of their appearance; only in extremely seldom cases does the rate exceed 90 %. Figure 3 shows a summary of detection rates of popular AV solutions. Modern worms spreading via networks can infect all reachable systems within a few hours.

Figure 3: Detection rates of popular anti-virus solutions, accessed 8 December 2010 [39].

| vendor | detected | total | percent |
|---|---|---|---|
| AntiVir | 3,471,552 | 3,994,505 | 86.91% |
| NOD32 | 3,341,179 | 3,994,505 | 83.64% |
| DrWeb | 3,266,944 | 3,994,505 | 81.79% |
| F-Secure | 3,204,919 | 3,994,505 | 80.23% |
| Sophos | 3,197,983 | 3,994,505 | 80.06% |
| Kaspersky | 3,167,274 | 3,994,505 | 79.29% |
| VirusBuster | 3,144,314 | 3,994,505 | 78.72% |
| VBA32 | 3,062,898 | 3,994,505 | 76.68% |
| Norman | 3,028,199 | 3,994,505 | 75.81% |
| AVG7 | 2,887,545 | 3,994,505 | 72.29% |
| QuickHeal | 2,848,256 | 3,994,505 | 71.30% |
| TrendMicro | 2,825,513 | 3,994,505 | 70.73% |
| Clam | 2,813,341 | 3,994,505 | 70.43% |
| Ikarus | 2,746,674 | 3,994,505 | 68.76% |
| Avast-Commercial | 2,702,956 | 3,994,505 | 67.67% |
| F-Prot6 | 2,148,018 | 3,994,505 | 53.77% |
| BitDefender | 1,961,341 | 3,994,505 | 49.10% |
| McAfee | 1,652,032 | 3,994,505 | 41.36% |
| Panda | 1,515,043 | 3,994,505 | 37.93% |
| Vexira | 1,182,029 | 3,994,505 | 29.59% |
| G-Data | 200,326 | 3,994,505 | 5.02% |

The only reason why the detection rate is this high is that newly released malware is often only a slight variation of older malware. The deployment of up-to-date AV software is advisable, but especially against targeted attacks (e. g. via »personalized« infected documents) it is highly ineffective. In a way, this is the current crisis of the AV industry and there is a general consensus that current solutions are neither scalable nor sustainable. Often manual analysis of malware samples is required because the available tools are not advanced enough. The question of finances is another issue. AV vendors often do not have the financial resources of botnet operators whose business model relies on their software being undetectable. Estimates say that botnet-generated income is in the region of 10,000-10,000,000 USD per botnet per month [24], [42] which can be invested in professional malware development. With over 200 *ZeuS*-based botnets alone being tracked in 2009, the resulting sums are quite substantial. In comparison, F-Secure and Kaspersky, two notable AV vendors, had

yearly revenues of 125,000,000 and 100,000,000 EUR, respectively. With profit being only a part of these sums, not the entire amount can be invested into botnet research.

## 4.1 Economic Aspects

In 2008, spammers earned an estimated 780 million USD [30] and there is an upward trend to these numbers. With this ever increasing amount of money to be made by operating or renting out botnets, we have observed an increasing professionalization in the domain. Structures similar to free market (sub-) economies are emerging where prices and the availability of products and services are regulated by demand. There are even marketing campaigns on underground forums promoting certain products.

On the lowest layer, this is manifested in traditional criminal schemes such as extortion. Often it is enough to threaten Web site operators with large-scale DDoS attacks for them to agree to payments. An example of the free market nature of an extortion scheme is the e-mail depicted in Figure 4. Here, payment is demanded or a site will be attacked. However, should the site operator pay within the next 24 hours, there is a discount; the »price« is reduced.

Figure 4: Extortion e-mail offering a discount on early payment (translated from German).

```
ddosforeveryone@gmx.de

Dear Management,

As you may have noticed, your online shop abc.de was under DDoS attack on
Thursday, 08.04.10 from 12-5 pm. We will perform another DDoS attack on your
online shop on Friday 09.04.10 unless you meet the demand below. During this
time your online shop would again not be available because of the attack. The
result would be loss of revenue for you.

We offer you the following option:

1. Go to http://www.ukash.com/de/de/where-to-get.aspx and enter your ZIP code.

2. Drive to one of the addresses given to you and purchase ukash to the value
of 250€.

3. Send me the received voucher number to this e-mail address before Friday
2 pm.

Should we not hear from you, the fee will increase from 250€ to 350€ and your
online shop will be attacked again.

If you send me the ukash voucher before today 11.59 pm, I will give you a
discount of 100€ - today until 11.59 pm the fee is only 150€.

Kind regards

PS: Resistance is futile, you cannot win this resource war
```

Another example of market forces is the merger of two botnets. Until very recently, the *ZeuS* and *SpyEye* botnets were

competitors. Both are banking Trojan horses and have the same target: harvesting of online banking credentials and credit card information. Infection of a system with either malware would result in the removal of the other and in some cases the immunization against further infection by the other malware. Not too long ago, either a »merger« or a »hostile takeover« occurred, with the *Zeus* codebase now being merged with that of *SpyEye* and technical support being handled by the *SpyEye* author [20]. Listing 2 shows an excerpt from a translated forum posting by the SpyEye author in which he announces technical support for *ZeuS* customers.

Listing 2: Forum posting of *SpyEye* author announcing support for *ZeuS*.

```
I will service the Zeus product beginning today and from here
on. I have been given the source codes free of charge so that
clients who bought the software are not left without tech
support.
```

Figure 5 shows a screenshot of a part of the Readme file from the Eleonore Browser Exploit Pack (along with the Google translation from Russian). Here, the author offers technical support via ICQ chat.

As can be seen, there is a market for technical support for malware products. In general, there is a trend towards all-inclusive packages, in which a customer purchases a botnet construction kit along with setup support (often round-the-clock via telephone or Internet chat), administration and bulletproof hosting for the C&C server.

Figure 5: Extract of the Readme file from the *Eleonore Browser Exploit Pack* offering support via ICQ chat.

```
=======================================
 ========>   Eleonore Exp pack 1.3.2  <==========
=======================================

Контакы:
-------------------------------------------------
ICQ: 9-000-001 (ExManoize) Автор/поддержка связки.

Contact:
-------------------------------------------------
ICQ: 9-000-001 (ExManoize) Author / support bundles.
```

Not only are support and consulting activities on the increase, but end-user training is offered as well. The »Cash Paradise University« (Figure 6, [6]) offers various courses in cybercrime.

Figure 6: Logo of the »Cash Paradise University«.

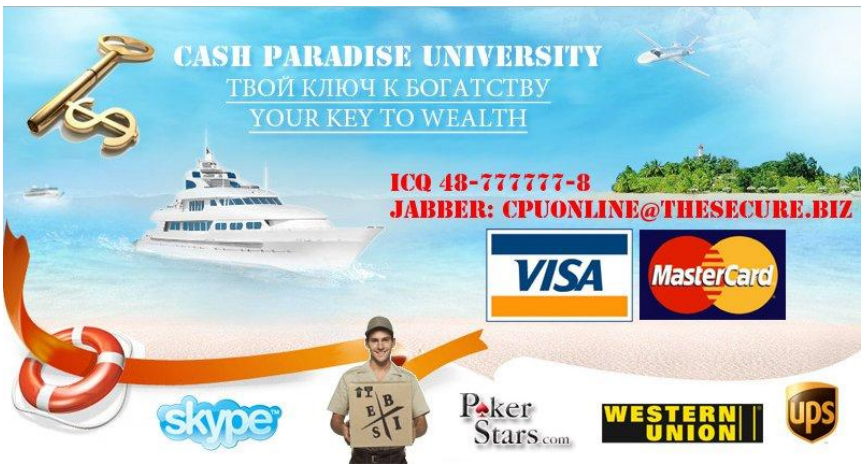


Figure 7 shows extracts of the Cash Paradise University's study program: an intermediate course in phishing and an advanced course in botnet setup. Similar to a regular university's curriculum these courses are built on top of one another with increasing skill level. The curriculum also includes courses on how to properly launder money made by botnet activities, often making use of unsuspecting so-called *money mules*.

Figure 7: Extracts of the *Cash Paradise University* study program (screenshots).



Beyond support services, developer groups are also – for a fee – offering custom extensions to existing products. For example, the basic *ZeuS* construction kit included modules for only one bank in Germany, the Postbank. However, extension modules for practically every other bank have been developed. This is similar to the extension of existing construction kits with modules for newly released exploits.

Beside the professionalization in the purely economic structures, there is a coinciding trend in the actual software development process where modern practices such as collaborative development via revision control systems and quality assurance by regression testing are employed [12]. In a sense this also has economic impact because the increased software quality translates directly into increased revenue.

Prices of botnet construction kits start at a few thousand USD, e. g. *ZeuS*: 3,000-4,000 USD. The more extensions are incorporated into the kits, the more expensive they are. Prices of *ZeuS* kits can range up to 10,000 USD [40]. As of late, botnet developers use licensing schemes such as *VMProtect* [45] to control in which way their software is used. It allows, among others, code protection, e. g. against reverse engineering, selective enabling of features and upgrade management. Figure 8 shows a screenshot of the *ZeuS* construction kit for which the license is invalid. The subsequently started light mode might include limitations to the features that can be included.

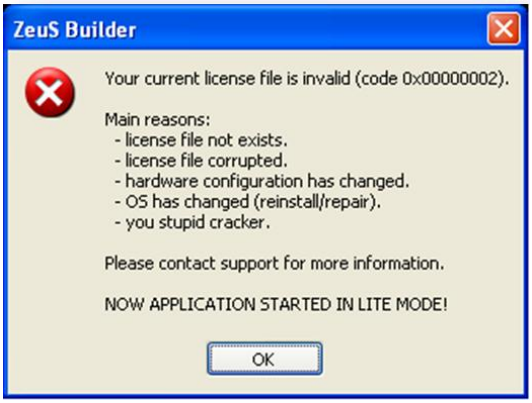


Figure 8: *ZeuS* construction kit screenshot with invalid license file.

These activities are illegal in most Western countries and result in strict criminal prosecution with some notable arrests having been made in many European countries [13], [47], [31]. The ensuing increase in market value for such services has made it lucrative for operators to move to low-income, often Eastern European and Asian countries with a low level of criminal prosecution (e. g. Russia, Ukraine). There are reports of entire suburbs or small towns devoted to malware development, distribution, marketing, money laundering, etc. Further information about the relationships and players behind botnet operator activities can be found in [24].

## 4.2 Infection Vectors

Originally, Internet worms, one of the earliest forms of malware, spread to new hosts was by exploiting server services on the target systems (e. g. Windows file sharing). While this is still relevant for second-tier infection, the prevalence of this infection vector is on the decline.

A new trend is the increasing exploitation of vulnerabilities in client-side applications. These are often ubiquitous on user desktops and thus an easy target. Examples for these kinds of applications are Adobe's Portable Document Format (PDF) reader and Flash, Microsoft Office programs, or Web browsers. The latter can be exploited by so-called *drive-by downloads* on infected Web sites. These Web sites can be both legitimate sites hacked by

criminals, or sites specially set up for the express purpose of infection. In the latter case, mass spam e-mails containing links to these sites lured users to these sites.

There is an increase in so-called *targeted attacks* which contain a social engineering component. Detailed background information is gathered on the intended targets and personalized messages are sent to the victims, either via e-mail or through social networking sites. By exploiting information about the target's current personal or professional situation (e. g. hobbies or work-related activities), the target can be tempted to open either infected attached files or visit suggested Web sites. To gather such personal information about potential targets, botnets trawl public social network profiles and attempt to find so-called social engineering vulnerabilities. An infection vector such as this is not suitable for large-scale infections, but if the botnet is highly specialized and targeting only a small community or company it can be very effective.

Because of the perceived relationship quality of connections in social networks, users place a high degree of trust in content received from members of such social networks. Accordingly, credentials of social networking sites are an ever popular target. These can be obtained in a multitude of ways. Figure 9 shows a phishing e-mail sent by one of the *ZeuS*-based botnets to trick users into entering their Facebook account password. After password entry, a file purportedly containing an update in reality contained the *ZeuS* banking Trojan horse. Similar e-mails containing »Facebook password reset links« were sent by the *Bredolab* botnet.


Figure 9: Phishing e-mail asking Facebook users to update their account [10].

A botnet specifically targeting social network login data is *koobface*. It spreads via messages to social network contact lists (friends). These messages contain links to a supposed Adobe Flash update file containing the *koobface* malware. Infected systems are then manipulated to click on online advertisements or users led to purchase so-called scareware, e. g. fake anti-virus software.

Also, by means of so-called session hijacking, malicious users can easily access the social network profiles of users on the same network (see e. g. the Firesheep plugin for the Mozilla Firefox browser [5]). The actual harvesting of login data is accomplished with keyloggers on the infected machines. Once profiles have been accessed, messages containing or linking to infected material can be sent to users' entire address books.

According to the Websense 2010 Threat report, 79.9 % of Web sites with malicious code were compromised legitimate sites [46]. Recently, Web sites containing political content have been a popular target for malicious code: the Web site of Amnesty International Hong Kong (a human rights group) was injected with a 0-day exploit for Microsoft's Internet Explorer browser along with exploits for Quicktime, Flash and Shockwave. Also, the Nobel Peace Prize laureate's Web site was injected with a 0-day exploit for the Firefox browser.

## 4.3 Technical Defensive Measures

To protect their software from reverse engineering and analysis, malware authors increasingly employ defensive measures on a technical level.

An often-used mechanism is encryption, both of the communication with the C&C server and of the malware binary itself. Circumvention of the former was possible, for example, in the *Storm* botnet, because static keys were used which could be extracted from binary code. This is an imminent weakness in botnets using encrypted communications because the encryption keys either need to be included in the binary or can be observed when processed in the binary during runtime. In the case of *Storm*, the binary includes a decryption routine which is only executed once the binary is loaded into memory. Thus, no decrypted software is ever stored on disk. However, there is a certain moment just before the code is executed, in which it can be observed in memory in decrypted form by using a debugger.

Obfuscation is a technique with which the fact can be hidden that different malware samples belong to the same botnet. A recent trend is so-called *server-side polymorphism*. Here, the server from which a newly infected machine retrieves the actual malware encodes the binary differently for every client. This can include differences in the encryption routine, encryption keys, etc. The result is that binaries from two different infected hosts have nothing in common at first glance. A further development of this concept is that malware servers only allow one connection from a single client within 24 hours. This makes it exceedingly difficult for AV vendors to gather enough samples to perform meaningful analyses. The collection of samples is also hindered by the fact that malware developers can explicitly avoid infecting systems with the sole purpose of malware analysis. This is done by

implementing blacklists of IP addresses of known honeypot or other analysis systems (see AV Tracker, Section 4.5).

What is more even more threatening, already existing malware can be precisely immunized against certain AV products or analysis tools. *RDG Tejon Crypter* [34] is a software tool which allows the user to specify against which security products a malware sample is to be protected. Partly, this works by modifying the malware binary using techniques described above. However, it is also possible to disable active AV solutions or analysis tools (runtime anti-reverse engineering). Thus, if malware detects that it is being executed within a debugger or a sandbox, it will either stay inactive or attempt to circumvent the analysis environment. Figure 10 shows a screenshot of an *RDG Tejon Crypter* dialog in which the user can specify AV solutions that should not be able to detect the malware binary.

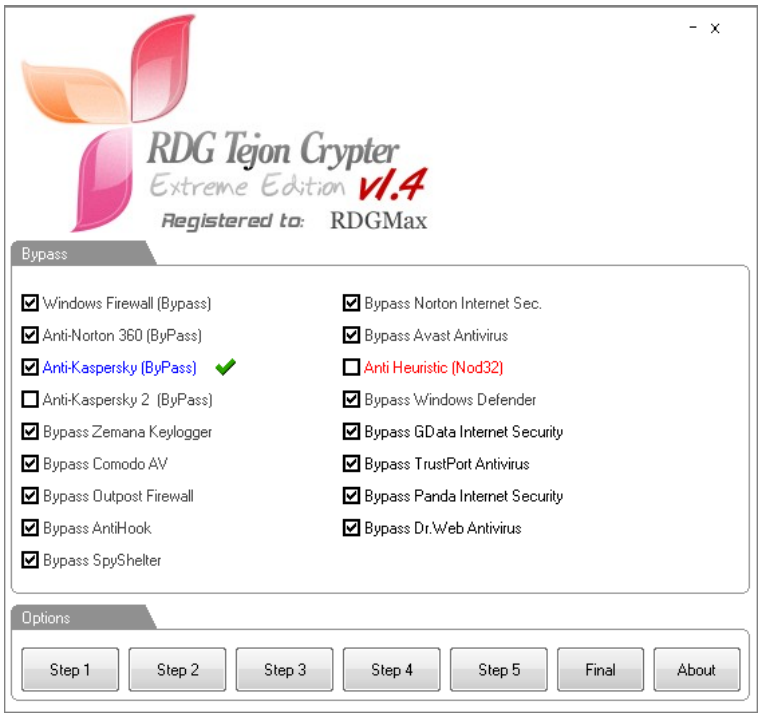


Figure 10: *RDG Tejon Crypter* screenshot showing products against which malware is immunized.

In general, it can be observed that open-source botnets typically have only few sophisticated defense mechanisms but sometimes use custom adaptations of standard mechanisms. Botnets made with construction kits often make use of so-called defense kits. Just before the malware binary is compiled, the user can specify which kit is to be used for integrating defensive measures. Other construction kits come with their own defensive means built in. Where specialized botnets are concerned, typically, various sophisticated defense mechanisms are employed in addition to the standard variants. Because security researchers actively study and

circumvent these defensive mechanisms, the result is a constant arms race in which botnet operators and developers continually develop new and more advanced mechanisms which are then analyzed and bypassed by the security industry.

## 4.4 Botnet Command and Control

Command and control processes are a current topic of experimentation within the botnet development community. In general, we can observe that there is a tendency to reuse (or misuse) existing protocols or services. This saves on development effort as well as it ensures functionality and scalability.

Traditional approaches to botnet C&C are the use of the Hypertext Transfer Protocol (HTTP, [11]) and the Internet Relay Chat (IRC) protocol [28]. Both were briefly mentioned in Section 2.2. The former is a pull-based variant in which bots query a specific Web address for instructions. The latter is push-based and requires bots to be constantly connected to an IRC chat server. Commands issued by a botherder in a chat channel are then transmitted to the bots simultaneously.

The increasing prevalence of social networks such as Twitter and Facebook has prompted botnet developers to experiment with these services for command and control. Both possess an extremely high-performance infrastructure capable of coping with several hundred million users. In the case of Twitter, the service is anonymous and botnet construction kits offer easy integration by allowing the simple specification of a Twitter channel, a so-called *tweet*. Figure 11 shows an example of this. Bots then »follow« (i. e. subscribe to) the tweet and thus receive instructions. The disadvantage is that there is a single point of failure. At any time, Twitter can decide to delete the tweet.

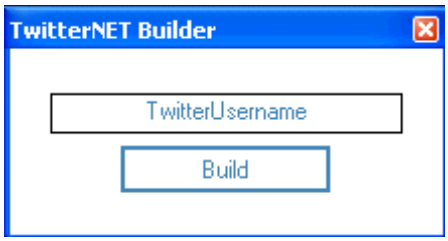


Figure 11: Screenshot of the last step of the creation process in the *TweBot* construction kit.

For Facebook C&C, fake personal accounts with public profiles are created that are then accessed by the individual bots. Botherders issue instructions by posting (encoded) commands onto the profile. Again, there is a single point of failure. In the recent past, Facebook has created a very effective anti-abuse department. Most of the time, it takes only minutes to hours until such fake accounts are deleted.

Because of these drawbacks, botnets with Web 2.0-based C&C mechanisms, e. g. *TweBot*, are not very widespread. The *Koobface* botnet uses Facebook solely for distribution, C&C is achieved via HTTP.

»Locomotive« botnets have C&C servers whose location changes very often. This could either be a static list of locations resulting in round robin-based load balancing for the C&C servers. Or, in botnets such as *Conficker*, *Kraken* and a newer variant of *ZeuS*, bots generate a list of random domain names every day (50,000 per day for *Conficker C*) under which they try to reach a C&C server. It is often possible to predict the generated domains; they can then be pre-registered with the DNS registries and thus made unavailable for botnet operators. This was done successfully for *Conficker*. Whereas the technical aspects of domain name prediction are simple and straight-forward, the coordination with and cooperation of DNS registrars is a challenge. These entities do not benefit from domain preregistration other than through the generated media attention. Normally, it is only a question of time until they cease to cooperate.

In traditional drive-by-download schemes, users are redirected to a fixed malicious Web site for infection. A recent addition by some developers of the *mebroot* and *torpig* botnets is to seed a domain name generator with current Twitter search trends [15]. Infected legitimate Web sites then redirect users to these generated domains. Figure 12 shows a Javascript function injected into a legitimate Web site which retrieves Twitter search trends.

Figure 12: Javascript function retrieving Twitter search trends.

```
function c(x){
    window.tw = x;
    eval(z($));
    document.write($);
}
document.write("<scr"+"ipt language=javascript"+
" src='http://search.twitter.com/trends/weekly.json?callback=c&exclude=hashtags'>"
+ "</scr" + "ipt>");
```

Also widely used for malware communication, especially with online banking Trojan horses, are instant messaging protocols such as XMPP (Jabber) [37]. Here, the focus is less on C&C but rather on the efficient, real-time transmission of harvested account information and user credentials, e. g. for the instant, online modification of initiated bank transaction. A prominent example of this is an available module of the *ZeuS* botnet, *jabberzeus*.

## 4.5 Inside Information on AV Industry

Botnet operators and developers are extremely well-informed about current products, technologies and innovations of the AV industry.

One of the reasons for this is a Web site called *VirusTotal* [43]. Here, malware samples can be submitted for scanning with all major anti-virus products. In a sense, VirusTotal can be referred to as a meta-anti-virus solution. Thus, malware developers can selectively harden their software against detection mechanisms. A disadvantage (from the botnet developers' point of view) is that VirusTotal passes on all submitted malware samples to the AV vendors who can then, in turn, optimize their products. Because of this, alternatives to VirusTotal have appeared which – for a fee – keep submitted samples to themselves. Examples of such services are *VirusTrap* [44], *avhide* [4] and *NoVirusThanks* [27].

Another Web site benefitting malware authors is *AV Tracker* [19]. It contains a comprehensive list of sandboxes, Honeypots and other analysis systems operated by the AV industry and malware researchers world-wide. The list is populated submitting a software sample to VirusTotal whose sole purpose is the transmission of the IP address of the infected system to its author. Since VirusTotal passes submitted samples on to analysts, this is a simple method for finding out the addresses of analysis systems. With the knowledge of these addresses, malware authors can implement blacklists of addresses to which their software should not spread or on which it should remain inactive, thus avoiding unwanted analysis. Going one step further, *ZeuS* operators set up a honeypot-like system to analyze and provide further information about researchers trying to infiltrate its administrative interface [17].

# 5 Requirements for Setting up Botnets

In this section, we discuss the requirements for setting up a botnet. Similar to the structure of Section 3, we differentiate between open-source botnets (or components thereof), botnets generated by construction kits, and highly specialized botnets.

## 5.1 Open-Source Software

Examples of still widely used open-source botnets are variants of *SdBot* and *RBot*. In general, setting up and deploying open-source malware is no more difficult than installing regular open-source software. The user needs basic knowledge of source code configuration and needs to be able to use a compiler. Figure 13 shows a screenshot of the *SdBot* source code section in which the bot is configured.

Figure 13: Screenshot of *SdBot* source code section in which the bot is configured.

```
//
// ** standard configuration:**
//

int port = 6667;                        // server port
int port2 = 6667;                       // backup server port
int socks4port = 12525;                 // Port # for sock4 daemon to run on  - CHANGE THIS!!!
int tftpport = 1576;                    // Port # for tftp daemon to run on
int httpport = 8180;                    // Port # for http daemon to run on
int rloginport = 519;                   // Port # for rlogin daemon to run on
int maxrand = 5;                                // how many random numbers in the nick
int nicktype = CONSTNICK;               // nick type (see rndnick.h)


BOOL nickprefix = FALSE;                // nick uptime & mirc prefix
BOOL topiccmd = TRUE;                   // set to TRUE to enable topic commands
BOOL rndfilename = FALSE;               // use random file name
BOOL AutoStart = FALSE;                 // enable autostart registry keys

//

char prefix = '.';                      // command prefix (one character max.)
const char mutex[] = "testbot";         // mutex, change this to whatever
char lsaip[] = "";                      // ** DONT CHANGE THIS LINE **
char lsaport[] = "2227";                // change this to whatever port you want.
char lsuser[] = "schi";                 // change this to whatever user you want.
char lspass[] = "infi";                 // change this to whatever pass you want.
char botid[] = "testbot";               // bot id
char password[] = "testbot";            // bot password
char server[] = "127.0.0.1";            // server
char serverpass[] = "";                 // server password
char channel[] = "#testbot";            // channel that the bot should join
char chanpass[] = "";                   // channel password
char server2[] = "";                    // backup server (optional)
char channel2[] = "";                   // backup channel (optional)
char chanpass2[] = "";                  // backup channel password (optional)
char filename[] = "testbot.exe";        // destination file name
char keylogfile[] = "testbot.db";       // keylog filename
char valuename[] = "Windows Registry Name";     // value name for autostart
char nickconst[] = "BOT-";              // first part to the bot's nick
char szLocalPayloadFile[]="testbot.dat";        // Payload filename
char modeonconn[] = "+xi";              // Can be more than one mode and contain both + and -
char exploitchan[] = "#exploit-channel";        // Channel where exploit messages get redirected
char psniffchan[] = "#sniff-channel";   // sniffing channel
char keylogchan[] = "#keylog-channel";  // Channel where keylog messages get redirected
char version[] = "- xerion v2 -";       // !bot version reply
unsigned int daysup = 1;                // minimum number (of days) to show in uptime prefix
char *authost[] = {
        "*@*127.0.0.1",
};
```

A non-technical user might not be familiar with this, but these skills can be learned in a matter of hours. Complications could, however, arise in case of badly programmed malware.

In addition to the compilation skills, the C&C server needs to be set up for hosting. However, this is also no more difficult than setting up regular content management systems (CMS) and can be easily learned. More difficult is hardening these servers against monitoring of researchers and against takeover attempts from competing groups. Different hosting providers offer ready-made solutions for »bulletproof« C&C servers.

A serious difficulty with setting up open-source botnets is that certain components (e. g. exploit packs or binary encryption routines) might not be made publicly available due to their high market value. Developing individual packer (encryption) routines would result in extensive development work which is very susceptible to errors.

The construction of the C&C server is only part of the botnet creation process. Depending on which exploits for host infection are used, the creation of a »sufficient« number of bots can vary. The collection of 10,000 bots can be accomplished within several hours but may also take days and even weeks, although the latter is very rare.

## 5.2 Construction Kits

Compared to open-source botnets, the effort required for setting up botnets with the help of construction kits is almost negligible. Since these kits are typically for sale and paying customers expect return on investment, the kits are developed with user-friendliness in mind. A good example of this is *ZeuS* (and later *SpyEye*). Figure 14 shows screenshots of the *ZeuS* construction kit. It includes a removal tool for the malware itself, thus allowing the cleaning of test infections.

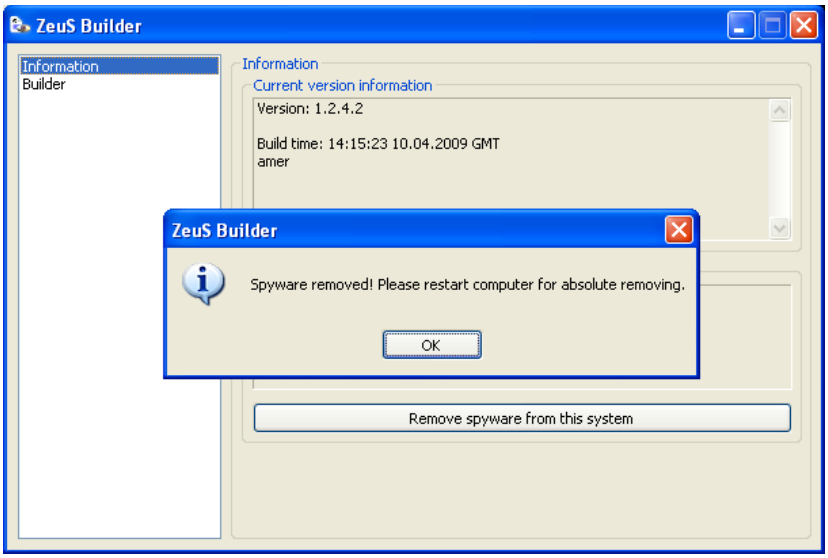


Figure 14: Self-disinfection functionality within the *ZeuS* construction kit.

Figure 15 shows a screenshot of the *Illusion* DDoS botnet construction kit. As can be seen, the client configuration is very straight-forward and can be accomplished with a small number of clicks.

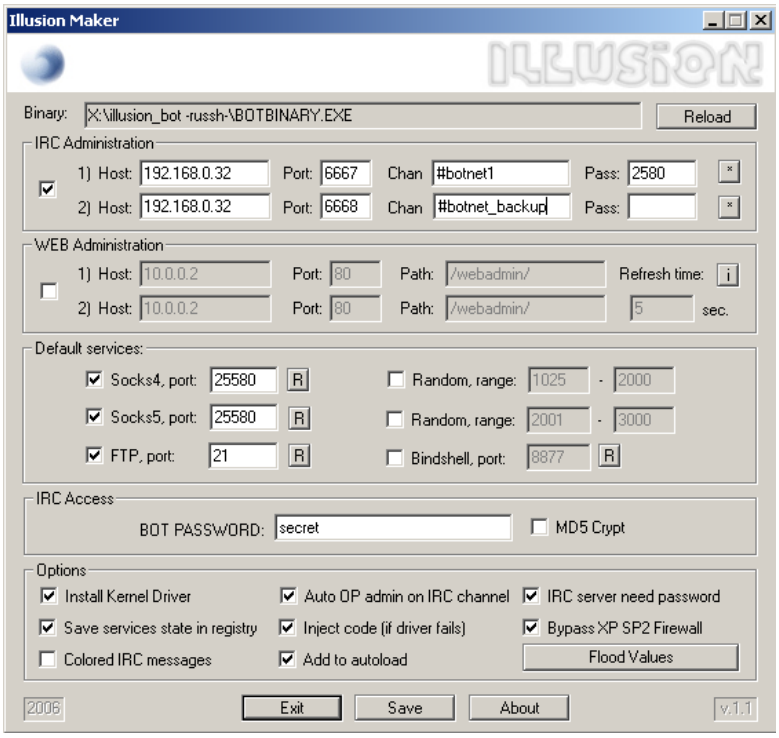


Figure 15: Screenshot of the *Illusion* botnet construction kit.

Especially the high-value banking Trojan horses require no technical skill to set up and manage and can be used by nearly everyone. They are easily configured by either adapting text configuration files or buying ready-made files. Configuration options include Facebook accounts and target banks. However, for most »standard« users, the default configuration is sufficient. Similar to the open-source botnets, gathering of results involves only the setting up of a Web server to which the results are published by the individual bots. Again, the effort is comparable to that of setting up any other CMS and most construction kits provide adequate support.

The construction kits often include the packer with which the malware binaries are encrypted. Depending on the amount of money invested, this could be a dynamic packer in combination with server-side polymorphism (see Section 4.3). Similarly for sale are integrated services that include technical support and hosting which ignores abuse reports. In this case, no skills are required at all for setting up the botnet.

Botnet construction kits do not need to be for sale. Notably, the botnet construction kit *Poison Ivy* [32] (officially a remote administration toolkit) is free of charge. The danger with such

offers (from a botnet operator's point of view) is that construction kit authors have the possibility of inserting backdoors into the C&C server or the malware constructed with these kits. Although principally, this risk is present in purchasable botnet construction kits as well, the probability of this happening is rather small, because vendors selling software with backdoors are likely to be out of business quite quickly.

Similar to the open-source botnets, infection time can vary for botnets made with construction kits. Often, separate infection or exploit kits can be bought along with the C&C construction kit. Figure 16 shows an example of infection statistics for an exploit pack. More information on exploit packs can be found in [23].

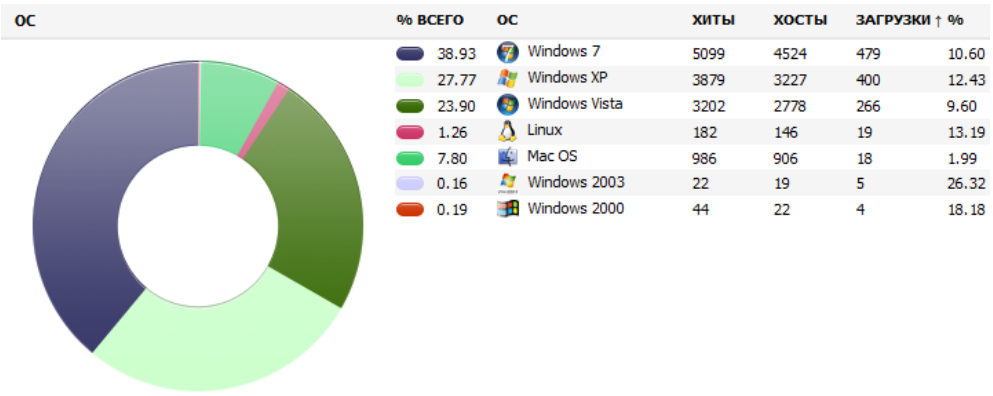


Figure 16: Infection statistics of an exploit kit.

## 5.3 Specialized Botnets

Examples of professionally developed, specialized botnets are *Conficker*, *Ghostnet* and, most recently, *Stuxnet*.

The biggest difficulty in creating and setting up such highly specialized and targeted botnets is the amalgamation of cross-domain know-how. *Stuxnet* targeted industrial control systems for centrifuges used, for example, in the uranium enrichment process. It made use of very advanced and specialized 0-day exploits for Windows. Estimates say that only about 1,000 people world-wide are capable of developing such exploits. Also, detailed knowledge about the targeted critical infrastructures is required to facilitate infection and spreading of malware. While separate expertise in the different areas might be more or less easily accessible, the combination of domains is very difficult.

In addition to this, difficulties also lie in the design and operation of the C&C server (similar to the other classes of botnets). However, since construction kits are generally not available, encountered problems need to be solved without the help of technical support staff. In a sense, the technical skill required here is comparable to that of open-source botnets, with the exception that no malware community is available for support. In most cases, outside help is

not even desirable because it can draw unwanted attention to the botnet activities. In general, it can be expected that developers of specialized botnets will also spend more effort where botnet C&C is concerned.

Moreover, it can be seen that even these kinds of botnets evolve over time; for example, the earliest versions of Conficker used only a single exploit – a Windows server service. Later variants included exploits for the file sharing service and spreading via USB devices.

The time required for the infection of hosts is difficult to estimate in the case of specialized botnets because it depends on the target domain and the employed exploits. However, the purpose of such botnets is often not financial gain but rather the accomplishment of long-term goals such as espionage. Therefore, the infection speed is not of the utmost importance.

## 5.4 Conclusions

From the point of view of setting up botnets, there is an increase in effort from the construction kit-based botnets, to the open-source ones and, most difficult, the specialized ones.

If a construction kit is purchased, possibly along with an infection kit, this normally includes technical and administrative support as well as updates so that the malware is always up-to-date. This ensures that newly discovered vulnerabilities can be exploited in the future, and that weaknesses in the malware itself are removed. In open-source botnets, this has to be done by the botnet developers themselves, although there is a certain level of community support. Also, legacy botnets can often be used for a long time since there are always systems in which vulnerabilities are not patched. Specialized botnets are the most difficult to develop and set up. This is due on the one hand to the required knowledge and skills for target domain exploit development and on the other hand the fact that, in some cases, stealthy operation is essential and no outside help can be used.

# 6 Requirements for Botnet Takedowns

In case the classical botnet countermeasures as described in Section 2.3 can either not be employed or do not work, proactive measures can be used. In this section we examine the requirements necessary for proactively taking down botnets. Such a botnet takedown involves its analysis, finding weaknesses either in the software itself or in the employed communication mechanisms, and the exploitation of these weaknesses. Once control of the botnet has been gained, subsequent actions depend on the nature of the bots. For example, it might be possible to instruct them to perform a self-disinfection, as in the example *Storm* botnet takedown.

In the discussion, we again focus on the three classes of malware first introduced in Section 3: open-source software, botnets based on construction kits and custom-made, specialized botnets.

For the most part, the approaches described in this section are manual processes. Automatic approaches have been investigated and are often possible from a technical point of view. For example, corporations such as *Arbor Networks* [3], communities such as *Shadowserver* [38], or individual efforts such as the *ZeuS Tracker* [48] monitor and track the activities of widespread botnets and could, in principle, disable them automatically. However, legal constraints prevent the application of these methods.

## 6.1 Open-Source Software

The major advantage in the analysis and takedown of open-source malware is that standard code auditing and analysis tools can be used for finding weaknesses in the source code. This task belongs to the standard repertoire of software consulting firms and can thus be easily outsourced.

Similar to regular software, weaknesses or vulnerabilities in malware can be extremely difficult to find, even if the source code is available. It requires creative approaches and out-of-the-box thinking on the part of the analyst. However, statistics show that commercial applications contain up to 50 errors per 1,000 lines of code [18]. Even if only one tenth of these errors can be exploited, the likelihood of finding an exploitable vulnerability in any given piece of malware is still reasonably high if the source code is available.

Unlike regular open-source software, no public repositories are available from which the source code can be freely obtained. Code

is generally only shared within trusted groups. Thus, evolution and extension of these botnets is only possible within such groups.

Depending on the detected vulnerability, different options for countermeasures against the botnet are possible. In some cases it may only help to better understand the piece of malware, e. g. find out the location of the C&C server. In other cases it might be possible to take over the entire botnet. The *RBot* botnet contains an FTP server from which other, newly infected machines download their malware specimens. This FTP server contained a buffer overflow vulnerability that allowed security researchers to completely control the botnet (by controlling individual infected machines) up until a new version of the botnet fixed the vulnerability. Another example is *ZeuS*, in whose case the (open-source) C&C server contained several design flaws. The server was based on PHP and allowed the uploading of arbitrary files [35], including executable PHP files. This design weakness could only be found through careful analysis of data transfer mechanisms and not, for example, by standard penetration techniques such as SQL injection.

The time required for taking down an open-source botnet is difficult to estimate. »Standard« errors such as buffer overflows in unsafe functions can be found with the help of standard software in a matter of minutes. A more realistic estimate is several days for the discovery of an exploitable weakness, because in most cases manual analyses need to be performed as available tools lack sufficient automation. Despite the actual bots being closed-source software, the C&C servers often used to be open-source, e. g. written in PHP. However, this is also changing; newer C&C servers consisting of compiled binaries have been observed (e. g. the *SpyEye* C&C server is written in C). In these more complicated cases, complex analyses are required which may take more several weeks.

Where personnel resources are concerned, a combination of skills in multiple areas is required. On the one hand, »black-hat« skills in penetration testing and exploit development are required. On the other hand, reverse engineering expertise is required with regard to design analysis, i. e, the actual takedown of the botnet.

Automation of analysis is possible insofar as monitoring systems are employed by groups like *Shadowserver* [38] and firms like *Arbor Networks* [3]. This makes the repeated analysis of similar malware samples easier but does not completely relieve the need for manual approaches.

## 6.2 Construction Kits

The big advantage with botnet construction kits is that they include the C&C server creation mechanisms. Thus, their purchase enables an in-depth analysis of all botnet components, the bot and the

server, as well as their interactions. Also, since these types of botnets are designed to be reused, a detected flaw or weakness in one incarnation can most probably be exploited in other incarnations as well.

A disadvantage for the analyst is that because these botnets are, in a sense, commercial software, multiple different versions can be in circulation at any given time. Also, regular patches and/or service packs are released by the developers to fix bugs exploited by security analysts. Some construction kit-based botnets offer the incorporation of external tools, e. g. the *RDG Crypter* described earlier, to protect against analysis and reverse engineering. Some construction kits, e. g. *ZeuS*, already contain their own protective measures without having to resort to 3rd-party tools.

The skills and time required for finding exploitable flaws in these kinds of malware are comparable to those described in the previous section. However, the unavailable source code makes additional reverse engineering skills necessary and greatly complicates deep analysis of the malware. For a description of the complete dissection and exploitation of a buffer overflow weakness in the C&C server of the *PoisonIvy* construction kit, please refer to [9].

## 6.3 Specialized Botnets

Professionally developed, specialized botnets are solutions to very specific problems such as, for example, espionage (e. g. *Ghostnet*) or sabotage (e. g. *Stuxnet*). Because of this, developers of these botnets spend substantial efforts in securing their software.

A disadvantage with these kinds of botnets is that no information about the internals of the C&C server is available. Its location can often be determined by analyzing either the bots themselves or the communication mechanisms of those bots. A common point for weaknesses is the incorrect use of standard cryptographic protocols. However, it first needs to be determined that these protocols are actually being used. Thus, reverse engineering skills are required. Flaws cannot only be found in the code itself but rather in the design of the software. This requires out-of-the-box thinking on the part of the analyst and is well outside the scope of standard sandboxing tools.

If the location of the C&C server is known, standard penetration techniques may be used to gain access to and control of that system. 0-day exploits for server software are helpful in this case but these are extremely rare. Standard starting points for vulnerability searches are, for example, buffer overflows in unsafe functions or badly seeded random data generators which always produce the same sequence of data. Standard penetration techniques were successfully used in the case of infiltrating the *Waledac* botnet.

The required time is difficult to predict. Standard server penetration can typically be achieved within half a day. The preceding activities vary with the complexity of the infiltrated botnet and can take from several weeks to months. A cautious estimate is a time span of two weeks for the infiltration. This is under the conditions that sufficient weaknesses are present, which must be exploitable, and that only standard anti-reverse engineering mechanisms are used which can be circumvented with standard tools, possibly augmented with few manual customizations. In general, an analysis is possible after 1-2 days, where a preceding step is the setup of a laboratory environment.

The 2010 *Waledac* takedown by Microsoft, Shadowserver, the University of Bonn, TU Wien, the University of Washington and the University of Mannheim required 4-5 days of preparatory work by one person. The analysis was complicated by a well-secured C&C server which could only be determined by registering an intermediate node within the botnet and performing extensive traffic analysis. The peer-to-peer botnet *Storm* required half a year's part-time work of multiple researchers whereas flaws in the *Conficker* botnet could be identified within three weeks.

## 6.4 Conclusions

The time involved with botnet analysis can be quite lengthy. The process includes bringing together experts in certain fields, obtaining botnet samples, setting up an analysis environment, finding exploitable weaknesses, and developing an actual counter strategy. Because of this time span, it is generally not possible to take down an attacking botnet unless it has been under observation and undergone intensive analysis before the attack.

One of the difficulties is to determine which botnet is actually attacking. This is not readily apparent, for example, in the case of simple DDoS attacks. However, if the botnet can be determined, and it is already under observation by the botnet research community, it can principally be taken down instantly. But this is only the case in ~20 % of situations.

# 7 Conclusion and Outlook

## 7.1 Summary

This study has provided an overview of current botnet technology as well as current trends in their development and surroundings.

While early malware relied on users executing files sent via e-mail, it has recently been observed that there is an increase in targeted attacks. Here, personalized e-mails or social network messages are sent to users inviting them to visit Web sites or open tailored infected documents. Both attack vectors make use of weaknesses in client-side software such as Web browsers (drive-by downloads on infected Web sites) or document viewer applications (e. g. PDF readers). This is a trend away from the exploitation of server software as used, for example, by early Internet worms.

There is an increase in professionalization both on the development and the distribution side of the malware business. Because of the large amount of money to be earned with the sale and renting out of botnets, developers and operators increasingly see their work as a financial investment. Thus, they are trying – and succeeding – to stay one step ahead of the anti-virus industry and security researchers. Because of their illegality, these activities have, for the most part, moved to countries with little to no criminal prosecution such as Eastern Europe and parts of Asia.

The commercially available botnets seldom require a great deal of technical skills to set up. Beside the technical support that some malware authors offer, there are often all-inclusive packages, which contain the botnet command-and-control server, easy-to-use configuration options, infection kits for targeted attacks, technical and administrative support as well as bulletproof hosting of the server to ensure that it cannot be taken down easily.

Beside the classical botnet mitigation techniques, taking down botnets is often possible because – similar to regular software – they contain exploitable errors. For this, experts are required in the fields of reverse engineering, penetration testing, design weakness analysis, exploit design, and, in general, software development. These are typically senior researchers with multiple years of experience in their specific fields.

Although technically possible, the associated time required for analysis and exploit development ranges from several days to weeks. Thus, unless this work has already been done, it is generally not possible to instantly take down an attacking new botnet.

## 7.2 Conclusion and Outlook

Currently, botnet developers and operators are ahead in the arms race against the anti-virus industry and law enforcement agencies. However, the anti-botnet activities of law enforcement, AV vendors and academia are not as aggressive as it could be. This is due to the fact that the amount of money lost because of botnet activity (e. g. because of DDoS extortion or banking/credit card fraud) is relatively small. Also, only a few dozen firms and research groups actively study and monitor the activity of the different botnets.

However, studies show that especially banks are recently losing more and more money. Thus, it stands to reason that with increasing losses, the funds made available for botnet research, mitigation and countermeasures will increase as well.

It is likely that there will be more offensive botnet takedown attempts. If successful, these, however, will prompt further developments on the side of the botnet developers. This is exemplified in the case of the takedown of McColo, Web hosting service provider, through which most of the world-wide spam e-mails were sent. After the takedown, the total amount of spam was reduced by up to 70 %. However, this slowly increased afterwards as spam senders switched to other providers. Also, after the takedown of the *Storm* botnet, *Waledac* appeared. This was followed by *Storm 2* after Waledac was taken down. Incidentally, at first, around 80 % of the code base of *Storm 2* was identical to that of *Storm*, indicating that there was active development of existing software towards new variants.

Furthermore, we anticipate the development of better AV and security tools, including automated analysis tools. Conversely however, the anti-analysis protection attempts of botnet developers will also become more advanced.

Public awareness of malware will need to be improved in the future, with ISPs taking on a more active role. In Germany, the »Botfrei« (bot-free) initiative [2] was started, offering information and a malware cleaning tool for end users, and there are similar initiatives in Japan, Turkey, Korea, Australia and the Netherlands. Also, there will be more botnet-related discussion and collaboration on a political level. First evidence of this is the establishment of the *European Network and Information Security Agency* (ENISA). Collaboration is also evident in the OECD context.

In general, however, it can be expected that the arms race between the security industry and botnet developers will get more aggressive in the mid-term future.

# References

[1] *AgoBot source code*. Available online: http://rapidshare.com/files/28952139/agobot.rar, accessed December 2010.

[2] *Anti-Botnet-Advisory Centre*. Available online: https://www.botfrei.de/en/index.html, accessed December 2010.

[3] *Arbor Networks homepage*. Available online: http://www.arbornetworks.com, accessed November 2010.

[4] *avhide.com homepage*. Available online: http://www.avhide.com, accessed November 2010.

[5] Butler, E. *Firesheep homepage*. Available online: http://codebutler.com/firesheep, accessed November 2010.

[6] *Cash Paradise University homepage*. Available online: http://www.cpuonline.in, accessed November 2010.

[7] Celeda, P., Krejci, R., Vykopal, J., Drasar, M. *Embedded Malware – An Analysis of the Chuck Norris Botnet*. In: Proceedings of the 6th European Conference on Computer Network Defense EC2ND, Berlin, Germany, October, 2010.

[8] *CVE-2010-2568*. Available online: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-2568, accessed November 2010.

[9] Dereszowski, A. *Targeted attacks: From being a victim to counter attacking*. March 2010. Available online: http://www.signal11.eu/en/research/articles/targeted_2010.pdf, accessed December 2010.

[10] Digital Degenerate. *Zeus Botnet Targets Facebook*. Available online: http://blog.appriver.com/2009/10/zeus-botnet-targets-facebook.html, accessed November 2010.

[11] Fielding, R. et al. *Hypertext Transfer Protocol – HTTP/1.1, RFC 2616*. May 1993. [Online]. Available: http://www.ietf.org/rfc/rfc2612.txt.

[12] Fisher, D. *Storm, Nugache lead dangerous new botnet barrage*. Available online: http://searchsecurity.techtarget.com/news/article/0,289142,sid14_gci1286808,00.html, accessed December 2010.

[13] F-Secure Weblog. *Mr. Anderson pleads guilty*. Available online: http://www.f-secure.com/weblog/archives/00002054.html, accessed November 2010.

[14] Gailly, J., Adler, M. *zlib Compression Library*. Available online: http://www.zlib.net, accessed November 2010.

[15] Hacker the Dude. *Torpig Domain Generator: Hackers Using Twitter Trending Topics*. Available online: http://hackerthedude.blogspot.com/2009/12/torpig-domain-generator-hacker-using.html, accessed December 2010.

[16] Helsingin Sanomat. *Virtual harassment, but for real*. Available online: http://www.hs.fi/english/article/Virtual+harassment+but+for+real+/1135227099868, accessed December 2010.

[17] Higgins, K. J. *Zeus Attackers Deploy Honeypot Against Researchers, Competitors*. Available online: http://www.darkreading.com/insider-threat/167801100/security/attacks-breaches/228200070/index.html, accessed December 2010.
[18] Hoglund, G., McGraw, G. *Exploiting Software – How to Break Code*. Addison Wesley, 2004.
[19] Kleissner, P. *AV Tracker homepage*. Available online: http://www.avtracker.info, accessed November 2010.
[20] Krebs, B. *Krebs on Security: SpyEye v. ZeuS Rivalry Ends in Quiet Merger*. Available online: http://krebsonsecurity.com/2010/10/spyeye-v-zeus-rivalry-ends-in-quiet-merger, accessed November 2010.
[21] Leder, F. *Waledac is wishing merry Christmas*. The Honeynet Project. Available online: http://www.honeynet.org/node/325, accessed December 2010.
[22] Leder, F., Werner, T., Martini, P. *Proactive Botnet Countermeasures – An Offensive Approach*. In: C. Czosseck, K. Geers (Eds.), »The Virtual Battlefield: Perspectives on Cyber Warfare«, IOS Press, 2009.
[23] M86 Security Labs. *Web Exploits: There's an App for That*. Technical Report. Available online: http://www.m86security.com/documents/pdfs/security_labs/m86_web_exploits_report.pdf, accessed November 2010.
[24] Menn, J. *Fatal System Error: The Hunt for the New Crime Lords Who are Bringing Down the Internet*. PublicAffairs, 2010.
[25] *Metasploit - Penetration Testing Resources*. Available online: http://www.metasploit.com, accessed November 2010.
[26] Microsoft. *Malicious Software Removal Tool*. Available online: http://www.microsoft.com/security/malwareremove/default.aspx, accessed December 2010.
[27] *NoVirusThanks homepage*. Available online: http://www.novirusthanks.org, accessed November 2010.
[28] Oikarinen, J., Reed, D. *Internet Relay Chat Protocol, RFC 1459*. May 1993. [Online]. Available: http://www.ietf.org/rfc/rfc1459.txt.
[29] *OpenSSL: The Open-Source Toolkit for SSL/TLS*. Available online: http://www.openssl.org, accessed November 2010.
[30] PC Plus. *Botnets Explained*. Available online: http://pcplus.techradar.com/feature/features/botnets-explained-30-09-10, accessed November 2010.
[31] PC World Business Center. *Spanish Police Take Down Massive Mariposa Botnet*. Available online: http://www.pcworld.com/businesscenter/article/190634/spanish_police_take_down_massive_mariposa_botnet.html, accessed November 2010.
[32] *Poison Ivy homepage*. Available online: http://www.poisonivy-rat.com, accessed November 2010.
[33] *Prolexic Technologies homepage*. Available online: http://www.prolexic.com, accessed November 2010.
[34] *RDG Tejon Crypter homepage*. Available online: http://rdgsoft.blogspot.com, accessed November 2010.
[35] Rios, B. *Turning the Tables – Part I*. Available online: http://xs-sniper.com/blog/2010/09/27/turning-the-tables/, accessed December 2010.

[36] Rivest, R. L. et al. *The MD6 hash function – A proposal to NIST for SHA-3*. Technical Report. Massachusetts Institute of Technology, Cambridge, MA, USA, April 2009.
[37] Saint-Andre, P. *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence, RFC 1459*. October 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3921.txt.
[38] *Shadowserver homepage*. Available online: http://www.shadowserver.org, accessed November 2010.
[39] Shadowserver. *60-Day Virus-Stats*. Available online: http://www.shadowserver.org/wiki/pmwiki.php/Stats/Virus60-DayStats, accessed 8 December 2010.
[40] Stevens, K., Jackson, D. *ZeuS Banking Trojan Report*. Available online: http://www.secureworks.com/research/threats/zeus, accessed December 2010.
[41] Stover, S., Dittrich, D., Hernandez, J., Dietrich, S. *Analysis of the Storm and Nugache trojans: P2P is here*. »;login:« Vol. 32, No. 6, Usenix.org, 2007. Available online: http://www.usenix.org/publications/login/2007-12/pdfs/stover.pdf, accessed November 2010.
[42] Villeneuve, N. *KOOBFACE: Inside a Crimeware Network*. November 2010. Available online: http://www.infowar-monitor.net/reports/iwm-koobface.pdf, accessed December 2010.
[43] *VirusTotal - Free Online Virus, Malware and URL Scanner*. Available online: http://www.virustotal.com, accessed November 2010.
[44] *VirusTrap homepage*. Available online: http://www.virus-trap.org, accessed November 2010.
[45] *VMProtect homepage*. Available online: http://vmpsoft.com, accessed November 2010.
[46] Websense, Inc. *Websense 2010 Threat Report*. Technical Report. Available online: http://www.websense.com/content/threat-report-2010-introduction.aspx, accessed November 2010.
[47] ZDNet. *Dutch police shut down Bredolab botnet*. Available online: http://www.zdnet.com/blog/security/dutch-police-shut-down-bredolab-botnet/7573, accessed November 2010.
[48] *ZeuS Tracker homepage*. Available online: https://zeustracker.abuse.ch, accessed November 2010.