User Assisted Botnet Detection
Final Report
Daniel Song (dws2127)

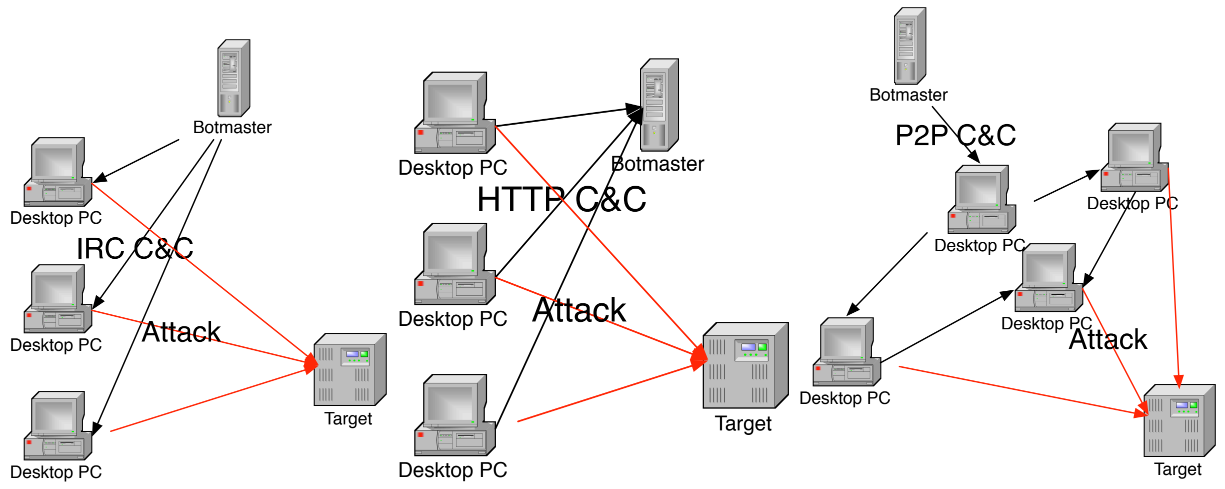
1. Introduction

Increasing cyber-threat to computer networks and systems made host-based intrusion detection system a vital component. Among the threats, botnets are regarded as the most dangerous security threat these days. Moreover, traditional host-based intrusion detection system cannot detect newly developed botnets.

We present a firm botnet detector system called "User Assisted Botnet Detection System" (UABDS) that uses user perception to detect newly developed botnets. Our system will oversee the traffic, detect suspicious activity, and then ask the user whether the traffic is legitimate or not. Accumulated data of user feedback will be used as to detect a newly developed botnets.

2. Botnet

Botnet is different from other kinds of malware in that it has a unique feature called command and control channel. Bots periodically communicate with Botmaster using command and control channel. There are three kinds of command and control channel: IRC C&C, HTTP C&C, and P2P C&C.



IRC C&C: Botmasters can interact with botnets in realtime using IRC PRIVMSG. Because botmasters contact bots to give the command, it is considered as push style.

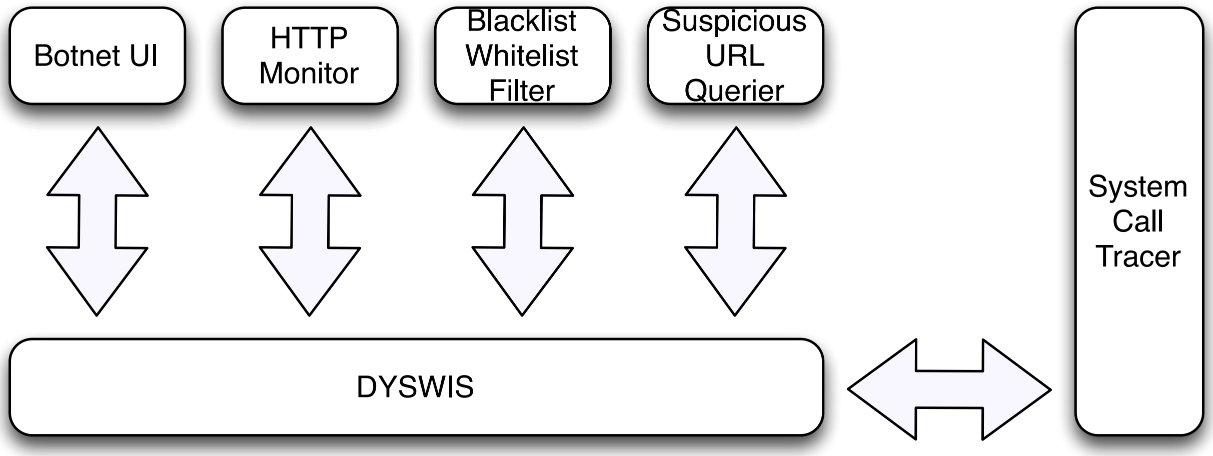
HTTP C&C: Bots periodically contact botmasters to obtain the command. Is is considered as pull style.

P2P C&C: Bots uses P2P to share the botmaster's command.

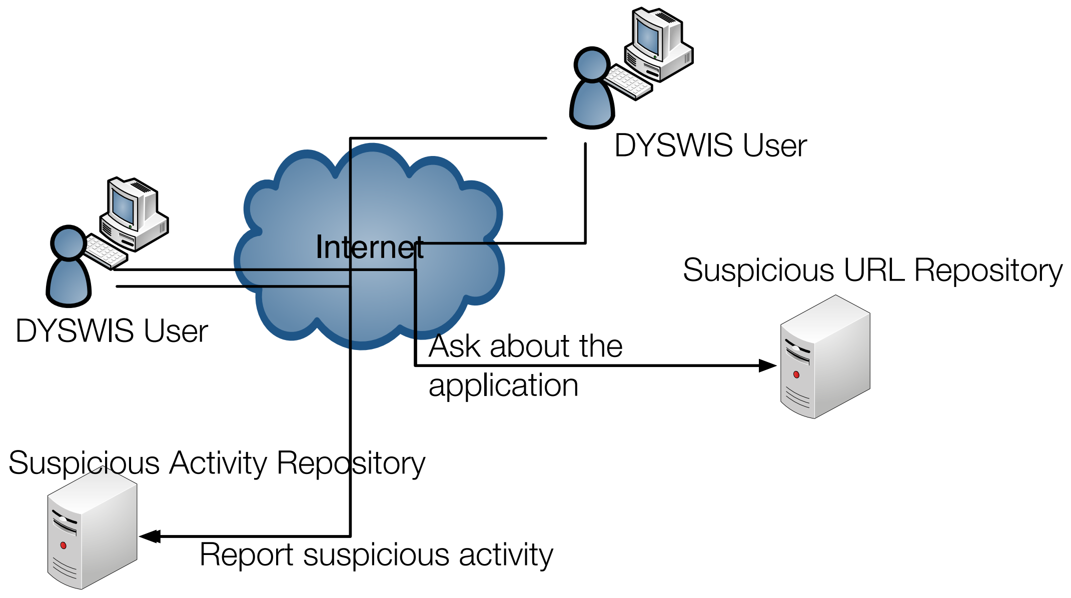
In this project we tackle HTTP C&C which can be easily detected by overseeing port 80.

3. System Architecture

UABDS can be incorporated with any existing host based intrusion detection system. In this project, we will build on top of DYSWIS framework for botnet detection. 5 new modules have been added to create the botnet detector: Botnet UI, HTTP Monitor, Blacklist Whitelist Filter, Suspicious URL Querier, and System Call Tracer.

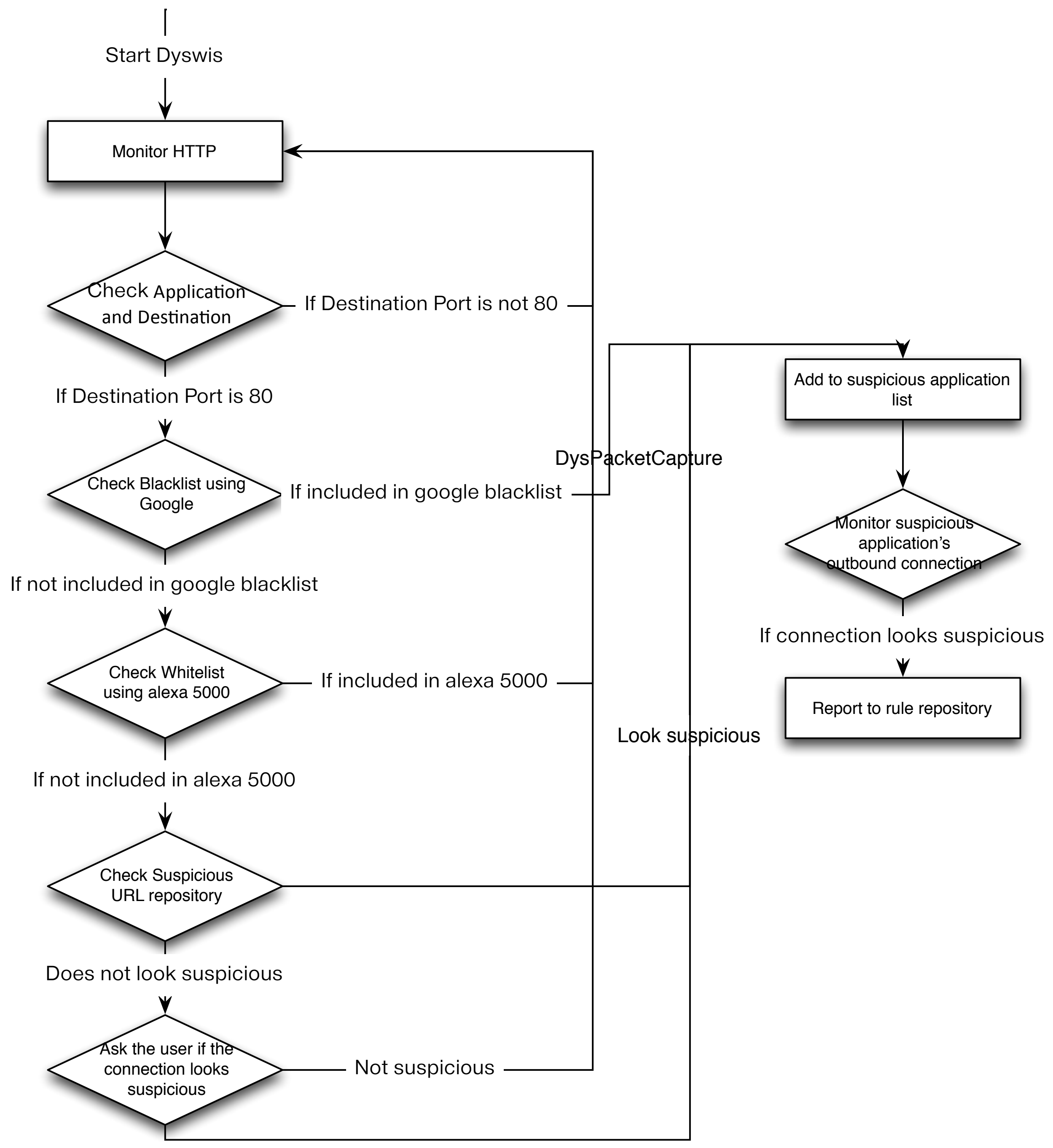


There also is a Suspicious URL Repository that keeps suspicious url. If the UABDS user thinks the destination URL look suspicious, it is added to the Suspicious URL Repository to share this information with other UABDS users. We keep suspicious destination and not suspicious application because some malwares uses randomized application names.


DYSWIS User
DYSWIS User
Internet
Suspicious URL Repository
Ask about the application
Suspicious Activity Repository
Report suspicious activity

4. Application Flow and Details

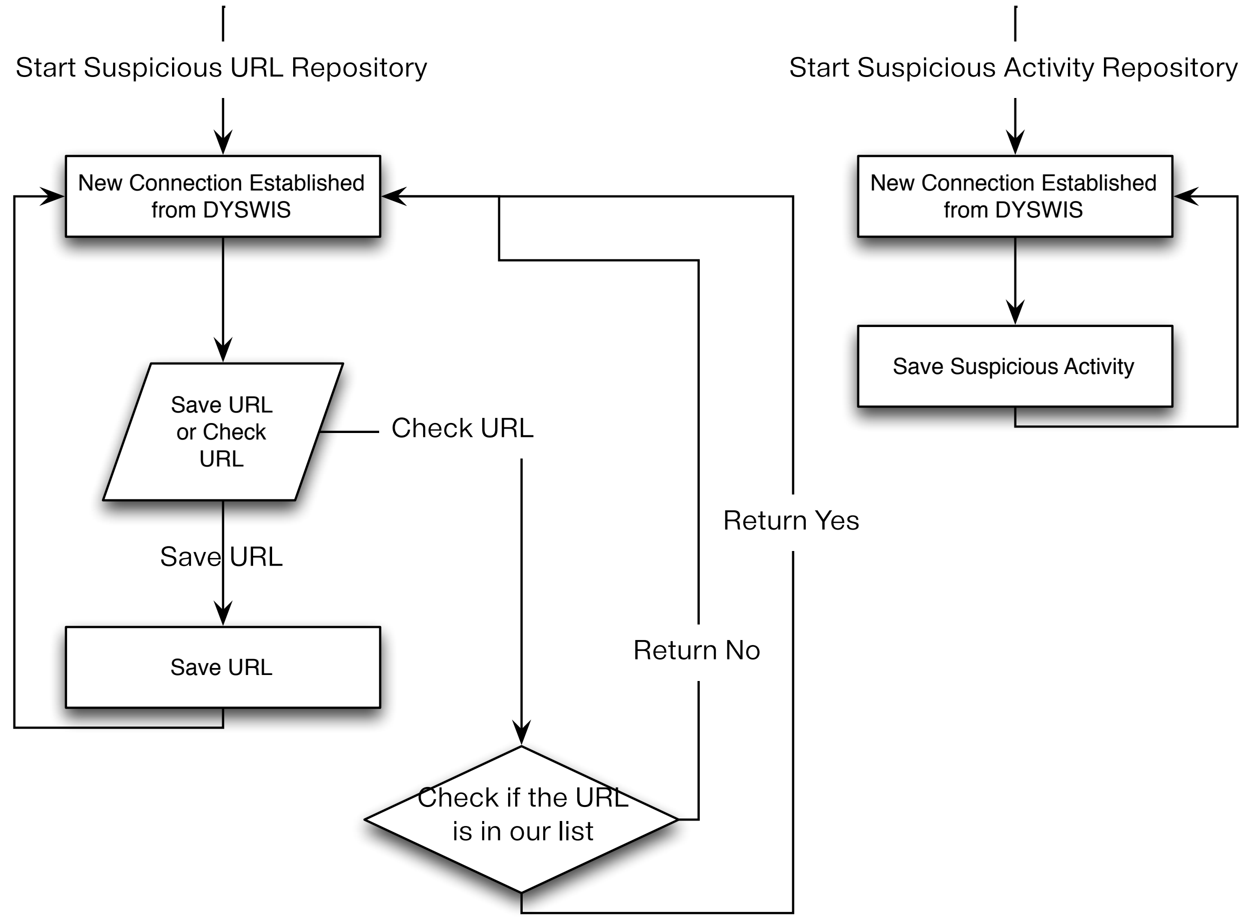
Botnet Detector


Start Dyswis
Monitor HTTP
Check Application and Destination
If Destination Port is not 80
If Destination Port is 80
Check Blacklist using Google
If included in google blacklist
DysPacketCapture
If not included in google blacklist
Check Whitelist using alexa 5000
If included in alexa 5000
If not included in alexa 5000
Check Suspicious URL repository
Look suspicious
Does not look suspicious
Ask the user if the connection looks suspicious
Not suspicious
Add to suspicious application list
Monitor suspicious application's outbound connection
If connection looks suspicious
Report to rule repository

UABDS uses several method to concentrate on the suspicious looking applications. First, since HTTP C&C uses HTTP to retrieve the command periodically, our module monitors applications that uses HTTP using HTTP Monitor module. Then we check if the destination is in the blacklist using google safe browsing. If the destination is in the blacklist, add the application to suspicious application list. If it is not in the blacklist check if it is in the whitelist using Alexa top 5000 sites. This blacklist whitelist filtering is done by Blacklist Whitelist Filter module.

If the destination is not in the whitelist, the module will ask the global Suspicious URL Repository, which keeps the blacklist that is added by the UABDS users. Finally, if it is not in the Suspicious URL Repository, we ask the user if the activity of this application looks suspicious. If the user thinks the destination looks suspicious, we add the destination to the Suspicious URL Repository. After an application is regarded as suspicious, we monitor every outbound connection of the suspicious application by tracing system calls of the application. If it is a malware, it will do various things like sending spams and user can report this activity to suspicious activity repository.

Suspicious URL Repository and Suspicious Repository



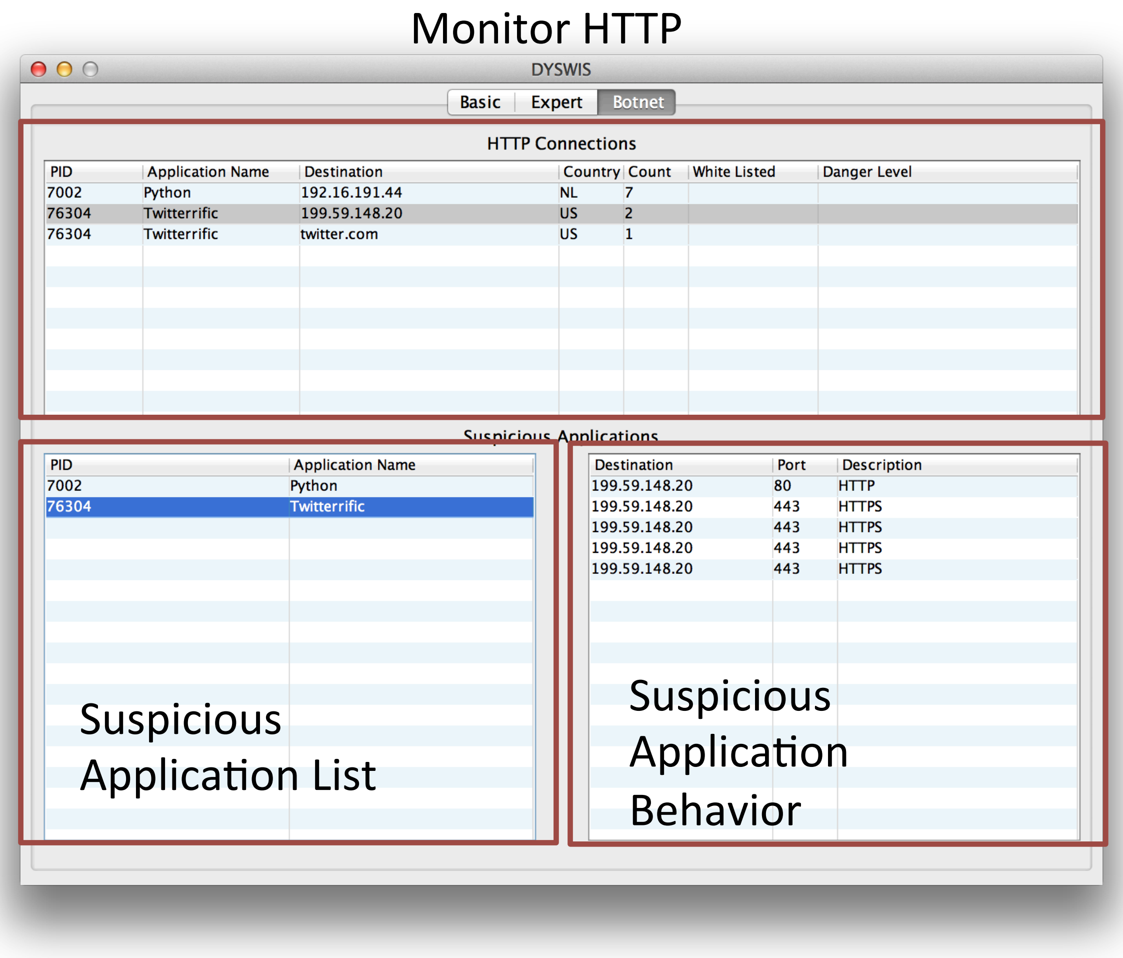
Start Suspicious URL Repository
New Connection Established from DYSWIS
Save URL or Check URL
Check URL
Save URL
Save URL
Return Yes
Return No
Check if the URL is in our list
Start Suspicious Activity Repository
New Connection Established from DYSWIS
Save Suspicious Activity

There is two central repositories that manages suspicious URL and suspicious activity. The Suspicious URL Repository will do two things. If the UABDS asks whether the repository has specific URL, the repository will respond by yes or no. The UABDS can also add suspicious URL to the Suspicious URL Repository.

The Suspicious Activity Repository will keep track of the botnets activities so that network admin can check the suspicious application's activities. For example, if a application is sending spam, user can report to the Suspicious Activity Repository and the admin can block that host from sending emails.

5. UI

Our botnet detector UI has three tables that shows HTTP connections, suspicious application list, and suspicious application behavior. HTTP connections table shows applications that uses HTTP connection. It shows PID, application name, destination, country, whether it is black or whitelisted, number of connections, and danger level.



HTTP Connections

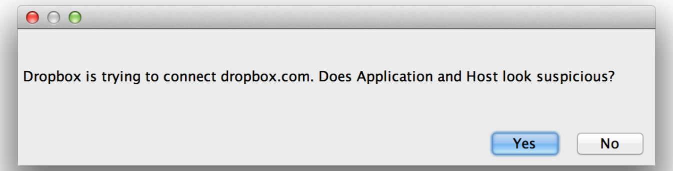
PID	Application Name	Destination	Country	Count	White Listed	Danger Level
7002	Python	192.16.191.44	NL	7		
76304	Twitterrific	199.59.148.20	US	2		
76304	Twitterrific	twitter.com	US	1		

Suspicious Applications

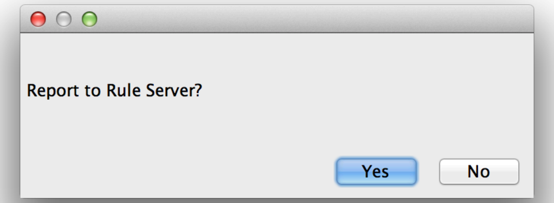
PID	Application Name
7002	Python
76304	Twitterrific

Destination	Port	Description
199.59.148.20	80	HTTP
199.59.148.20	443	HTTPS
199.59.148.20	443	HTTPS
199.59.148.20	443	HTTPS
199.59.148.20	443	HTTPS

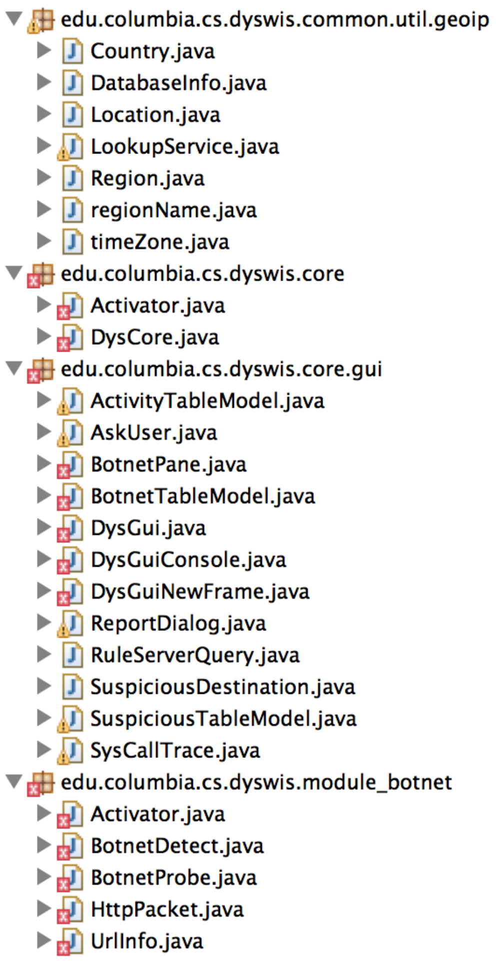
If it is in the google blacklist, it automatically adds the application to suspicious application list. Then the module will start monitoring the applications' system calls to detect every outbound connection of the application. If it is not in the blacklist, whitelist or the Suspicious URL Server, the module will ask the user whether the connection is legitimate.


Dropbox is trying to connect dropbox.com. Does Application and Host look suspicious?

If the user clicks yes, the application is added to suspicious application list and the destination is added to the Suspicious URL Server. When application is added to the suspicious application, the user can see the various activity of the application by selecting the application it in the suspicious application list. If the application is doing suspicious activity such as sending email, the user can report to the rule server by double clicking it.


Report to Rule Server?

6. Implementation Detail


```
edu.columbia.cs.dyswis.common.util.geoip
    Country.java
    DatabaseInfo.java
    Location.java
    LookupService.java
    Region.java
    regionName.java
    timeZone.java
edu.columbia.cs.dyswis.core
    Activator.java
    DysCore.java
edu.columbia.cs.dyswis.core.gui
    ActivityTableModel.java
    AskUser.java
    BotnetPane.java
    BotnetTableModel.java
    DysGui.java
    DysGuiConsole.java
    DysGuiNewFrame.java
    ReportDialog.java
    RuleServerQuery.java
    SuspiciousDestination.java
    SuspiciousTableModel.java
    SysCallTrace.java
edu.columbia.cs.dyswis.module_botnet
    Activator.java
    BotnetDetect.java
    BotnetProbe.java
    HttpPacket.java
    UrlInfo.java
```

Left is the created or edited source code of the botnet module. “geoip”, “core”, “gui”, and “module_botnet” package has been created or edited.

“edu.columbia.cs.dyswis.module_botnet” package is the HTTP Monitor module which uses DysPacketCapture.java to detect HTTP packet. DysPacketCapture.java is implemented using jpcap library which is a packet capture library using Java. Also, HTTP Monitor retrieves packet’s application name and PID using “lsof” command. “lsof” lists open files so that the module can backtrace the packet’s application name and PID. “BotnetDetect.java” also checks if the destination is in Google blacklist or Alexa whitelist.

“edu.columbia.cs.dyswis.common.util.geoip” is the module that detects the destination country using MaxMaind’s geoip data. It helps the user to choose whether the connection is suspicious or not.

“edu.columbia.cs.dyswis.core.gui” package is the gui module. “SysCallTrace.java” is the module that monitors application’s every outbound connection using strace of Linux and dtrace of Mac.

7. Future Work

Our module detects botnet by overseeing HTTP C&C. We can easily expand our work by adding IRC C&C detector and P2P C&C detector. People rarely uses IRC these days so we can easily detect suspicious looking IRC protocol. However, P2P C&C is more complicated and our module may need some modification since P2P C&C bots does not have central server that can be regarded as suspicious url.

Our module regards a URL as suspicious if the user regards it as a suspicious URL. However, user's answer may not be true. User who is not familiar with the computer can answer with wrong answers. To prevent this error, we need a statistical method to find suspicious destination. That is, we should regard a URL as suspicious URL if many people thinks it is a suspicious URL. We will need to experiment to determine how many is "many people".

Our module need to be deployed on many machines to collect suspicious URL list. Our module is deployed on a few machines and it is not enough to retrieve diverse activity of the users. As more and more users participate, our module will improve.

8. Reference

1. Seungwon Shin, Raymond Lin, Guofei Gu. "Cross-Analysis of Botnet Victims: New Insights and Implications." In Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection (RAID 2011), Menlo Park, California, September 2011.
2. Junjie Zhang, Xiapu Luo, Roberto Perdisci, Guofei Gu, Wenke Lee and Nick Feamster. "Boosting the Scalability of Botnet Detection Using Adaptive Traffic Sampling." In Proceedings of 2011 ACM Symposium on Information, Computer and Communications Security (ASIACCS'11), Hong Kong, March 2011.
3. Guofei Gu, Vinod Yegneswaran, Phillip Porras, Jennifer Stoll, and Wenke Lee. "Active Botnet Probing to Identify Obscure Command and Control Channels." In Proceedings of 2009 Annual Computer Security Applications Conference (ACSAC'09), Honolulu, Hawaii, December 2009.
4. Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. "BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection." In Proceedings of the 17th USENIX Security Symposium (Security'08), San Jose, CA, 2008.
5. Guofei Gu, Junjie Zhang, and Wenke Lee. "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic." In Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08), San Diego, CA, February 2008.
6. David Dagon, Guofei Gu, Chris Lee, and Wenke Lee. "A Taxonomy of Botnet Structures." In Proceedings of the 23 Annual Computer Security Applications Conference (ACSAC'07), Miami Beach, FL, December 2007.
7. Robin Sommer, Vern Paxson. "Outside the Closed World: On Using Machine Learning For Network Intrusion Detection." 2010 IEEE Symposium on Security and Privacy.