دانشگاه شهید بهشتی
دانشکده مهندسی برق و کامپیوتر

شناسایی Botnet بر اساس ناهنجاری در رفتار ترافیک شبکه

پایان‌نامه کارشناسی ارشد مهندسی کامپیوتر
گرایش نرم‌افزار

استاد راهنما:
دکتر مقصود عباسپور

توسط:
سجاد ارشد

استاد مشاور:
دکتر مهدی خرازی

بهمن ۱۳۸۹

دانشگاه شهید بهشتی
دانشکده مهندسی برق و کامپیوتر

پایان‌نامه کارشناسی ارشد مهندسی کامپیوتر – گرایش نرم‌افزار
تحت عنوان:

شناسایی Botnet بر اساس ناهنجاری در رفتار ترافیک شبکه

در تاریخ پایان‌نامه دانشجو، سجاد ارشد، توسط کمیته تخصصی داوران مورد بررسی و تصویب نهائی قرار گرفت.

۱ استاد راهنما: نام و نام خانوادگی دکتر مقصود عباسپور امضاء

۲ استاد مشاور: نام و نام خانوادگی دکتر مهدی خرازی امضاء

۳ استاد داور (داخلی): نام و نام خانوادگی دکتر فرشاد صفایی امضاء

۴ استاد داور (خارجی): نام و نام خانوادگی دکتر احمد خونساری امضاء

۵ نماینده تحصیلات تکمیلی: نام و نام خانوادگی دکتر اسلام ناظمی امضاء

با تشکر از همه اساتید و دوستانی که بدون یاری
ایشان انجام این پایان‌نامه امکان‌پذیر نبود. به ویژه
اساتید گرامی دکتر مقصود عباسپور و دکتر مهدی خرازی و
دوست خوبم هومن صنعتکار که همواره حامی اینجانب بودند.

این پایان‌نامه تحت حمایت مادی و معنوی

مؤسسه تحقیقات ارتباطات و فناوری اطلاعات

با شماره قرارداد ۸۹۷۴/۵۰۰ مورخ ۸۹/۶/۳۰ می‌باشد.

به نام خدا

نام و نام خانوادگی: سجاد ارشد

عنوان پایان‌نامه: شناسایی Botnet بر اساس ناهنجاری در رفتار ترافیک شبکه

استاد/اساتید راهنما: دکتر مقصود عباسپور

اینجانب سجاد ارشد تهیه‌کننده پایان‌نامه کارشناسی ارشد حاضر خود را ملـزم بـه حفـظ امانـت‌داری و قـدردانی از زحمات سایر محققین و نویسندگان بنابر قانون کپی رایت می‌دانم. بدین وسیله اعـلام مـی‌نمـایم کـه مسـئولیت کلیـه مطالب درج شده با اینجانب می‌باشد و در صورت استفاده از شکل‌ها، جدول‌ها و مطالب سایر منابع، بلافاصله مرجـع آن ذکر شده و سایر مطالب از کار تحقیقاتی اینجانب استخراج گشته است و امانت‌داری را به صورت کامل رعایت نمـوده‌ام. در صورتی که خلاف این مطلب ثابت شود، مسئولیت کلیه عواقب قانونی با شخص اینجانب می‌باشد.

نام و نام خانوادگی دانشجو: سجاد ارشد

امضاء و تاریخ:

تقدیم به مادر عزیزم

فهرست مطالب

فهرست شکل‌ها

فهرست جدول‌ها

چکیده

اکثر حملات و فعالیت‌های شیادانه در اینترنت توسط بدافزار[1]ها صورت می‌گیرد. به طور خاص، Botnetها به عنوان یکی از روش‌های اساسی برای حملات اینترنتی شناخته شده‌اند. Botnet شبکه‌ای از کامپیوترهای مورد سوء استفاده قرار گرفته شده[2] (Botها) است که از طریق یک کانال دستور و کنترل[3]، تحت کنترل Botmaster می‌باشند. یک Botnet معمولاً شامل ده‌ها تا صدها هزار Bot است، اما بعضی از Botnetها دارای میلیون‌ها Bot می‌باشند. Botnetها در حال حاضر برای حملات DDoS[4]، ارسال ایمیل‌های اسپم، مدیریت سایت‌های کلاهبرداری[5] و سرقت اطلاعات استفاده می‌شوند. با توجه به اینکه حملات انجام شده توسط Botهای یک Botnet به طور همزمان و هماهنگ انجام می‌گیرد، این حملات دارای قدرت زیادی می‌باشند و به همین خاطر، امروزه Botnetها به عنوان بزرگترین تهدید برای امنیت اینترنت شناخته می‌شوند. برای مقابله با این تهدید احتیاج به روش‌های شناسایی بهتری است. در این پایان‌نامه، ما روی مسئله شناسایی Botnet در شبکه‌های بزرگ تمرکز داریم. ما یک سیستم کاملاً مبتنی بر ناهنجاری[6] معرفی می‌کنیم که هیچ نوع فرضی در مورد امضاء[7] Botها و پروتکل کانال‌های دستور و کنترل و آدرس سرورهای دستور و کنترل Botnetها نمی‌کند. ابتدا خصوصیات ذاتی Botnetها را بررسی می‌کنیم. Botها باید به دستورات دریافت شده واکنش نشان دهند و Botهای متعلق به یک Botnet واکنش‌های مشابهی دارند. روش ما کامپیوترهای با رفتارهای مشابه در پنجره‌های زمانی[8] مختلف را خوشه‌بندی[9] می‌کند و با ایجاد همبستگی[10] بین این خوشه‌ها، کامپیوترهای آلوده را شناسایی می‌کند. ما سیستم BotGrabber را بر اساس این روش پیاده‌سازی کردیم و آن را با استفاده از ترافیک واقعی شبکه شامل ترافیک نرمال و ترافیک Botnet ارزیابی کردیم. نتایج نشان می‌دهند که BotGrabber دارای دقت شناسایی بالا و شناسایی نادرست[11] پایین است.

کلمات کلیدی: Botnet، Botmaster، کانال دستور و کنترل، ناهنجاری[12]، اتصال[13]، خوشه‌بندی.

---

[1] Malware
[2] Compromised
[3] Command and Control Channel (C&C)
[4] Distributed Denial of Service
[5] Phishing
[6] Anomaly-based
[7] Signature
[8] Time Window
[9] Clustering
[10] Correlation
[11] False Positive
[12] Anomaly
[13] Netflow

فصل اول: مقدمه

در طی سال‌های گذشته، ما شاهد ظهور چشمگیر اینترنت و کاربردهای مبتنی بر آن بوده‌ایم به طوری که امروزه بخش جدایی‌ناپذیر زندگی ما شده است. در حالی که اینترنت فراهم کننده آسایش برای بشریت است، رشد وابستگی به آن باعث ظهور چالش‌های امنیتی بزرگی شده است. بدین طریق، امنیت اینترنت برای آن‌هایی که از اینترنت برای کار، تجارت یا آموزش استفاده می‌کنند، اهمیت زیادی دارد.

اکثر حملات و فعالیت‌های شیادانه در اینترنت توسط نرم‌افزارهای بدخواه[1] (بدافزارها) صورت می‌گیرد که شامل ویروس[2]، تروجان[3]، کرم[4]، جاسوس‌افزار[5] و اخیراً Botnetها می‌باشند. چنین بدافزارهایی اولین منبع بیشتر فعالیت‌های اسکن کردن[6] [۱]، حملات DDoS [۲] و فعالیت‌های شیادانه [۴,۳] در سرتاسر اینترنت می‌باشند. شکل‌های این بدافزارها همواره در حال تکامل یافتن هستند (از ویروس‌ها به Botnetها). در میان تمام شکل‌های بدافزارها، Botnetها، به طور اخص، به عنوان اولین انتخاب تبهکاران سایبری شناخته شده هستند تا با استفاده از آن جرایم اینترنتی از قبیل حملات DDoS، ارسال ایمیل‌های اسپم، مدیریت سایت‌های کلاهبرداری را ترتیب دهند.

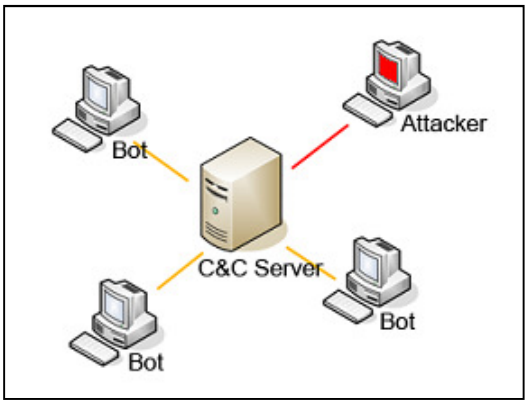
در این فصل، ابتدا مسئله Botnet را معرفی می‌کنیم و توضیح می‌دهیم چرا Botnet یک تهدید امنیتی جدی است. سپس چالش‌های موجود در در شناسایی Botnet را نشان می‌دهیم و اهدافی را که ما می‌خواهیم از ارائه روشمان به آن‌ها برسیم، معرفی می‌کنیم. در پایان، نوآوری‌ها و ساختار پایان‌نامه را ارائه می‌دهیم.

Botnet -۱-۱

ابتدا با معنی کلمات Bot و Botnet شروع می‌کنیم. Bot یک نرم‌افزار خودکار است یا به طور دقیق‌تر، بدافزاریست که به طور خودکار و مستقل روی یک کامپیوتر، بدون اجازه و آگاهی کاربر آن کامپیوتر اجرا می‌شود. کد Bot معمولاً توسط گروه‌های مجرم حرفه‌ای نوشته می‌شود و شامل مجموعه‌ای پر ارزش از قابلیت[7]ها است [۵] که برای انجام حملات و فعالیت‌های خرابکارنه استفاده می‌شوند. بعضی مواقع، از واژه Bot برای اشاره به کامپیوترهای آلوده به Bot استفاده می‌کنیم. Botnet شبکه‌ای از Botها است که تحت کنترل یک مهاجم (Botmaster یا Botherder) می‌باشد. اگر بخواهیم یک تعریف دقیق و رسمی از Botnet داشته باشیم، می‌توان گفت Botnet

[1] Malicious
[2] Virus
[3] Trojan Horse
[4] Worm
[5] Spyware
[6] Scanning
[7] Functionality

گروهی هماهنگ از Botها است که از طریق کانال دستور و کنترل تحت کنترل Botmaster می‌باشد. شکل ۱-۱ ساختار یک Botnet را نشان می‌دهد.



شکل ۱-۱) ساختار Botnet (برگرفته از [۶])

Bot معمولاً از ترکیبی از روش‌های پیشرفته بدافزارها استفاده می‌کند. به عنوان مثال، یک Bot می‌تواند از ثبت‌کننده کلید[1] (برای ضبط کردن کلیدهای فشار داده شده کیبورد مانند پسورد) و ابزارهای پایه‌ای[2] (برای پنهان نگهداشتن حضورش در سیستم) استفاده کند. علاوه بر آن، مانند نسل‌های قبلی بدافزارها مانند کرم‌ها، Bot می‌تواند خود را در اینترنت تکثیر کند تا اندازه Botnet را افزایش دهد (آلوده کردن کامپیوترهای آسیب‌پذیر[3] از راه دور یا روش‌های مهندسی اجتماعی[4] مانند ایمیل). اخیراً، Botmasterها با استفاده از سرورهای وب[5] آلوده سعی در آلوده کردن کاربرانی دارند که از این وب سایت‌ها دیدن می‌کنند [۷]. با استفاده از روش‌های متعدد انتشار، Botmaster می‌تواند تعداد زیادی Bot به خدمت بگیرد [۸]. امروزه Botnetها شامل ده‌ها تا صدها هزار Bot می‌باشند، اما بعضی از آن‌ها دارای میلیون‌ها Bot می‌باشند.

Botها به دلیل توانایی‌شان در برقراری کانال دستور و کنترل که از طریق آن می‌توانند توسط Botmaster به روز رسانی شده و مدیریت شوند، از شکل‌های قبلی بدافزار تمیز داده می‌شوند. Botها وقتی همگی تحت کنترل Botmaster قرار می‌گیرند، تشکیل Botnet می‌دهند. برای کنترل کردن Botnetها، Botmaster می‌تواند از چندین مکانیزم کنترل (ساختار و پروتکل) استفاده کند. پروتکل چت[6] یکی از نخستین و پر استفاده‌ترین پروتکل‌های کانال دستور و کنترل است.

---

[1] Keylogger
[2] Rootkit
[3] Vulanerable
[4] Social Engineering
[5] Web Server
[6] Internet Relay Chat (IRC)

پروتکل وب[1] به خاطر فیلتر نشدن ترافیک وب همچنان مورد استفاده است. اگرچه کنترل متمرکز[2] در گذشته بسیار موفقیت آمیز بوده است، اما Botmasterها برای مقابله با مشکل خرابی در یک قسمت[3]، به کنترل توزیع‌شده[4] روی آورده‌اند. به عنوان مثال، آن‌ها می‌توانند از ساختارهای نظیر به نظیر[5] برای سازماندهی و کنترل Botnetها استفاده کنند [۱۲,۱۱,۱۰,۹]. در بخش‌های بعدی توضیحات بیشتری در مورد مکانیزم‌های کانال دستور و کنترل Botnetها ارائه خواهیم داد.

برخلاف بدافزارهای پیشین مانند کرم‌ها، که برای سرگرمی استفاده می‌شدند، Botnetها برای بدست آوردن منفعت مالی استفاده می‌شوند. با توجه به اینکه حملات انجام شده توسط Botهای یک Botnet به طور همزمان و هماهنگ انجام می‌گیرد، حملات ترتیب داده شده توسط Botnetها دارای قدرت زیادی می‌باشند و به همین خاطر، امروزه Botnetها به عنوان بزرگترین تهدید برای امنیت اینترنت شناخته می‌شوند. امروزه، Botnetها ریشه بسیاری از حملات اینترنتی و فعالیت‌های غیرقانونی می‌باشند [۱۳,۳] که عبارتند از:

- حملات DDoS: Botmaster می‌تواند به Botها فرمان دهد تا به یک کامپیوتر داده ارسال کنند تا منابع (پهنای باند و پردازنده) کامپیوتر مورد نظر مصرف شده و نتواند سرویس‌های مورد نظر را به کاربران خود ارائه دهد. امروزه، اکثر حملات DDoS توسط Botnetها صورت می‌گیرند. اگرچه حملات DDoS دارای روش‌های ساده‌ای هستند، اما حملات DDoS به خاطر تعداد زیاد Botها و در نتیجه حجم زیاد داده ارسالی، بسیار کارآ هستند و مقابله با این نوع حملات خیلی دشوار می‌باشد. حملات DDoSای که روی وب سایت‌های کشور استونیان در سال ۲۰۰۷ صورت گرفت، نمونه شناخته شده این حملات هستند.
- ارسال ایمیل‌های اسپم: بیشتر از ۹۵٪ ایمیل‌های ارسالی در اینترنت، اسپم هستند. بیشتر این ایمیل‌های اسپم توسط Botnetها ارسال می‌شوند [۱۵,۱۴]. تعدادی از Botnetهای شناخته شده عمدتاً برای ارسال اسپم استفاده می‌شوند که به عنوان نمونه می‌توان از Bobax (مبتنی بر وب) و StormWorm یا Peacomm [۱۰,۹] (مبتنی بر نظیر به نظیر) نام برد.
- سرقت اطلاعات: Botها به صورت گسترده برای سرقت اطلاعات حساس از قبیل شماره کارت‌های اعتباری[6]،

---

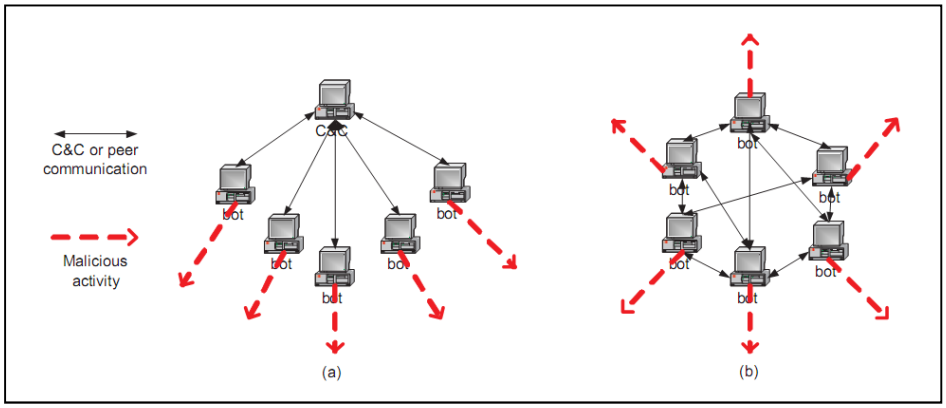
[1] Hyper Text Transfer Protocol (HTTP)
[2] Centralized
[3] Single Point of Failure
[4] Distributed
[5] Peer-to-Peer (P2P)
[6] Credit Card

پسورد و ... استفاده می‌شوند. با استفاده از روش‌های ثبت کلید، یک Bot به آسانی می‌تواند پسورد کاربر یک بانک آنلاین را بدزدد. این قضیه تبدیل به یک مشکل خیلی جدی شده است.

- سایت‌های کلاهبرداری: Botnetها به طور گسترده به عنوان میزبان[1] سایت‌های کلاهبرداری استفاده می‌شوند. تبهکاران معمولاً با ارسال ایمیل‌های اسپم (با استفاده از Botnetها) کاربران را فریب می‌دهند تا از سایت‌های کلاهبرداری (نمونه قلابی سایت‌های واقعی) بازدید کنند و بدین طریق اطلاعات حساس کاربران از قبیل نام کاربری، پسورد و شماره کارت اعتباری آن‌ها را به سرقت می‌برند [۱۶].
- تکثیر دیگر بدافزارها: Botnetها اصولاً سکو[2]ی خوبی برای توزیع شکل‌های دیگر بدافزار هستند. Botnetها می‌توانند شکل‌های دیگری از حملات و فعالیت‌های شیادانه را نیز انجام دهند.

۱-۲- ساختارها و پروتکل‌های دستور و کنترل

با توجه به تعریفی که در بخش قبل از Botnet داشتیم، Botnet با توجه به ارتباطات دستور و کنترل و فعالیت‌های خرابکارانه آن‌ها شناخته می‌شوند. شکل ۱-۲ دو ساختار معمول Botnetها را نشان می‌دهد: متمرکز و نظیر به نظیر.



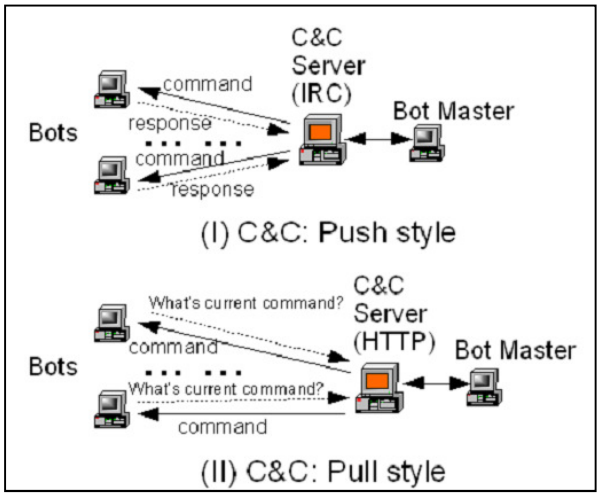
شکل ۱-۲) ساختارهای Botnet: (a) متمرکز (b) نظیر به نظیر (برگرفته از [۱۷])

۱-۲-۱- ساختار متمرکز

به طور کلی می‌توان ساختارهای متمرکز Botnet را بر اساس نحوه ارسال دستورات به Botها توسط Botmaster، به دو دسته تقسیم کرد: فشاری[3] و کششی[4]. شکل ۱-۳ نمایی از این ساختار را نشان می‌دهد.

---

[1] Host
[2] Platform
[3] Push
[4] Pull



شکل ۱-۳) پروتکل‌های متمرکز Botnet (برگرفته از [۱۸])

در ساختار فشاری، Botها به سرور دستور و کنترل (مانند چت) متصل بوده و منتظر دریافت دستور از Botmaster هستند. Botmaster دستورات را در کانال قرار می‌دهد و تمام Botهای متصل به کانال دستورات را به صورت بلادرنگ[1] دریافت می‌کنند. در ساختار فشاری Botmaster کنترلی بلادرنگ روی Botnet دارد. یکی از نمونه‌های بارز و معروف ساختار فشاری، پروتکل چت است. اکثر Botnetهای موجود مانند SpyBot، SdBot و PhatBot از این پروتکل استفاده می‌کنند.

در ساختار کششی، Botmaster فایل دستورات را در سرور دستور و کنترل (مانند وب) قرار می‌دهد. Botها به طور پی در پی به سرور دستور و کنترل وصل می‌شوند و فایل دستورات را دانلود می‌کنند. بنابراین Botmaster کنترلی بلادرنگ روی Botnet ندارد چرا که یک تأخیر بین زمانی که Botmaster فایل دستورات را در سرور دستور و کنترل قرار می‌دهد و زمانی که این فایل توسط Botها دانلود می‌شود وجود دارد. Botnetهایی مانند Bobax از این ساختار استفاده می‌کنند.

۱-۲-۲- ساختار نظیر به نظیر

در حال حاضر، ساختارهای نظیر به نظیر در سیستم‌های به اشتراک‌گذاری فایل[2] استفاده می‌شوند. StormWorm [۱۰] یکی از Botnetهای معروفی است که از ساختار نظیر به نظیر برای دستور و کنترل استفاده می‌کند. در این Botnet هر Bot به طور پی در پی با کامپیوترهای دیگر تماس برقرار می‌کند تا بتواند فایل‌های مربوط به دستورات را پیدا کند.

---
[1] Real-time
[2] File Sharing

۱-۳- شناسایی Botnet: چالش‌ها و اهداف

برای مقابله با حملات Botnetها، بزرگترین تهدید امنیتی حال حاضر، ابتدا باید Botnetها (Botهـا و سـرورهای دستور و کنترل) را در شبکه در حال مانیتور شناسایی کرد تا بتوان به طور مؤثر با آن‌ها مقابله کرد. شناسایی Botnet دارای چالش‌های بسیاری است که عبارتند از:

- Botها به صورت مخفیانه روی کامپیوترهای آلوده در حال اجرا هستند. از آنجایی که آن‌ها برای اهداف طولانی مدت طراحی شده‌اند تا منفعت رسانی کنند، Botها معمولاً از منابع سیستم مانند پردازنـده، حافظـه و پهنـای باند خیلی کم استفاده می‌کنند و سعی می‌کنند آسیبی به کامپیوتر نرسانند تا توجه کاربر به آن‌ها جلب نشـود. آن‌ها آنتی‌ویروس‌های روی کامپیوتر را غیر فعال می‌کنند و از ابزارهـای ریشـه‌ای بـرای جلـوگیری از شـناخته شدن استفاده می‌کنند. بنابراین، راهکارهای مبتنی بر میزبان[1] [۲۰,۱۹] خیلی کارا نیستند. در این پایـان‌نامـه، روی روش‌های مبتنی بر شبکه[2] تمرکز داریم.
- آلوده شدن کامپیوترها به Botها معمولاً یک فرایند چنـد مرحلـه‌ای اسـت و بررسـی یـک جنبـه خـاص کـارا نمی‌باشد و منجر به عدم شناسایی نادرست[3] و شناسایی نادرست بالا می‌شـود. در مقابـل، بررسـی چنـد جنبـه مختلف به طور همزمان کارایی بیشتری داشته و اطلاعات بیشتری از نحوه آلودگی در اختیار می‌گذارد.
- Botها به طور پویا در حال تکامل هستند. به عنوان مثال، آن‌ها می‌توانند پی در پی فایل باینری خود را به روز رسانی کنند، حتی سریعتر از به روز رسانی شدن پایگاه امضاءی آنتی‌ویروس‌ها. بنـابراین، روش‌هـای مبتنـی بـر امضاء[4] نمی‌توانند خیلی کارا باشند.
- Botnetها می‌توانند کانال‌های دستور و کنترل متفاوتی داشته باشند. آن‌ها می‌توانند از پروتکل‌هـای مختلفـی مانند چت و وب استفاده کنند. آن‌ها می‌توانند محتوای ارتباطات دسـتور و کنتـرل را رمـز کننـد. آن‌هـا حتـی می‌توانند از ساختارهای مختلفی مانند روش‌های نظیر به نظیر برای سازماندهی و کنترل Botها استفاده کنند. بنابراین روش‌هایی که وابسته به یک ساختار یا پروتکل دستور و کنترل باشند، خیلی مطلوب نیستند.

به خاطر این چالش‌ها، روش‌های موجود مانند آنتی‌ویروس‌های معمولی نمی‌توانند به طور کامـل مشـکل شناسـایی Botnet را حل کنند. در فصل دوم، خلاصه‌ای از روش‌های مرتبط با شناسـایی Botnet را ارائـه مـی‌دهـیم و توضـیح

[1] Host-based
[2] Network-based
[3] False Negative
[4] Signature-based

می‌دهیم چرا این روش‌ها برای شناسایی Botnet کافی نیستند.

در این پایان‌نامه، یک روش مبتنی بر شبکه برای شناسایی Botnet پیشنهاد می‌کنیم. در طراحی روشمان، اهداف زیر در نظر گرفته شده‌اند:

۱. روشمان باید رفتارهای بنیادی و تغییر ناپذیر Botnetها را شناسایی کند.
۲. روشمان باید فراگیر باشد. بدین معنی که روش نباید مختص یک نوع Botnet خاص باشد یا وابسته به یک ساختار یا پروتکل خاص دستور و کنترل باشد.
۳. روشمان باید یک سیستم شناسایی عملی فراهم کند که بتواند در شبکه‌های واقعی کار کند. بدین معنی که سیستم شناسایی بتواند به طور کامل Botnetهای واقعی را شناسایی کند و دارای شناسایی نادرست پایین و مصرف منابع معقولی باشد.

۱-۴- نوآوری‌ها و ساختار پایان‌نامه

این پایان‌نامه دارای نوآوری‌های زیر می‌باشد:

۱. ما یک روش کاملاً مبتنی بر ناهنجاری در رفتار ترافیک شبکه برای شناسایی Botnet ارائه داده‌ایم. این روش هیچ نوع فرضی در مورد امضاءی فایل‌های باینری Botها، امضاءی ارتباطات Botها با سرورهای دستور و کنترل و ساختار و پروتکل کانال دستور و کنترل Botnetها نمی‌کند. همچنین این روش قادر است به صورت بلادرنگ Botهای موجود در شبکه را شناسایی کند.
۲. روش ما Botnetهایی را که فعالیت‌های خرابکارانه انجام نمی‌دهند یا فعالیت‌های خرابکارانه آن‌ها قابل شناسایی نیست را نیز می‌تواند شناسایی کند. در عین حال اگر Botnetها فعالیت خرابکارانه انجام دهند، روش ما می‌تواند با سرعت بیشتری آن‌ها را شناسایی کند.
۳. ما یک سیستم واقعی و عملی (BotGrabber) بر اساس روشمان پیاده‌سازی کرده‌ایم. این سیستم با استفاده از ترافیک واقعی شبکه ارزیابی شده است و نتایج نشان می‌دهد سیستم دارای شناسایی نادرست و عدم شناسایی نادرست پایینی است.

ساختار پایان‌نامه بدین ترتیب است: در فصل دوم کارهای مرتبط با شناسایی Botnet را معرفی می‌کنیم و توضیح می‌دهیم چرا روش‌های موجود نمی‌توانند مسئله شناسایی Botnet را به طور کامل حل کنند. همچنین خصوصیات

روش‌های شناسایی Botnet را بررسی می‌کنیم. در فصل سوم سیستم BotGrabber را معرفی می‌کنیم و خصوصیات، طراحی، پیاده‌سازی و ارزیابی این سیستم را ارائه می‌دهیم. در پایان، پایان‌نامه را نتیجه‌گیری می‌کنیم و کارهای آینده را شرح می‌دهیم.

فصل دوم: کارهای مرتبط

شناسایی Botnet نسبتاً حوزه جدیدی است. اخیراً چندین روش برای شناسایی Botnet معرفی شده‌اند. برای اینکه یک دانش سیستماتیک از روابط و تفاوت‌های سیستم‌های شناسایی Botnet فراهم کنیم، ابتدا خصوصیات سیستم‌های شناسایی Botnet را از هفت جنبه بررسی می‌کنیم و سپس چند نمونه از روش‌های شناسایی Botnet را تشریح می‌کنیم.

۲-۱- خصوصیات روش‌های شناسایی Botnet

در این بخش خصوصیات سیستم‌های شناسایی Botnet را از هفت جنبه بررسی می‌کنیم. جدول ۲-۱ خصوصیات روش‌های شناسایی Botnet را نشان می‌دهد.

جدول ۲-۱) خصوصیات روش‌های شناسایی Botnet

مبتنی بر میزبان یا مبتنی بر شبکه	در روش‌های مبتنی بر میزبان، سیستم روی کامپیوتر مورد نظر نصب می‌شود و علاوه بر اطلاعات مربوط به ترافیک شبکه، اطلاعات دیگری از قبیل نحوه فراخوانی توابع سیستمی[1] و ثبت‌های[2] سیستم عامل را بررسی می‌کند. از مزایای این روش این است که به تمام اطلاعات یک میزبان دسترسی دارد و دقت آن بیشتر است. از کاستی‌های این روش این است که فقط به اطلاعات یک میزبان دسترسی دارد و قادر به شناسایی حملاتی که روی شبکه صورت می‌گیرد، نیست. در روش‌های مبتنی بر شبکه، سیستم در درگاه[3] شبکه قرار می‌گیرد و به ترافیک تمام کامپیوترهای موجود در شبکه دسترسی دارد. از مزایای این روش این است که به ترافیک تمام کامپیوترهای موجود در شبکه دسترسی دارد و قادر به شناسایی حملات روی شبکه است. از کاستی‌های این روش این است که به اتفاقاتی در میزبان‌های شبکه صورت می‌گیرد به اطلاع است و همین باعث کاهش دقت آن می‌شود.
مبتنی بر امضاء یا مبتنی بر ناهنجاری	روش‌های مبتنی بر امضاء بر اساس امضاءهایی که از حملات شناخته شده جمع‌آوری شده‌اند کار می کند. از مزایای این روش این هست که خطای پایینی دارد و خیلی سریع کار می‌کند. از کاستی‌های این روش این است که قادر به شناسایی حملاتی که

[1] System Call
[2] Log
[3] Gateway

	قبلاً شناسایی نشده‌اند، نیست. روش‌های مبتنی بر ناهنجاری با استفاده از مدلی نرمال، سعی در شناسایی رفتارهای ناهنجار بر خلاف مدل نرمال دارند. از مزایای این روش این است که قادر به شناسایی حملات ناشناخته هستند. از کاستی‌های این روش سرعت پایین و خطای بالا هستند.
منفعل[1] یا فعال[2]	روش‌های منفعل ترافیک شبکه را مانیتور می‌کنند و در ارتباطات و فعالیت‌های Botnet مداخله نمی‌کنند. از مزایای این روش عدم جلب توجه Botmasterها است. از کاستی‌های این روش کندی و طولانی بودن زمان مانیتورینگ است. روش‌های فعال در ارتباطات Botnet شرکت می‌کنند و ترافیک جدید در ارتباطات قرار می‌دهند و یا ترافیک آن را تغییر می‌دهند. از مزایای این روش سرعت است. از کاستی‌های این روش جلب توجه کردن Botmasterها و بالطبع تغییر رفتار Botnet است.
فاز شناسایی: **آماده‌سازی[3] یا عملیاتی[4]**	چرخه زندگی یک Bot به طور کلی دارای دو فاز است: آماده‌سازی و عملیاتی. در فاز آماده‌سازی یک کامپیوتر تمیز بر اثر سوء استفاده از آسیب‌پذیری از راه دور یا اجرای فایل‌های اجرایی خرابکار (مانند پیوست ایمیل)، تبدیل به یک Bot می‌شود. وقتی که Bot به کانال دستور و کنترل وصل شد، فاز عملیاتی شروع می‌شود، جایی که Bot به طور مستقیم تحت فرمان Botmaster قرار می‌گیرد تا هر فعالیتی را انجام دهد.
شناسایی یک Bot مجزا یا شناسایی گروهی از Botها	اگر هدف شناسایی گروهی از Botها باشد، حداقل دو Bot باید در شبکه موجود باشند. از کاستی‌های این رویکرد این است که حداقل باید دو Bot در شبکه موجود باشند. اگر هدف شناسایی یک Bot باشد، آنگاه با وجود یک Bot در شبکه نیز می‌توان آن را شناسایی کرد. از مزایای این رویکرد این است که با وجود فقط یک Bot در شبکه

[1] Passive
[2] Active
[3] Preparation
[4] Operation

	باز هم می‌توان آن را پیدا کرد.
میزان وابستگی به ساختارها و پروتکل‌های کانال‌های دستور و کنترل	بعضی از روش‌ها به ساختارها و پروتکل‌های کانال‌های دستور و کنترل وابسته هستند. در حال حاضر پروتکل‌های IRC، HTTP و P2P پر کاربردترین پروتکل‌هایی هستند که در Botnetهای امروزی استفاده می‌شوند. از مزایای روش‌هایی که وابستگی‌ای به ساختارها و پروتکل‌های دستور و کنترل ندارند این است که قادر به شناسایی انواع Botnetها هستند. ولی در عین حال این روش‌ها دارای سرعت پایینی بوده و کارایی پایین‌تری دارند.
داده مورد نیاز: اتصال، سربار[1] بسته یا کل بسته	روش‌های شناسایی Botnet از اطلاعات مختلفی در لایه‌های شبکه برای شناسایی Botnet استفاده می‌کنند. هر چه داده مورد نیاز بیشتر باشد، کارایی سیستم کاهش می‌یابد ولی دقت آن افزایش می‌یابد.

۲-۲-روش‌های شناسایی Botnet

روش اول [۲۱]: Botnet Detection and Response: The Network is the Infection

خصوصیات: مبتنی بر شبکه، مبتنی بر ناهنجاری، منفعل، فاز شناسایی: عملیاتی، شناسایی گروهی از Botها، وابسته به ساختارهای متمرکز دستور و کنترل، داده مورد نیاز: کل بسته.

شرح: این روش با استفاده از مسأله گردآوری و ساماندهی[2] Botnet سعی در شناسایی دامنه‌های مربوط به Botnet دارد. او با شمردن تعداد درخواست‌های DDNS، دامنه‌هایی را که ناهنجاری داشته باشند را پیدا می‌کند. رابطه (۲-۱) و رابطه (۲-۲) نحوه محاسبه درخواست‌ها و تعیین دامنه‌های غیر عادی را نشان می‌دهند.

$$C_{SLD_i}=R_{SLD_i}+\sum_{j=1}^{|SLD_i|} R_{3LD_j} \qquad \text{(۲-۱) (برگرفته از [۲۱])}$$

$$P(|X-\mu|\geq t)\leq\frac{\sigma^2}{t} \qquad \text{(۲-۲) (برگرفته از [۲۱])}$$

در رابطه (۲-۱) R_{SLD_i} تعداد درخواست‌هایی است که به سطح اول دامنه i ام صورت گرفته و R_{3LD_j} تعداد

[1] Header
[2] Rallying

درخواست‌هایی است که به سطح دوم دامنه i ام صورت گرفته است. در بعضی موارد استفاده از رابطه (۲-۲) جوابگو نیست. به همین خاطر، برای هر روز یک بردار با ۲۴ مؤلفه می‌سازیم که هر مؤلفه تعداد درخواست‌ها می‌باشد و این مؤلفه‌ها به صورت نزولی مرتب می‌شوند و از رابطه (۲-۳) برای تعیین بردارهای ناهنجار استفاده می‌کنیم.

$$d(x,\bar{y})=\sum_{i=0}^{n-1}\left(\frac{|x_i-\bar{y}_i|}{\bar{\sigma}_i}\right) \qquad \text{(۲-۳) (برگرفته از [۲۱])}$$

روش دوم [۲۲]: An Algorithm for Anomaly-based Botnet Detection

خصوصیات: مبتنی بر شبکه، مبتنی بر امضاء و ناهنجاری، منفعل، فاز شناسایی: عملیاتی، شناسایی گروهی از Botها، وابسته به ساختارهای متمرکز دستور و کنترل، داده مورد نیاز: کل بسته.

شرح: این روش برای شناسایی Botnetهای مبتنی بر چت، اطلاعات و آمار مربوط به کانال‌های چت را با ناهنجاری‌های مربوط به فعالیت‌های اسکن کردن در پروتکل TCP ترکیب کرده است. این روش ابتدا بسته‌های TCP را جمع‌آوری می‌کند و کانال‌های چت را شناسایی می‌کند. به طور همزمان IPهایی که اسکن انجام می‌دهند را نیز پیدا می‌کند. سپس از ترکیب این دو لیست به کامپیوترهای چت مشکوک و در ادامه به دستور و کنترل پی می‌برد. هر کامپیوتری که TcpWorkWeight بالایی داشته باشد، با احتمال زیادی یک اسکنر است. اما اگر یک کامپیوتر در یک کانال دارای وزن بالایی بود، لزوماً آن کانال مشکل دار نیست بلکه اگر تعداد قابل توجهی از کامپیوترهای چت در یک کانال دارای وزن بالایی باشند، می‌توان گفت آن کانال مربوط به Botnet است. رابطه (۲-۴) نحوه محاسبه TcpWorkWeight را نشان می‌دهد.

$$TcpWorkWeight = (S_s+F_s+R_r)/T_{sr} \qquad \text{(۲-۴) (برگرفته از [۲۲])}$$

روش سوم [۲۳]: Detecting Botnets by Analyzing DNS Traffic

خصوصیات: مبتنی بر شبکه، مبتنی بر ناهنجاری، منفعل، فاز شناسایی: عملیاتی، شناسایی گروهی از Botها، وابسته به ساختارهای متمرکز دستور و کنترل، داده مورد نیاز: کل بسته.

شرح: این روش برای شناسایی دامنه‌های مشکوک از موارد زیر استفاده می‌کند:

- سرورهای DNS غیر محلی: در مواردی که کامپیوترهای داخل شبکه به جای استفاده از سرورهای DNS

محلی که توسط ISP مشخص شده‌اند، از سرورهای DNS متعدد غیر محلی استفاده می‌کنند.

- پرسش‌های با نوع غیر معمول: در مواردی که کامپیوترها تعداد زیادی از نوع‌های پرسش MX و AXFR/IXFR را استفاده می‌کنند.

با استفاده از موارد فوق، دامنه‌های مشکوک را شناسایی کرده و با استفاده از روش‌های یادگیری ماشین، دامنه‌های مربوط به Botnet را تشخیص می‌دهد.

روش چهارم [۲۴]: Detection and mitigation of fast-flux service networks

خصوصیات: مبتنی بر شبکه، مبتنی بر ناهنجاری، منفعل، فاز شناسایی: عملیاتی، شناسایی گروهی از Botها، وابسته به ساختارهای متمرکز دستور و کنترل، داده مورد نیاز: کل بسته.

شرح: این روش تکنیکی به نام Fast-Flux [۲۶,۲۵] را معرفی می‌کند که معمولاً از سوی Botnetها برای جلوگیری از شناسایی صورت می‌گیرد. در این تکنیک در هر دفعه پرس و جوی دامنه Botnet، IPهای سرورها تعویض می‌شوند. این عمل موجب می‌شود سرور دستور و کنترل توزیع شود و نتوان با غیر فعال کردن آن کل Botnet را غیر فعال کرد. ولی همین تغییرات پی در پی هم می‌تواند عاملی برای شناسایی دامنه‌های مشکوک از دامنه‌های عادی باشد. شبکه‌های Fast-Flux از دو محدودیت رنج می‌برند: پراکندگی کامپیوترها[1] و عدم کنترل فیزیکی روی کامپیوترها. از آنجایی که Botها در سراسر دنیا پخش هستند، تعداد ASNهایی که مسئول این IPها هستند به مرور زمان افزایش می یابد. از طرفی چون Botmaster روی کامپیوترها کنترل فیزیکی ندارد، برای اطمینان از فعال بودن یکی از کامپیوترها، تعداد رکوردهای A و NS برگشتی را زیاد می‌گذارد. برای شناسایی یک دامنه Fast-Flux از پارامتر Fluxiness استفاده می‌شود که هر چه نرخ رکوردهای A بازگشتی بیشتر باشد، دامنه با احتمال بیشتری Fast-Flux است. رابطه (۲-۵) نحوه محاسبه پارامتر Fluxiness را نشان می‌دهد.

$$\varphi = n_A / n_{single} \quad \text{(۲-۵) (بر گرفته از [۲۴])}$$

همچنین برای شناسایی دامنه‌های Fast-Flux می‌توان از سه پارامتر تعداد رکوردهای A، رکوردهای NS و تعداد ASN استفاده کرد. رابطه‌های (۲-۶) و (۲-۷) نحوه پیدا کردن دامنه‌های Fast-Flux را نشان می‌دهند.

$$f(x)=\omega^T x=\omega_1 \times n_A+\omega_2 \times n_{ASN}+\omega_3 \times n_{NS} \quad \text{(۲-۶) (بر گرفته از [۲۴])}$$

[1] IP Address Diversity

$$F(x)=\begin{cases}\omega^T x-b>0, & \text{if x is a fast-flux domain}\\ \omega^T x-b\leq 0, & \text{if x is a benign domain}\end{cases}$$ (۲-۷) (بر گرفته از [۲۴])

روش پنجم [۲۷]: FluXOR: detecting and monitoring fast-flux service networks

خصوصیات: مبتنی بر شبکه، مبتنی بر ناهنجاری، منفعل، فاز شناسایی: عملیاتی، شناسایی گروهی از Botها، وابسته به ساختارهای متمرکز دستور و کنترل، داده مورد نیاز: کل بسته.

شرح: این روش ابتدا به صورت دستی دامنه‌های خوب و بد را تعیین و یک مدل کلاس‌بندی تهیه می‌کند. سپس با استفاده از آن مدل، دامنه‌های جدید را کلاس‌بندی می‌کند. جدول ۲-۲ خصوصیات مورد نظر این دامنه‌ها را نشان می‌دهد. جدول ۲-۳ و جدول ۲-۴ خصوصیات ذکر شده را برای چند دامنه نمونه نشان می‌دهند.

جدول ۲-۲) خصوصیات دامنه‌ها (برگرفته از [۲۷])

Category	#	*Description*
Domain name	F_1	Domain age
	F_2	Domain registrar
Availability of the network	F_3	Number of distinct DNS records of type “A”
	F_4	Time-to-live of DNS resource records
Heterogeneity of the agents	F_5	Number of distinct networks
	F_6	Number of distinct autonomous systems
	F_7	Number of distinct resolved qualified domain names
	F_8	Number of distinct assigned network names
	F_9	Number of distinct organisations

جدول ۲-۳) نمونه‌ای از دامنه‌های سالم و ناسالم (برگرفته از [۲۷])

	FQDN	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
Benign	www.avast.com	539	NetworkSolutions	12	3600	5	3	1	5	2
	adriaticobishkek.com	65	Melbourne IT	21	1200	1	1	1	1	1
	google.com	542	MarkMonitor	3	300	2	1	1	1	1
	Mean	**493.27**	**N/A**	**2.86**	**4592.53**	**1.27**	**1.11**	**1.08**	**1.21**	**1.07**
	Standard dev.	**289.27**	**N/A**	**3.89**	**7668.74**	**0.65**	**0.36**	**0.74**	**0.58**	**0.25**
Malicious	www.eveningher.com	18	PayCenter	127	300	83	49	33	71	54
	www.factvillage.com	2	PayCenter	117	300	81	46	34	67	54
	www.doacasino.com	2	NameCheap	33	180	19	14	11	19	14
	Mean	**4.85**	**N/A**	**98.13**	**261.49**	**63.75**	**38.36**	**27.98**	**53.58**	**41.47**
	Standard dev.	**4.9**	**N/A**	**37.27**	**59.64**	**23.91**	**12.34**	**8.5**	**18.73**	**15.41**

جدول ۲-۴) مقایسه خصوصیات مربوط به کامپیوتر (برگرفته از [۲۷])

IP address	F_5	F_6	F_7	F_8	F_9
15.216.110.140	**15.0.0.0/8**	**AS9218**	**polyserve.com**	**HP-INTERNET**	**Hewlett-Packard**
15.192.45.22	**15.0.0.0/8**	**AS9218**	**polyserve.com**	**HP-INTERNET**	**Hewlett-Packard**
15.200.30.24	**15.0.0.0/8**	**AS9218**	**polyserve.com**	**HP-INTERNET**	**Hewlett-Packard**

(a) `hp.com` (benign)

IP address	F_5	F_6	F_7	F_8	F_9
67.228.112.196	67.228.0.0/16	**AS36351**	**avast.com**	SOFTLAYER-4-5	**SoftLayer Tech.**
216.12.205.130	216.12.192.0/19	AS36420	**avast.com**	EVRY-BLK-4	Everyone Internet
74.86.245.119	74.86.0.0/16	**AS36351**	**avast.com**	SOFTLAYER-4-4	**SoftLayer Tech.**

(b) `www.avast.com` (benign)

IP address	F_5	F_6	F_7	F_8	F_9
61.18.66.?	61.18.0.0/16	AS9908	hkcable.com.hk	HKCABLE-HK	HK Cable TV
218.47.195.?	218.47.0.0/16	AS4713	ap.plala.or.jp	PLALA	Plala Net. Inc.
81.173.151.?	81.173.151.0/24	AS8422	netcologne.de	NC-DIAL-IN-POOL	NetCologne

(c) `www.factvillage.com` (malicious)

روش ششم [۲۸]: Botnet Detection by Monitoring Group Activities in DNS Traffic

خصوصیات: مبتنی بر شبکه، مبتنی بر ناهنجاری، منفعل، فاز شناسایی: عملیاتی، شناسایی گروهی از Botها، وابسته به ساختارهای متمرکز دستور و کنترل، داده مورد نیاز: کل بسته.

شرح: این روش با استفاده از تعداد پرس و جوها توسط Botها سعی در شناسایی دامنه‌های ناسالم دارد. این مقاله همچنین نشان می‌دهد که Botmasterها، سرورهای دستور و کنترل را هر چند وقت یکبار تغییر می‌دهند[1]. زمان‌هایی که Botها DNS را مورد پرس و جو قرار می‌دهند بدین ترتیب است: هنگام گردآوری و ساماندهی Botها بعد از آلوده کردن سیستم، هنگام خرابکاری مانند حملات DDoS و فرستادن اسپم، بعد از قطع ارتباط با سرور دستور و کنترل (Bot مدتی بعد از قطع ارتباط، اقدام به پرس و جو می‌کند)، هنگام مهاجرت سرور دستور و کنترل که دامنه سرور دستور و کنترل تغییر می‌کند و هنگام تغییر IP سرور دستور و کنترل [۲۹].

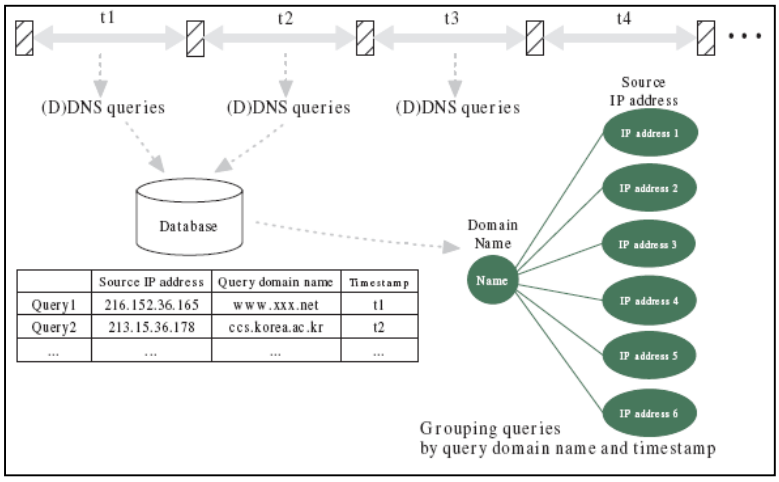
در این روش برای اضافه کردن پرس و جوها به پایگاه‌داده، ابتدا بازه‌های زمانی مشخصی را در نظر می‌گیریم و سپس پرس و جوهای در آن بازه زمانی را بر حسب دامنه گروه‌بندی می‌کنیم. به هر دامنه یک لیست اختصاص می‌دهیم که در بردارنده IPهایی است که این دامنه را مورد پرس و جو قرار می‌دهند. اگر یک دامنه‌ای در لیست سفید (دامنه‌های سالم) بود یا تعداد IPهایی که آن را مورد پرس و جو قرار داده‌اند کمتر از یک مقدار خاص بود، آن دامنه از

[1] C&C Server Migration

پایگاه‌داده حذف می‌شود. شکل ۲-۱ نحوه اضافه کردن پرس و جوها به پایگاه‌داده را نشان می‌دهد. جدول ۲-۵ پرس و جوهای دامنه‌های سالم و ناسالم را نشان می‌دهد.

جدول ۲-۵) تفاوت پرس و جوهای دامنه‌های سالم و ناسالم (برگرفته از [۲۸])

	Source IPs accessed to domain name	Activity and Appearance Patterns	DNS Type
Botnet DNS	Fixed size Group (Botnet members)	Group activity Intermittently appeared (Specific situation)	Usually DDNS
Legitimate DNS	Anonymous (Legitimate users)	Non-group activity Randomly and continuously appered (Usually)	Usually DNS



شکل ۲-۱) نحوه اضافه کردن پرس و جوها به پایگاه‌داده (برگرفته از [۲۸])

بعد از اضافه کردن پرس و جوها به پایگاه‌داده، نوبت به شناسایی دامنه‌های ناسالم می‌رسد. بدین ترتیب که اگر دامنه‌ای در بازه‌های زمانی مختلف مورد پرس و جو قرار گرفته باشد، می‌توان میزان شباهت بین بازه‌های مختلف را از رابطه (۲-۸) بدست آورد.

$$S=\frac{1}{2}\times\left(\frac{C}{A}+\frac{C}{B}\right) \qquad (A\neq 0,B\neq 0)$$

(۲-۸) (برگرفته از [۲۸])

در رابطه (۲-۸)، A و B تعداد IPهای موجود در لیست‌های دو بازه زمانی مورد نظر و C تعداد IPهای مشترک دو لیست مورد نظر است. اگر دامنه مورد نظر فقط در یک بازه زمانی وجود داشت، صبر می‌کنیم تا در بازه‌های زمانی دیگر پیدا شود. اگر S به صفر نزدیک بود، دامنه مورد نظر را به لیست دامنه‌های سالم اضافه و از پایگاه‌داده حذف می‌کنیم.

برای شناسایی دامنه‌های ناسالم در حال مهاجرت، از الگوریتم قبلی استفاده می‌کنیم ولی با این تفاوت که این شباهت بین دو دامنه صورت می‌گیرد و اندازه لیست IP این دو دامنه باید نزدیک به هم باشد. از کاستی‌های این روش این است که مهاجمین می‌توانند با تولید بسته‌های تقلبی، این روش را دچار مشکل کنند. در مقابل می‌توان با بررسی ترافیک‌های دیگر غیر از DNS بسته‌های تقلبی را تشخیص داد.

روش هفتم [۴]: Revealing botnet membership using DNSBL counter-intelligence

خصوصیات: مبتنی بر شبکه، مبتنی بر ناهنجاری، منفعل، فاز شناسایی: عملیاتی، شناسایی گروهی از Botها، عدم وابستگی به ساختارها و پروتکل‌های دستور و کنترل، داده مورد نیاز: کل بسته.

شرح: این روش از لیست سیاه[1] DNS (DNSBL) برای شناسایی Botnetهایی که اسپم تولید می‌کنند، استفاده می‌کند. اصولاً میل سرورها برای چک کردن اسپم، آدرس فرستنده را در DNSBL چک می‌کنند. اساس فرض این روش این است که Botmaster برای اطلاع یافتن از وضعیت Botهای خود، DNSBL را مورد جستجو قرار می‌دهند و در نتیجه کامپیوتری که وضعیت دیگران را مورد پرس و جو قرار می‌دهد و خود خیلی کم مورد جستجو قرار می‌گیرد، مشکوک است. در این روش ابتدا گرافی از پرس و جوها تولید می‌شود به طوری که یال از گره A به گره B نشان‌دهنده این است که گره A از DNSBL در مورد گره B پرس و جو انجام داده است. این پرس و جوها دو خاصیت مهم دارند: خاصیت مکانی و خاصیت زمانی. خاصیت مکانی بدین معنی است که یک میل سرور قانونی هم دیگران را مورد پرس و جو قرار می‌دهد و هم توسط میل سرورهای دیگر مورد پرس و جو قرار می‌گیرد. در نتیجه، پرس و جوگرهای غیر قانونی دیگران را مورد پرس و جو قرار می‌دهند ولی خود مورد پرس و جو قرار نمی‌گیرند. ولی در بعضی از شرکت‌ها، میل سرورهای درون‌سو[2] و برون‌سو[3] با هم فرق می‌کنند و در نتیجه با این خاصیت میل سرورهای درون سو هم مورد سوء ظن قرار می‌گیرند. پس کامپیوترهایی که دارای درجه خروجی بالا و درجه ورودی پایین باشند، به احتمال زیاد غیر قانونی هستند. در مواقعی که Botهای پرس و جوگر از Botهای فرستنده اسپم جدا باشند، این خاصیت مؤثر است. رابطه (۲-۹) نحوه محاسبه احتمال Bot بودن یک کامپیوتر را نشان می‌دهد.

[1] Blacklist
[2] In-bound
[3] Out-bound

$$\lambda_n = \frac{d_{n,out}}{d_{n,in}}$$ (۲-۹) (برگرفته از [۴])

خاصیت زمانی بدین معنی است که الگوی زمانی ورود درخواست‌ها به DNSBLها برای میل سرورهای قانونی و کامپیوترهای غیرقانونی متفاوت است. چرا که الگوی پرس و جوی میل سرورهای قانونی تابعی از الگوی ورود ایمیل‌های واقعی به آن میل سرور است (در حقیقت نرخ ورود درخواست‌ها نشان‌دهنده نرخ ورود ایمیل‌ها به میل سرور است) ولی الگوی پرس و جوی کامپیوترهای غیر قانونی تابع هیچ قانونی نیست. همچنین ورود ایمیل‌های عادی به میل سرورها در بازه زمانی خاصی صورت می‌گیرد.

در حال حاضر سه نوع روش برای پرس و جوی Botnet از DNSBL وجود دارد: شناسایی تکی[۱] که نرخ پرس و جو کامپیوتر بالا است. خود شناسایی[۲] که Botها به راحتی شناسایی می‌شوند و شناسایی توزیع شده[۳] که نرخ پرس و جو کامپیوتر کم می‌شود ولی پیدا کردن این نوع خیلی سخت است مگر با استفاده از خاصیت زمانی.

این مقاله نشان می‌دهد که گره‌هایی که دارای درجه خروجی بالایی بودند، Botهای شناخته شده بودند و گره‌هایی که مورد پرس و جو قرار گرفته بودند، Botهای جدید و ناشناخته بودند. پس با ساخت گراف پرس و جو و تهیه لیست Botهای شناخته شده، می‌توان Botهای جدید را پیدا کرد.

همچنین می‌توان با استفاده از روش مسموم کردن شناسایی[۴] (اگر کامپیوتر در لیست بود، بگوییم نیست و اگر کامپیوتر در لیست نبود، بگوییم هست)، Botmasterها را گمراه کرد ولی مشکل شناسایی نادرست دارد.

روش استفاده DNSBL برای شناسایی Botها ممکن است در مواردی خاص مفید باشد، اما به طور کلی برقرار نیست و ممکن است تعداد زیادی شناسایی نادرست تولید کند. در نتیجه این روش منحصر به تعداد محدودی از Botnetهای تولیدکننده اسپم است.

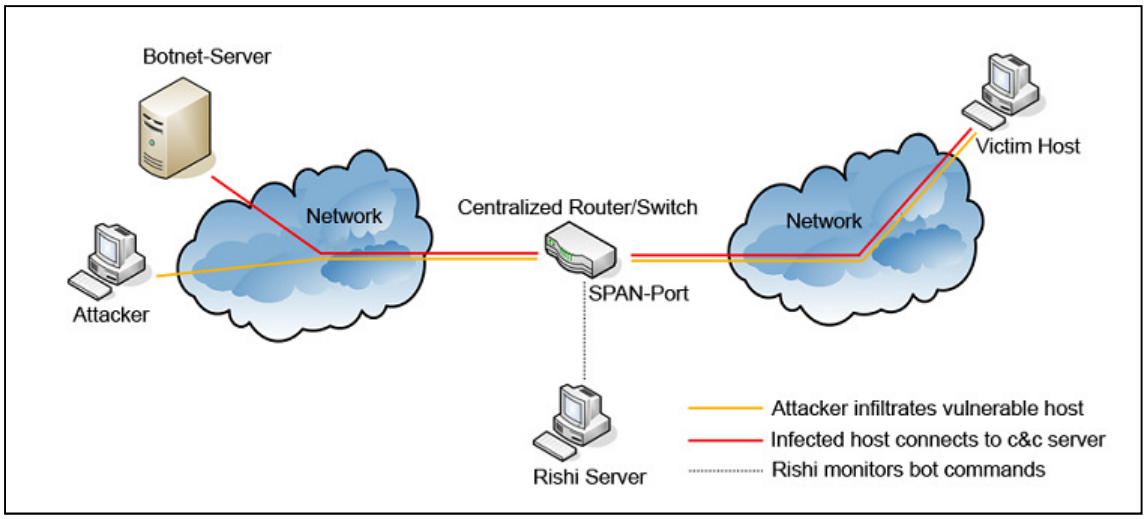
روش هشتم [۶]: Rishi: Identify bot contaminated hosts by irc nickname evaluation

خصوصیات: مبتنی بر شبکه، مبتنی بر امضاء و ناهنجاری، منفعل، فاز شناسایی: عملیاتی، شناسایی گروهی از Botها، وابسته به ساختارهای متمرکز دستور و کنترل، داده مورد نیاز: کل بسته.

---

[1] Third-party or Single-host Reconnaissance
[2] Self-Reconnaissance
[3] Distributed Reconnaissance
[4] Reconnaissance Poisoning

شرح: این روش یک سیستم شناسایی Botnet مبتنی بر امضاء و چت را معرفی می‌کند که با استفاده از الگوهای شناخته شده اسم خاص[1] Botها، سعی در شناسایی آن‌ها دارد. مشابه ابزارهای مبتنی بر امضاء مانند آنتی‌ویروس‌ها و IDSها[2]، این روش در صورتی دقیق است که پایگاه امضاءیی فراگیر و دقیق در اختیار باشد. به هر حال این روش تمام ضعف‌های ذاتی روش‌های مبتنی بر امضاء را به همراه دارد و توانایی شناسایی Botهایی را که الگوی اسم خاص آن‌ها شناخته شده نیست را ندارد. Botها بعد از آلوده کردن سیستم‌ها، اقدام به برقراری ارتباط با سرور دستور و کنترل می‌کنند. اصولاً اسم‌های خاص استفاده شده توسط Botها دارای شباهت‌هایی هستند. هر Bot موجود در یک کانال چت باید دارای یک اسم خاص یکتا باشد. برای یکتا بودن این اسم خاص، باید چند کلمه ثابت را با یک عدد تصادفی به هم متصل کرد. (مثل USA|016887436 یا DE|028509327). استفاده از مکان نصب Bot (اسم کشورها)، سیستم عامل مورد استفاده، نوع Botnet و ... برای کلمه‌های ثابت، یکی دیگر از روش‌های شناسایی Botها است. شکل ۲-۲ نمایی کلی از نحوه کار Rishi را نشان می‌دهد.



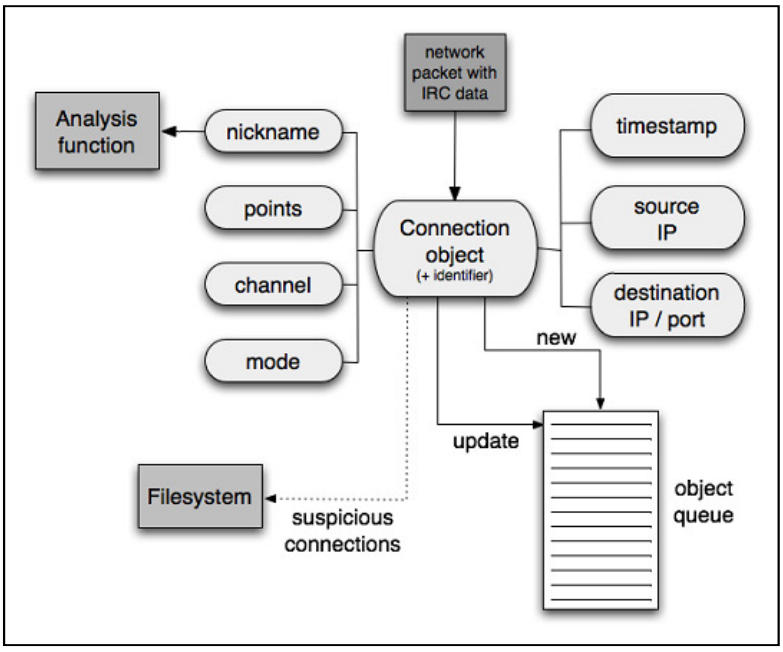
شکل ۲-۲) نمایی کلی از نحوه کار Rishi (برگرفته از [۶])

این روش ابتدا با مانیتور کردن بسته‌های TCP، وقوع یکی از دستورات NICK، JOIN، USER، QUIT و MODE را شناسایی می‌کند. از بسته‌هایی که از مرحله قبل بدست می آیند، زمان اتصال، IP و پورت کامپیوتر مبدأ[3]، IP و پورت سرور چت، کانال‌های عضو شده و اسم‌های خاص مورد استفاده استخراج می‌شوند.

هر اتصال با IP مبدأ و IP/پورت مقصد[4] مشخص می‌شود و اطلاعات فوق برای هر اتصال نگهداری می‌شود. اگر

[1] Nickname
[2] Intrusion Detection System
[3] Source
[4] Destination

اتصالی وجود داشت، اطلاعات آن به روز می‌شود و در غیر اینصورت اتصال جدید ساخته می‌شود. با دیدن دستور QUIT اتصال از سیستم حذف می‌شود. شکل ۲-۳ معماری سیستم Rishi را نشان می‌دهد. در این سیستم برای امتیاز دادن به هر اسم خاص یک تابع امتیازدهی استفاده می‌شود و هر چه این امتیاز بیشتر باشد، احتمال Bot بودن کامپیوتر بیشتر است. اسم‌های خاص با امتیاز بالاتر از یک آستانه[1]، به عنوان Bot شناخته می‌شوند و اسم‌های خاص با امتیاز صفر، به لیست سفید[2] پویا اضافه می‌شوند.



شکل ۲-۳) معماری سیستم Rishi (برگرفته از [۶])

در این سیستم چند معیار برای امتیازدهی معرفی شده‌اند که عبارتند از:

۱. وجود زیررشته‌های[3] مشکوک در اسم خاص مانند اسم Botها (مثل RBot)، اسم مخفف کشورها (مثل USA) و اسم سیستم‌های عامل (مثل XP و 2K) که به ازای هر زیر رشته مشکوک، یک واحد به امتیاز اضافه می‌کنیم.

۲. کاراکترهای خاص مانند |,[,] که به ازای هر کاراکتر خاص، یک واحد به امتیاز اضافه می‌کنیم.

۳. رشته‌های عددی طولانی که به ازای هر دو رقم پشت سرهم، یک واحد به امتیاز اضافه می‌کنیم.

در مواردی بیشتر از یک واحد به امتیاز اضافه می‌شود که شامل مطابقت با عبارات منظم[4] (عبارات منظم در

[1] Threshold
[2] Whitelist
[3] Substring
[4] Regular Expression

حقیقت نشان‌دهنده اسم‌های خاصی هستند که توسط Botهای شناخته شده استفاده شده‌اند. هر اسم خاصی که با یکی از این عبارات منظم مطابقت داشته باشد، Bot شناخته می‌شود)، اتصال به یک سرور موجود در لیست سیاه و استفاده از یک اسم خاص موجود در لیست سیاه می‌شود. لیست سفید ایستا[1] از سه بخش IP مبدأ، IP مقصد و اسم خاص تشکیل شده است. لیست سفید پویا شامل اسم‌های خاصی است که امتیاز صفر گرفته‌اند و با روش n-gram، اسم‌های خاص جدید با این لیست مطابقت داده می‌شوند. لیست سیاه شامل اسم‌های خاص با امتیاز بیشتر از آستانه و سرورهای مقصد مشکوک است.

پس به طور کلی نحوه کار این روش بدین صورت است که بعد از مشاهده یک اسم خاص جدید، ابتدا آدرس مبدأ، آدرس مقصد و اسم خاص (با روش n-gram) را با لیست سفید مقایسه می‌کند و در صورت مطابقت هر یک از موارد فوق، اسم خاص جدید را به لیست سفید اضافه می‌کند. در غیر اینصورت، آدرس مقصد و اسم خاص (با روش n-gram) را با لیست سیاه مقایسه می‌کند و در صورت مطابقت هر یک از موارد فوق، اسم خاص جدید را به لیست سیاه اضافه می‌کند. در غیر اینصورت، با استفاده از تابع امتیازدهی، یک امتیاز به اسم خاص تعلق می‌گیرد. اگر امتیاز داده شده، صفر باشد، اسم خاص به لیست سفید منتقل شده و اگر امتیاز داده شده بالاتر از یک آستانه باشد، اسم خاص به لیست سیاه اضافه می‌شود.

روش نهم [۳۰] و [۳۱]: Using machine learning techniques to identify botnet traffic و Detecting botnets with tight command and control

خصوصیات: مبتنی بر شبکه، مبتنی بر امضاء و ناهنجاری، منفعل، فاز شناسایی: عملیاتی، شناسایی گروهی از Botها، وابسته به ساختارهای متمرکز دستور و کنترل، داده مورد نیاز: اتصال.

شرح: این روش مبتنی بر یادگیری ماشین[2] است که از تعدادی خصوصیات مربوط به ترافیک شبکه چت مانند پروتکل چت استفاده می‌کند. این روش از دو مرحله تشکیل شده است: تمییز دادن ترافیک‌های چت از غیر چت و تمییز دادن ترافیک چت Botnet از چت نرمال. شکل ۲-۴ ساختار Botnetهای مبتنی بر چت را نشان می‌دهد. در الگوریتم‌های کلاس‌بندی، پیدا کردن مجموعه آموزش[3] خیلی اهمیت دارد. برای این کار مراحل زیر را باید انجام داد:

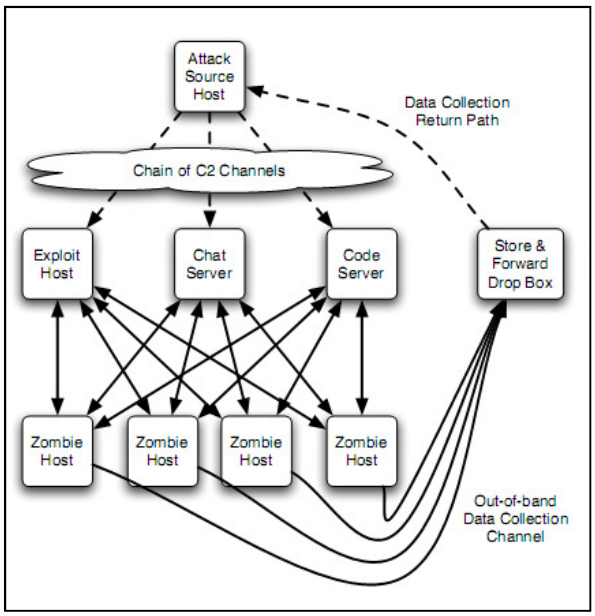
۱. از روی پورت پیش فرض و محتوای بسته می‌توان ترافیک‌های چت را تشخیص داد.

---

[1] Static
[2] Machine Learning
[3] Training Set

۲. استفاده از یک محیط تست برای تولید ترافیک‌های Botnet، استفاده از ترافیک کامپیوترهای آلوده (کامپیوترهایی که عملیات اسکن کردن انجام می‌دهند) و بررسی محتوای بسته‌ها برای پیدا کردن Botnetهای شناخته شده.

جدول ۲-۶ خصوصیات ارتباطات شبکه برای کلاس‌بندی را نشان می‌دهد.



شکل ۲-۴) ساختار Botnetهای مبتنی بر چت (برگرفته از [۳۰])

جدول ۲-۶) خصوصیات ارتباطات شبکه برای کلاس‌بندی (برگرفته از [۳۰])

start/end	Flow start/end times
IP-proto	IP protocol of flow
TCP flags	Summary of TCP SYN/FIN/ACK flags
pkts	Total pkts exchanged in flow
Bytes	Total Bytes exchanged in flow
pushed pkts	Total packets pushed in flow
duration	Flow duration
maxwin	Maximum initial congestion window
role	Whether client or server initiated flow
Bpp	Average Bytes-per-packet for flow
bps	Average bits-per-second for flow
pps	Average packets-per-second for flow
PctPktsPushed	Percentage of packets pushed in flow
PctBppHistBin0–7	Percent of packets in one of eight packet size bins; these variables collectively form a histogram of packet size for flow
varIAT	Variance of packet inter-arrival time for flow
varBpp	Variance of Bytes-per-packet for flow

از آنجایی که تعداد ارتباطات می‌تواند زیاد باشد، برای کاهش دادن تعداد آن‌ها از سه روش زیر استفاده می‌کنیم:

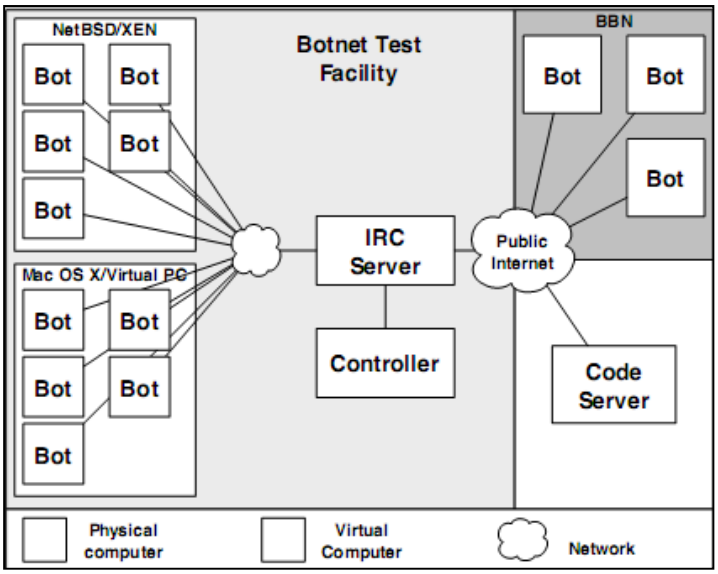
۱. حذف ارتباطات مربوط به اسکن کردن پورت: اصولاً این ارتباطات فقط شامل TCP SYN یا

TCP RST هستند و به طور کامل برقرار نشده‌اند.

۲. حذف ارتباطات با پنهای باند بالا مانند نظیر به نظیر و صفحات وب سنگین.

۳. حذف ارتباطاتی که دارای تعداد کمی بسته بوده یا مدت زمان کمی فعال بودند.

شکل ۲-۵ معماری محیط تست Botnet را نشان می‌دهد.



شکل ۲-۵) معماری محیط تست Botnet (برگرفته از [۳۰])

روش دهم [۳۲]: Wide-scale botnet detection and characterization

خصوصیات: مبتنی بر شبکه، مبتنی بر ناهنجاری، منفعل، فاز شناسایی: عملیاتی، شناسایی گروهی از Botها، وابسته به ساختارهای متمرکز دستور و کنترل، داده مورد نیاز: اتصال.

شرح: این روش مطالعاتی در زمینه شناسایی سرورهای دستور و کنترل از روی خصوصیات مربوط به اتصالات شبکه[1] انجام داده است. در این روش اتصالات شبکه دارای فیلدهای IP و پورت مبدأ، IP و پورت مقصد، تعداد بسته‌ها، تعداد بایت‌ها، OR پرچم‌های مورد استفاده در TCP، زمان شروع، زمان پایان و پروتکل لایه انتقال (TCP[2] یا UDP[3]) می‌باشند. شکل ۲-۶ ارتباط بین Bot، کنترلر و Botmaster را نشان می‌دهد.

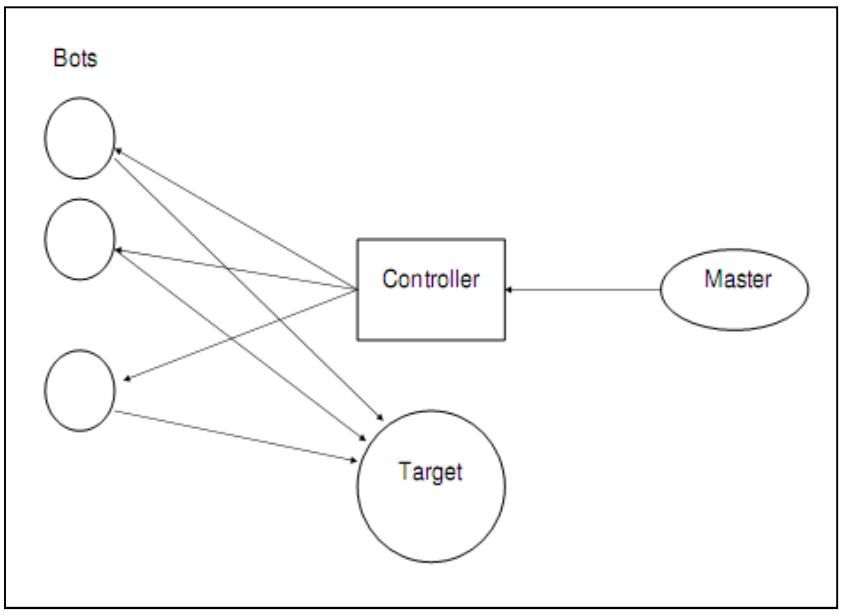
مراحل شناسایی کنترلر Botnet در این روش عبارتند از:

۱. جمع‌آوری Trigger Eventها. شناسایی کامپیوترهای با رفتار مشکوک و انتخاب ارتباطات به/از این

[1] Network Flows
[2] Transmission Control Protocol
[3] User Datagram Protocol

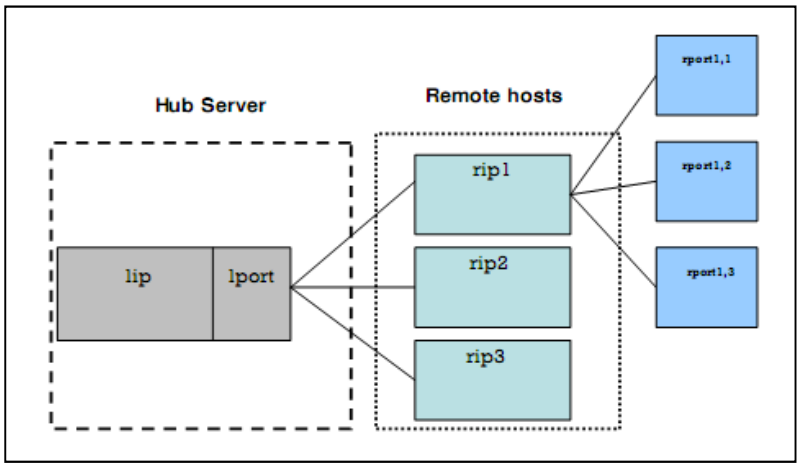
کامپیوترها که شامل هشدار[1]های مربوط به کرم‌ها، فعالیت‌های مربوط به اسکن (یک پورت توسط تعداد زیادی کامپیوتر اسکن می‌شود)، فعالیت‌های مربوط به ارسال اسپم، لیست‌های مراقب و فعالیت‌های مربوط به DDoS می‌باشد.

۲. تحلیل ارتباطات و شناسایی گفتگوهای[2] کنترلرهای کاندید. یک گفتگو شامل تعدادی ارتباط بین یک کامپیوتر محلی (مشکوک به Bot) و یک کامپیوتر ریموت، روی یک پورت ریموت مورد نظر است. برای تعیین کنترکرهای کاندید می‌توان از سه روش استفاده کرد. اولین روش، کنترلرهایی است که از پورت‌های استاندارد چت استفاده می‌کنند. دومین روش، کامپیوترهای ریموتی که نقش هاب سرور دارند (کامپیوترهای هاب سرور به کامپیوترهایی گفته می‌شود که چند اتصال از کامپیوترهای مشکوک روی یک یا چند پورت خود دارند). این روش به دنبال جفت lip/lport می‌گردد که دارای اتصال با چند rip یا اتصال با یک rip و چند rport باشد. کامپیوترهایی که اسکن انجام می‌دهند نیز هاب سرور محسوب می‌شوند. برای پردازش ارتباطات از آدرس مبدأ و مقصد استفاده می‌کند. روش سوم، استفاده از یک مدل مرجع برای شناسایی گفتگوهای کنترلر کاندید است. هر گفتگو شامل فیلدها IP کامپیوتر، IP سرور ریموت، پورت سرور ریموت، تعداد ارتباطات، تعداد بسته‌ها، تعداد بایت‌ها و برچسب زمانی[3] اولین و آخرین ارتباط می‌باشد. شکل ۲-۷ نمایی از هاب سرور را نشان می‌دهد.



شکل ۲-۶) ارتباط بین Bot، کنترلر و Botmaster (برگرفته از [۳۲])

[1] Alert
[2] Conversation
[3] Time Stamp



شکل ۲-۷) نمایی از هاب سرور (برگرفته از [۳۲])

۳. تحلیل گفتگوهای کنترلرهای کاندید. ابتدا تعداد کامپیوترهای مشکوک متصل به یک سرور ریموت (IP/Port) را محاسبه می‌شود. برای این کار بعد از مرتب کردن سرورها بر حسب تعداد کامپیوتر مشکوک، سرورهایی را انتخاب می‌کند که از یک آستانه‌ای بیشتر باشند. سپس فاصله بین ترافیک به پورت سرورهای ریموت و ترافیک مدل مرجع را محاسبه می‌کند. برای این کار از متریک‌هایی مانند تعداد اتصالات یک آدرس[۱]، تعداد بسته‌های یک اتصال[۲] و تعداد متوسط بایت‌ها در یک بسته[۳] استفاده می‌شود. به ازای هر پورت، روی تمام ارتباطات بین تمام کامپیوترهای مشکوک و تمام سرورها رابطه (۲-۱۰) را محاسبه کرده و پورت‌هایی که فاصله‌شان از یک آستانه کمتر باشد، به عنوان پورت کاندید انتخاب می‌کند. بعد از این دو مرحله، امتیاز سرور ریموت را محاسبه می‌کند. برای سرور ریموتی (IP/Port) که هر دو شرط داشتن کامپیوترهای مشکوک متصل و فاصله ترافیک کم با ترافیک مدل را دارا باشد، امتیازی محاسبه می‌شود. تعداد کامپیوترهای بیکار سرور ریموت یک مؤلفه برای محاسبه امتیاز محسوب می‌شود. اگر کامپیوتری که به یک سرور ریموت متصل بود و زمان بین[۴] ارتباطات آن با سرور ریموت یک الگوی مشخص داشت، به آن سرور امتیاز بیشتری اختصاص داده می‌شود.

$$D_P=\frac{1}{n}\sum_{i=1}^{N_m}\sqrt{\sum_{j=1}^{N_s}\left(X_{ij}-M_{ij}\right)^2} \qquad \text{(برگرفته از [۳۲])} \quad (۲-۱۰)$$

بعد از شناسایی کنترلر، کامپیوترها بر حسب پورت‌های مورد استفاده دسته‌بندی می‌شوند. هر یک از کامپیوترها را با یک آرایه که شامل پورت‌های مورد استفاده است در نظر می‌گیرد. پورت‌ها را بر اساس تعداد اتصالاتی که این

[1] Flows per address
[2] Packets per flow
[3] Bytes per packet
[4] Interval Time

کامپیوتر با آن‌ها برقرار کرده مرتب کرده و با استفاده از رابطه (۲-۱۱) شباهت آن‌ها را بدست می‌آورد و با یک الگوریتم خوشه‌بندی آن‌ها را دسته‌بندی می‌کند تا کامپیوترهای با رفتار مشابه را دسته‌بندی کند.

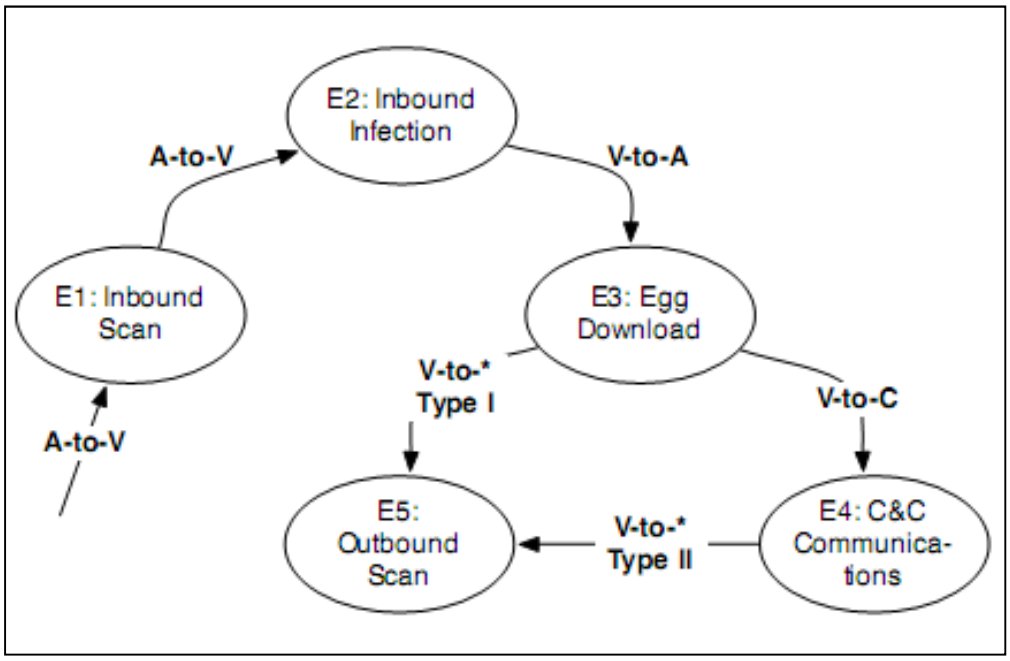
$$S(i,j)=\frac{\sum_{k=1}^{M} I_k(M-O_i+1)(N-O_j+1)}{N(N+1)(2N+1)/6} \qquad (۲-۱۱)$$ (برگرفته از [۳۲])

روش یـازدهم [۳۳]: BotHunter: Detecting malware infection through ids-driven dialog correlation

خصوصیات: مبتنی بر شبکه، مبتنی بر امضاء و ناهنجاری، منفعل، فاز شناسایی: آماده‌سازی و عملیـاتی، شناسـایی یک Bot مجزا، عدم وابستگی به ساختارها و پروتکل‌های دستور و کنترل، داده مورد نیاز: کل بسته.

شرح: این روش بر اساس الگوی تعریف شده توسط کاربر بین هشـدارهای IDS همبسـتگی ایجـاد مـی‌کنـد. ایـن سیستم مفهومی به نام همبستگی دیالوگ[۱] را معرفی می‌کند که بدین معنی است که آلودگی سیستم‌ها توسط Bot بـا یک مجموعه ارتباطات مدل می‌شود که این ارتباطات بین یک کامپیوتر داخلی و یک یا چند کـامپیوتر خـارجی برقـرار هستند و با یک ترتیب زمانی به دنبال هم ایجاد می‌شوند. همه Botها یـک وجـه مشـترک دارنـد و آن چرخـه حیـات نفوذشان به سیستم است که شامل اسکن کردن هدف[۲]، سوء استفاده از آسیب‌پـذیری[۳]، دانلـود بـاینری و اجـرای آن[۴]، برقراری ارتباط با دستور و کنترل[۵] و اسکن کردن کامپیوترهای بیرونی[۶] است. برای نفوذ یک Bot، همـه مراحـل فـوق لزوماً با هم برقرار نیستند. شکل ۲-۸ مدل دیالوگ نفوذ Bot را نشان می‌دهد.

[1] Dialog Correlation
[2] Target Scanning
[3] Expliot
[4] Binary Egg Download & Execution
[5] C&C Channel Establishment
[6] Outbound Scanning



شکل ۲-۸) مدل دیالوگ نفوذ Bot (برگرفته از [۳۳])

شکل ۲-۹ مراحل دیالوگ نفوذ Bot را نشان می‌دهد.

– E1: External to Internal Inbound Scan
– E2: External to Internal Inbound Exploit
– E3: Internal to External Binary Acquisition
– E4: Internal to External $C\&C$ Communication
– E5: Internal to External Outbound Infection Scanning

شکل ۲-۹) مراحل دیالوگ نفوذ Bot (برگرفته از [۳۳])

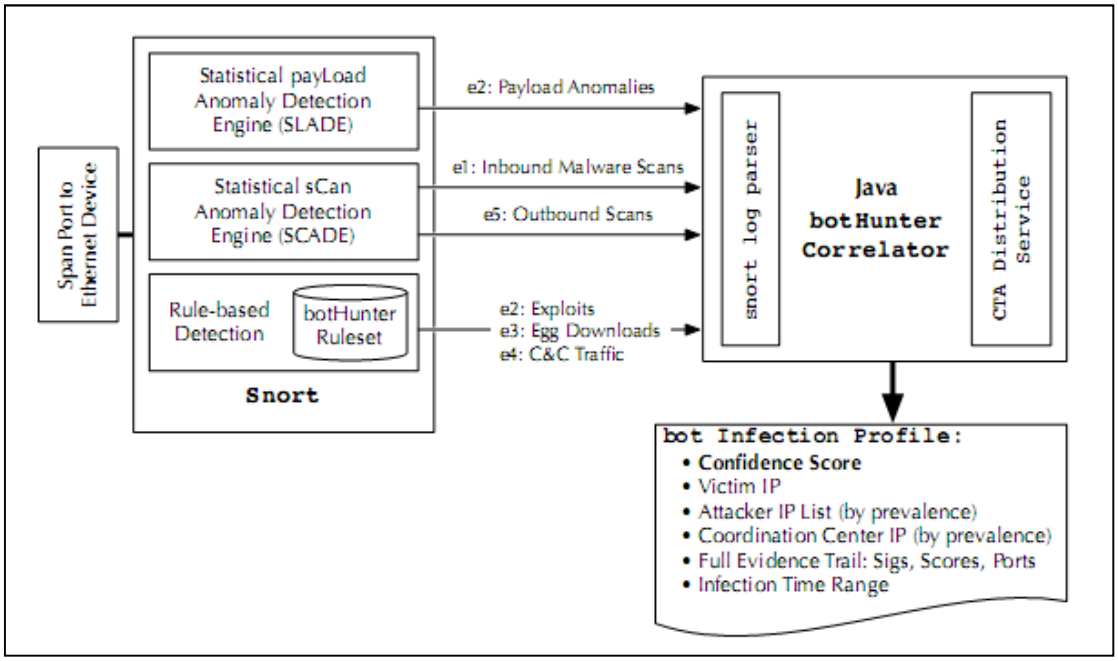
شکل ۲-۱۰ شروط ایجاد همبستگی بین دیالوگ‌ها را نشان می‌دهد.

Condition 1: Evidence of local host infection (E2), AND evidence of outward bot coordination or attack propagation (E3-E5); or

Condition 2: At least two distinct signs of outward bot coordination or attack propagation (E3-E5).

شکل ۲-۱۰) شروط ایجاد همبستگی بین دیالوگ‌ها (برگرفته از [۳۳])

شکل ۲-۱۱ معماری سیستم BotHunter را نشان می‌دهد.



شکل ۲-۱۱) معماری سیستم BotHunter (برگرفته از [۳۳])

روش دوازدهـــم [۱۸]: BotSniffer: Detecting botnet command and control channels in network traffic

خصوصیات: مبتنی بر شبکه، مبتنی بر امضاء و ناهنجاری، منفعل، فـاز شناسـایی: عملیـاتی، شناسـایی گروهـی از Botها، وابسته به ساختارهای متمرکز دستور و کنترل، داده مورد نیاز: کل بسته.

شرح: این روش با انجام دادن یک تحلیل گروهی فضایی-زمـانی[1] بـر روی کـامپیوترهـای داخـل شـبکه، سـعی در شناسایی سرورهای دستور و کنترل متمرکز (چت و HTTP) دارد. از آنجایی که Botهـای متعلـق بـه یـک Botnet، یک فایل باینری را اجرا می‌کنند، از حیث نحوه پاسخ به دستورات و حملات شبیه به هم هستند. Botهایی که در یـک Botnet قرار دارند، هنگام پاسخ دادن دارای شباهت و تناظر زمانی و مکانی هستند (دارای تکثیر، ارتباط و فعالیت‌های شیادانه و حملات هماهنگ هستند). دلیل این هماهنگی بین Botهـا بـه خـاطر یکسـان بـودن نحـوه پاسـخگویی بـه دستورات[2] است. در Botnetهای مبتنی بر چت، Botmaster با استفاده از PRIVMSG با Botها گفتگو می‌کند. Botها ترافیک پیام و فعالیت مشابه، در یک پنجره زمانی مشابه ارسال می‌کنند. این رفتار Botها منجر به مشاهده

[1] Spatio-Temporal Group Analysis
[2] Spatial-Temporal Correlation and Similarity

پاسخ گروهی[1] می‌شود. پاسخ Botها هماهنگ‌تر[2] و همبسته‌تر[3] از انسان‌ها است. چند رفتار مشترک بین Botnetها مستقل از ساختارشان وجود دارد:

۱. Botها باید برای گرفتن دستورات به سرورهای دستور و کنترل متصل باشند.

۲. Botها باید پاسخ دستورات دریافت شده را بدهند. به طور کل دو نوع پاسخ وجود دارد: پاسخ پیامی[4] (حالت سیستم[5]) و پاسخ فعالیتی[6] (ارسال اسپم، اسکن کردن و به روز رسانی فایل باینری).

سیستم BotSniffer از دو مؤلفه اصلی تشکیل شده است:

۱. موتور مانیتورینگ[7] که وظیفه آن تولید اطلاعات مربوط به پروتکل‌های دستور و کنترل مشکوک، شناسایی رفتارهای پاسخ فعالیتی (اسکن کردن، ارسال اسپم و به روز رسانی فایل باینری)، شناسایی رفتارهای پاسخ پیامی (PRIVMSG) است. این مؤلفه از سه بخش تشکیل شده است:

أ. پیش‌پردازش[8]: این بخش وظیفه فیلتر کردن ترافیک‌های نامربوط برای کاهش ترافیک، فیلتر کردن ترافیک‌های مربوط به پروتکل‌های غیر معمول دستور و کنترل مثل ICMP[9] و UDP، استفاده از لیست سفید برای فیلتر کردن ترافیک سرورهایی که دستور و کنترل نیستند مثل یاهو و گوگل و استفاده از لیست مراقب برای در نظر گرفتن کامپیوترهایی که در آن قرار دارند و نه تمام کامپیوترهای موجود در شبکه.

ب. تطبیق‌دهنده پروتکل دستور و کنترل[10]: این بخش وظیفه شناسایی کامپیوترهایی را دارد که از پروتکل‌های دستور و کنترل استفاده می‌کنند. برای شناسایی چت از پیام‌های PASS، NICK و USER و برای شناسایی HTTP از پیام‌های GET، POST و HEAD استفاده می‌شود.

ت. تشخیص پاسخ پیامی و پاسخ عملیاتی: این بخش برای کامپیوترهایی که دارای ارتباطات HTTP یا چت بودند، پاسخ پیامی مانند پیام‌های PRIVMSG و پاسخ فعالیتی مانند اسکن کردن (نرخ بالای اسکن و نرخ بالای اتصالات برقرار نشده)، ارسال اسپم (بررسی پرسش‌های DNS با رکورد MX و اتصالات

---

[1] Response Crowd
[2] Synchronization
[3] Correlation
[4] Message Response
[5] System Status
[6] Activity Response
[7] Monitor Engine
[8] Preprocessing
[9] Internet Control Message Protocol
[10] C&C Protocol Matcher

SMTP) و دانلود باینری را تشخیص می‌دهد.

۲. موتور همبستگی[1]: این مؤلفه ابتدا کامپیوترها را بر اساس IP/Port مقصد گروه‌بندی می‌کند. این مؤلفه برای پیدا کردن سرورهای دستور و کنترل سه الگوریتم معرفی کرده است:

أ. Response-Crowd-Density-Check: در این الگوریتم به ازای هر گروه، در بازه‌های زمانی مختلف، چک می‌کند که آیا بیشتر از یک آستانه (مثلاً ۵۰٪) تعداد کل کامپیوترهای موجود در آن گروه، پاسخ پیامی یا فعالیتی داریم یا خیر که اصطلاحاً به این حالت تراکم[2] می گوییم. این روش برای پاسخ فعالیتی خوب است چرا که در مواقعی پاسخ‌های پیامی کم هستند ولی پاسخ‌های فعالیتی زیاد اتفاق می‌افتند.

ب. Response-Crowd-Homogeneity-Check: در این الگوریتم فرض بر این است که Botها پاسخ‌های پیامی ساختار و محتوای مشابهی دارند. همچنین Botها پاسخ‌های فعالیتی مشابهی دارند مثل اسکن کردن محدوده IP و پورت یکسان. به ازای هر گروه، این الگوریتم، در بازه‌های زمانی مختلف، بزرگترین خوشه کامپیوترهای به هم شبیه را بدست آورده و بر تعداد کل کامپیوترها تقسیم می‌کند و اگر از یک آستانه گذشت، آن گروه را همگن در نظر می‌گیرد. در یک بازه زمانی، اگر یک کامپیوتر بیش از یک پیام فرستاده باشد، پیام‌ها را به هم می چسبانیم و یک پیام درست می‌کنیم. پیام‌های کامپیوترها را دو به دو مقایسه می‌کنیم و اگر تعداد پیام‌های مشابه از یک آستانه بیشتر بود، آن بازه زمانی را همگن در نظر می‌گیریم.

ت. Activity-Response-Crowd-Homogeneity-Check: در این الگوریتم از اسکن کردن (آنتروپی یا پراکندگی IPها و پورت‌های اسکن شده شبیه است) ، ارسال اسپم (تعداد سرورهای میل مشترک و شباهت ساختار و محتوای (URLs) میل‌های اسپم، و دانلود باینری (توزیع بایت‌ها[3] یا آنتروپی فواصل بین دو فایل باینری) استفاده می‌شود.

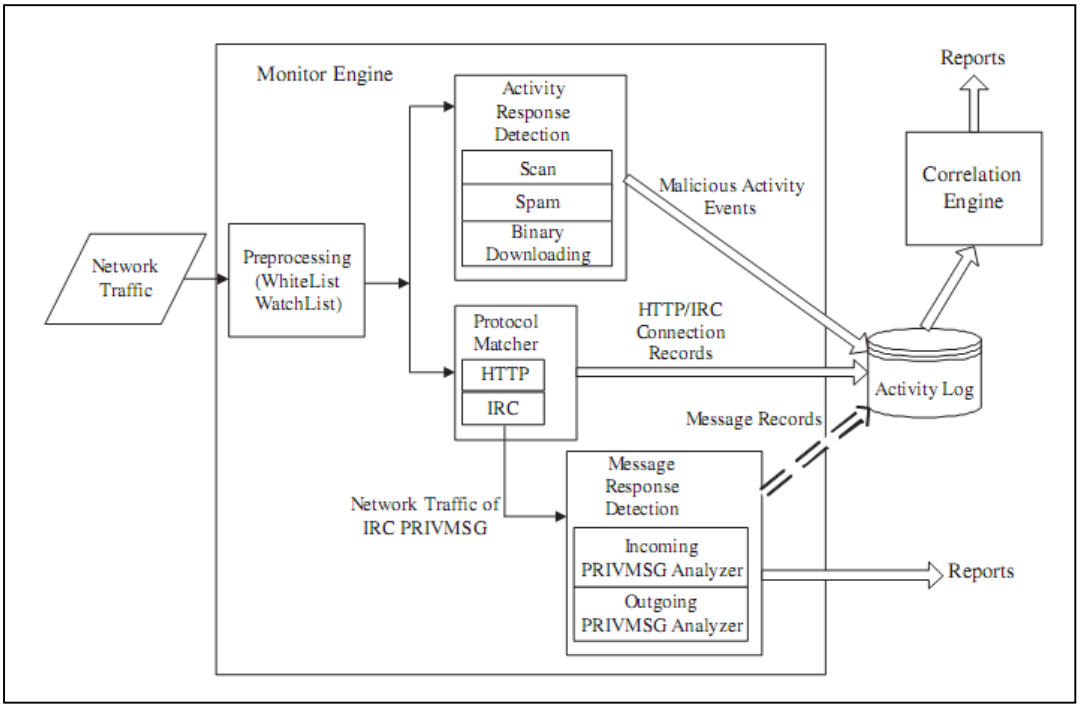
این روش قابلیت شناسایی سرور دستور و کنترل با یک کامپیوتر را نیز دارد. در پروتکل چت، پیام هر کامپیوتر در کانال اعلام همگانی[4] می‌شود. بنابراین هر کامپیوتر می‌تواند پیام تمام کامپیوترهای موجود در کانال را بشنود. پس می‌توان به جای چک کردن همگن بودن پیام‌های کامپیوترهای در یک گروه، می‌توان همگن بودن پیام‌های ورودی به

[1] Correlation Engine
[2] Dense
[3] Byte Distribution
[4] Broadcast

یک کامپیوتر را بررسی کرد. این روش برای حالاتی که خاصیت اعلام همگانی غیر فعال شده باشد کار نمی‌کند. Botهای مبتنی بر HTTP به صورت متناوب به سرور دستور و کنترل تماس برقرار می‌کنند تا دستورات جدید را دریافت کنند. با روش خود همبستگی[1] می‌توان این Botها را شناسایی کرد. خود همبستگی نسبت به آنالیز گروهی بدتر جواب می‌دهد. همچنین می‌توان از روش‌های حریصانه‌ای[2] استفاده کرد:

- اینکه کاربران نرمال چت به ندرت از دستوراتی مانند who, whois, list, names استفاده می‌کنند.
- در سرورهای چت نرمال اعلام همگانی فعال است، پس تعداد پیام‌های ورودی یک کامپیوتر چت به مراتب بیشتر از تعداد پیام‌های خروجی است. ولی در Botnetها اصولاً خاصیت اعلام همگانی غیرفعال است و تعداد پیام‌های ورودی یک کامپیوتر چت تقریباً با تعداد پیام‌های خروجی یکسان است.

شکل ۲-۱۲ معماری سیستم BotSniffer را نشان می‌دهد.



شکل ۲-۱۲) معماری سیستم BotSniffer (برگرفته از [۱۸])

این روش همچنین دارای محدودیت‌هایی است که به شرح زیرند:

۱. لیست سفید: Botmaster می‌تواند از کامپیوترهای موجود در لیست سفید به عنوان پراکسی استفاده کند تا شناسایی نشود. یا اینکه Botmaster ابتدا از سرور دستور و کنترل به صورت نرمال استفاده می‌کند تا سرور

[1] Self-Correlation
[2] Heuristic

در لیست سفید قرار گیرد و سپس از آن استفاده‌های بدخواهانه می‌کند که می‌توان برای هر سرور موجود در لیست سفید یک برچسب زمانی قرار دهیم و بعد از پایان مهلت برچسب زمانی، سرور را از لیست حذف کرد.

۲. رمزنگاری: در این مورد Botmaster می‌تواند ارتباطات را رمز کند. با این حال فقط پاسخ‌های پیامی محدود می‌شوند و می‌توان باز هم از پاسخ‌های فعالیتی استفاده کرد. همچنین می‌توان از روش‌های دیگری برای شناسایی پیام‌های همگن استفاده کرد.

۳. گول زدن تطابق پروتکل.

۴. گول زدن با تأخیر زیاد در جواب که این به نفع Botmaster نیست چرا که کارایی Botnet را پایین می‌آورد.

۵. گول زدن با تزریق بسته‌های نویز تصادفی، گذاشتن اطلاعات آشغال در بسته و استفاده از تأخیرهای تصادفی در موقع جواب دادن که با این حال فقط پاسخ‌های پیامی محدود می‌شوند و می‌توان باز هم از پاسخ‌های فعالیتی استفاده کرد. همچنین می‌توان با بزرگتر کردن پنجره زمانی بر این مشکل فائق آمد.

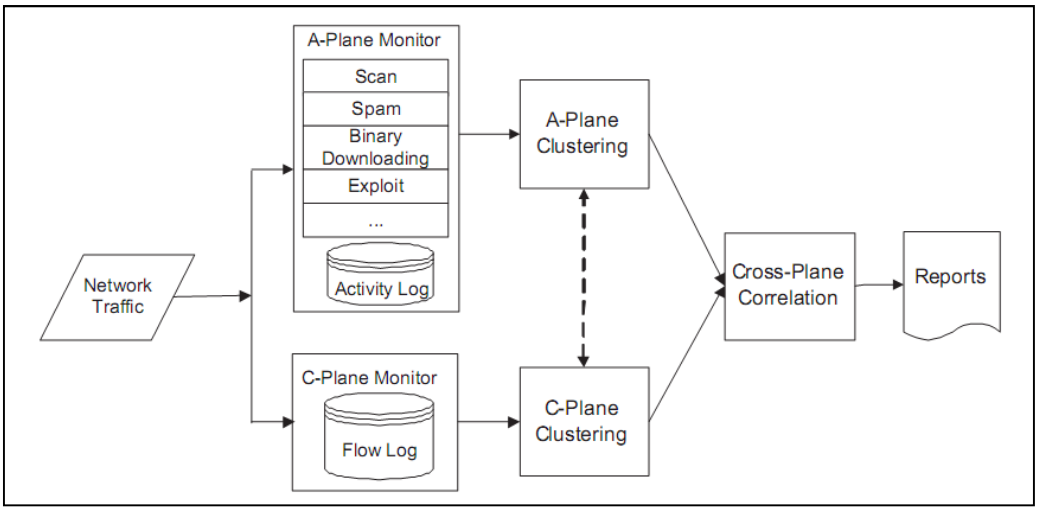
روش سیزدهم [۱۷]: BotMiner: Clustering analysis of network traffic for protocol and structure independent botnet detection

خصوصیات: مبتنی بر شبکه، مبتنی بر امضاء و ناهنجاری، منفعل، فاز شناسایی: عملیاتی، شناسایی گروهی از Botها، عدم وابستگی به ساختارها و پروتکل‌های دستور و کنترل، داده مورد نیاز: کل بسته.

شرح: این روش کامپیوترهای داخل شبکه را بر اساس خصوصیات اتصالشان (تعداد بایت‌ها، تعداد بسته‌ها و ...) و فعالیت‌های بدخواهانه‌شان (اسکن کردن، اسپم فرستادن، دانلود باینری و سوء استفاده از آسیب‌پذیری‌ها) خوشه‌بندی می‌کند و سپس با همبستگی ایجاد کردن بین این خوشه‌ها Botهای داخل شبکه را پیدا می‌کند. نحوه کار این روش بدین صورت است که ابتدا تعیین می‌کند چه کسی با چه کسی صحبت می‌کند (شناسایی ارتباطات دستور و کنترل) و ارتباطات شبیه را خوشه‌بندی می‌کند (C-plane). سپس تعیین می‌کند چه کسی چه کاری انجام می‌دهد (شناسایی فعالیت‌های بدخواهانه) و فعالیت‌های شبیه را خوشه‌بندی می‌کند (A-plane). در پایان این دو خوشه‌بندی را ارتباط می‌دهد و کامپیوترهایی که الگوی ارتباطات دستور و کنترل و فعالیت‌های خرابکاری شبیهی دارند را شناسایی می‌کند.

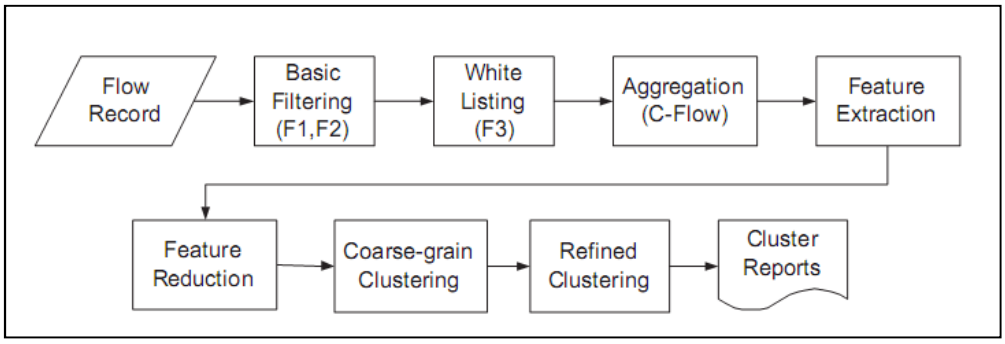
Botnetها بوسیله کانال ارتباط دستور و کنترل و فعالیت‌های بدخواهانه از دیگران متمایز می‌شوند. کرم‌ها فعالیت‌های بدخواهانه انجام می‌دهند، ولی ارتباطات دستور و کنترل ندارند. نرم‌افزارهایی مثل کامپیوتر چت ارتباطات

دستور و کنترل دارند، ولی فعالیت‌های بدخواهانه انجام نمی‌دهند. هدف این روش مانیتور کردن یک شبکه و یافتن کامپیوترهای آلوده‌ای که جزء یک Botnet هستند، می‌باشد. هر ارتباطات دربردارنده خصوصیات زمان، مدت، IP مبدأ، پورت مبدأ، IP مقصد، پورت مقصد، تعداد بسته‌ها و بایت‌های انتقال داده شده است. شکل ۲-۱۳ معماری سیستم BotMiner را نشان می‌دهد.



شکل ۲-۱۳) معماری سیستم BotMiner (برگرفته از [۱۷])

سیستم مانیتورینگ A-plane فعالیت‌های اسکن کردن، انتشار، حملات DDoS، ارسال اسپم، دانلود فایل‌های باینری، سوء استفاده از آسیب‌پذیری زیر را ثبت می‌کند. ماژول مانیتورینگ A-plane روی Snort پیاده‌سازی شده است. ماژول خوشه‌بندی C-plane وظیفه خوشه‌بندی کامپیوترهای با الگوی ارتباطی شبیه است. شکل ۲-۱۴ معماری خوشه‌بندی C-plane را نشان می‌دهد.



شکل ۲-۱۴) معماری خوشه‌بندی C-plane (برگرفته از [۱۷])

در خوشه‌بندی C-plane دو نوع فیلترینگ صورت می‌گیرد که عبارتند از:

۱. فیلترینگ پایه: این فیلترینگ دارای دو مرحله است. مرحله اول، فیلترینگ اتصالاتی است که از سمت کامپیوترهای داخل به کامپیوترهای بیرون نیستند. این اتصالات شامل اتصالاتی که بین کامپیوترهای داخل

هستند و اتصالاتی که از سمت کامپیوترهای بیرون شروع شده‌اند. مرحله دوم، فیلترینگ اتصالاتی است که به طور کامل برقرار نشده‌اند. این اتصالات شامل اتصالاتی است که فقط ترافیک یک طرفه دارند و اتصالات مربوط به اسکن کردن هستند.

۲. فیلترینگ لیست سفید: این فیلترینگ شامل فیلتر کردن اتصالاتی است که مقصدشان سرورهای قانونی مثل یاهو و گوگل هستند.

یک C-flow مجموعه‌ای از اتصالات در یک بازه زمانی مشخص (یک روز) است که دارای پروتکل، IP مبدأ، IP مقصد و پورت مقصد یکسانی هستند. خصوصیاتی که از هر C-flow می‌توان بدست آورد شامل تعداد اتصالات در هر ساعت روز (fph)، تعداد بسته‌ها در هر اتصال (ppf)، متوسط تعداد بایت‌ها در بسته‌ها (bpp) و متوسط تعداد بایت‌ها در ثانیه (bps) می‌باشد.

اتصالات C-plane بر اساس خصوصیاتی که ذکر شد، به خاطر مسائل کارایی به صورت دو مرحله‌ای خوشه‌بندی می‌شوند. هشدارهای A-plane ابتدا به چهار خوشه کلی اسکن، اسپم، سوء استفاده از آسیب‌پذیری و دانلود باینری تقسیم می‌شوند و در هر خوشه، با استفاده از پارامترهای زیر خوشه‌بندی صورت می‌گیرد:

- اسکن: در این خوشه، هشدارهای اسکن بر اساس پورت‌های اسکن شده و شبکه اسکن شده خوشه‌بندی می‌شوند.
- اسپم: در این خوشه، هشدارهای اسپم بر اساس اتصالات SMTP با مقصد یکسان، محتویات شبیه اسپم ها، URLهای شبیه در متن اسپم و فرکانس اتصالات SMTP خوشه‌بندی می‌شوند.
- سوء استفاده از آسیب‌پذیری‌ها: در این خوشه، هشدارهای مربوط به سوء استفاده از آسیب‌پذیری‌ها بر اساس شناسه Snort خوشه‌بندی می‌شوند.
- دانلود باینری: در این خوشه، هشدارهای مربوط به دانلود باینری بر اساس شباهت فایل‌های باینری صورت می‌گیرد.

بعد از خوشه‌بندی A-plane و C-plane، نوبت به ایجاد همبستگی بین خوشه‌ها می‌رسد. در مرحله ایجاد همبستگی، به هر کامپیوتر امتیازی داده می‌شود و اگر امتیاز کامپیوترها به یک آستانه مشخص رسید، آن کامپیوتر به عنوان Bot معرفی می‌شود. رابطه (۲-۱۲) نحوه محاسبه امتیاز هر کامپیوتر را نشان می‌دهد.

$$s(h)=\sum_{\substack{i,j \\ j>i \\ t(A_i)\neq t(A_j)}} \omega(A_i)\omega(A_j)\frac{|A_i\cap A_j|}{|A_i\cup A_j|}+\sum_{i,k}\omega(A_i)\frac{|A_i\cap C_k|}{|A_i\cup C_k|} \qquad \text{(۲-۱۲) (برگرفته از [۱۷])}$$

کامپیوتر h امتیاز بیشتری می‌گیرد اگر:

- A-plane با A-plane: تعداد زیادی فعالیت مشکوک انجام داده باشد (در خوشه‌های بیشتری باشد). بنابراین کامپیوترهایی که با کامپیوتر h خوشه‌بندی شده‌اند، همان رفتار h را داشته باشند.
- A-plane با C-plane: اگر h عضو خوشه‌ای از A-plane باشد که اشتراک زیادی با خوشه‌ای از C-plane داشته باشد، نشان‌دهنده این است که تعدادی کامپیوتر علاوه بر فعالیت‌های مشابه، اتصالات مشابهی هم داشته باشند.

اگر چند Bot دارای خوشه‌های A-plane مشترکی بوده و حداقل یک خوشه مشترک در C-plane داشته باشند، می‌توان آن‌ها را عضو یک Botnet در نظر گرفت. رابطه (۲-۱۳) نحوه محاسبه شباهت دو کامپیوتر را نشان می‌دهد.

$$sim(h_i,h_j)=\sum_{k=1}^{m_B} I(b_k^{(i)}=b_k^{(j)})+I(\sum_{k=m_B+1}^{m_B+n_B} I\left(b_k^{(i)}=b_k^{(j)}\right)\geq 1) \qquad \text{(۲-۱۳) (برگرفته از [۱۷])}$$

سیستم BotMiner دارای محدودیت‌هایی هم هست که عبارتند از:

- گول زدن C-plane: Botnetها می‌توانند از سرورهای موجود در لیست سفید برای دستور و کنترل استفاده کنند. همچنین Botها می‌توانند با عوض کردن سرورهای دستور و کنترل، تصادفی کردن الگوی ارتباطی هر Bot، تقلید رفتار ارتباطی کامپیوترهای نرمال توسط Botها و استفاده از کانال نهفته[1] برای پنهان کردن ارتباطات دستور و کنترل الگوی ارتباطی را تغییر دهند. Botها می‌توانند با تغییر تعداد بسته‌ها در اتصال (تزریق بسته‌های تصادفی در اتصال) و تعداد بایت‌ها در بسته (اضافه کردن بایت‌های تصادفی به بسته) می‌توانند الگوی ارتباطی Botها را تصادفی کنند. البته این کارهای تصادفی باعث ایجاد سوء ظن می‌شود چرا که کاربران معمولی هم اینقدر رفتار تصادفی ندارند.
- گول زدن A-plane: Botnetها می‌توانند عملیات بدخواهانه را به صورت مخفیانه (اسکن کردن با نرخ خیلی پایین و فرستادن اسپم با نرخ خیلی پایین) انجام دهند که این باعث کاهش کارایی Botnet می‌شود. همچنین

[1] Covert Channel

Botmaster می‌تواند به جای دستورات گروهی، به صورت انفرادی به Botها دستور دهد که در این صورت دیگر Botها تفاوتی با بدافزارهای دیگر ندارند. Botmaster می‌تواند به Botهایی که در یک شبکه قرار دارند، دستورات مختلف بدهد که این هزینه اضافه‌ای برای Botmaster ایجاد می‌کند و می‌توان با استفاده از حسگرهای توزیع‌شده این مشکل را حل کرد.

- گول زدن Cross-plane: Botها می‌توانند دستورات دریافت شده را با تأخیر اجرا کنند. این کار باعث کم شدن کارایی Botnet می‌شود چرا که ممکن است در مدت تأخیر کامپیوترها از دسترس خارج شوند (خاموش شوند). ولی می‌توان با افزایش بازه زمانی این مشکل را حل کرد.

روش چهاردهم [۳۴]: Measurement and classification of humans and bots in internet chat

خصوصیات: مبتنی بر شبکه، مبتنی بر امضاء و ناهنجاری، منفعل، فاز شناسایی: عملیاتی، شناسایی گروهی از Botها، وابسته به ساختارهای متمرکز دستور و کنترل، داده مورد نیاز: کل بسته.

شرح: این روش مبتنی بر یادگیری ماشین است. کاربردهای Chatbot شامل ارسال لینک‌های اسپم به کاربران دیگر، ارسال پیامی حاوی لینک اسپم به اتاق چت، گذاشتن لینک اسپم در پروفایل Bot و ترغیب دیگران برای رجوع به پروفایل و کلیک بر روی لینک و پخش بدافزار است. انواع Chatbotها عبارتند از Botهای تناوبی[۱]، Botهای تصادفی[۲]، Botهای پاسخ‌دهنده[۳] و Botهای بازپخش[۴].

نشست‌های چت[۵] نرمال اصولاً طولانی مدت هستند. زمان بین نشست‌های چت از توزیع نمایی پیروی می‌کند ولی زمان بین پیام‌ها از توزیع نمایی پیروی نمی‌کند. اندازه پیام‌ها کوچک است. تعداد پیام‌های دریافتی کاربر به مراتب بیشتر از تعداد پیام‌های ارسالی کاربر است.

این روش نشان می‌دهد رفتار انسان از رفتار Bot پیچیده‌تر است و می‌توان برای اساس زمان، اندازه و محتوای پیام تفاوت انسان و Bot را تشخیص داد. به طور کلی دو دیدگاه برای شناسایی Chatbotها وجود دارد: شناسایی بر اساس کلمه[۶] و شناسایی از طریق ارتباط فعال[۱]. در روش شناسایی بر اساس کلمه نرخ عدم شناسایی نادرست زیاد است و

[1] Periodic Bot
[2] Random Bot
[3] Responder Bot
[4] Replay Bot
[5] Chat Session
[6] Keyword-based

شناسایی Botهای ناشناخته به دلیل تعویض کلمات ناممکن است. یکی از سیستم‌های موجـود در شناسـایی از طریـق ارتباط فعال، CAPTCHA می‌باشد.

اما Chatbotها برای فرار از شناسایی شدن، با ایجاد کاراکتر و فاصله های تصادفی در پیام‌ها، اسـتفاده از واژه‌هـای هم معنا به جای هم، شکستن پیام‌های بزرگ به چند پیام کوچک (با این روش، متدهایی که روی تک تک پیام‌هـا کـار می‌کنند، دچار مشکل می‌شوند.) و باز ارسال پیام‌های دریافت شده از دیگر کاربران سعی می‌کنند متن را مبهم کنند.

روش پـانزدهم [۳۵]: Active Botnet Probing to Identify Obscure Command and Control Channels

خصوصیات: مبتنی بر شبکه، مبتنی بر امضاء و ناهنجاری، فعال، فاز شناسایی: عملیاتی، شناسایی یک Bot مجـزا، وابسته به ساختارهای متمرکز دستور و کنترل، داده مورد نیاز: کل بسته.

شرح: این روش دیالوگ‌های مربوط به دستور و کنترل و انسان با انسان را از هـم جـدا کنـد. Botmasterهـا بـه دلایل زیر از پروتکل‌های شناخته شده (وب و چت) برای کانال‌های دستور و کنترل استفاده می‌کنند:

۱. پروتکل‌های موجود تست شده و انعطاف‌پذیر هستند و می‌توان از نرم‌افزارهای موجود بهره برد.
۲. پروتکل‌های موجود سوء ظن کمتری را نسبت به پروتکل‌های جدید به خود جلب می‌کنند.
۳. پروتکل‌های جدید نیازهای Botnetها را برطرف کرده و احتیاجی به طراحی پروتکل جدید نمی‌باشد.

دو اصل اساسی که باید در مورد Botها دانست به شرح زیرند:

۱. اصولاً Botها بدون حالت[۲] هستند و در نتیجه رفتاری قابل پیشبینی دارند و در حالی که رفتـار انسـان‌هـا غیـر قابل پیشبینی است.
۲. Botها برای انجام دستوراتی مشخص ساخته شده‌انـد و اگـر دسـتوری دارای خطـای املائـی باشـد، تشـخیص نمی‌دهند، در حالی که انسان‌ها به راحتی تشخیص می‌دهند.

با توجه به اصول فوق، نتیجه می‌گیریم که ارتباطات Botها با سرورهای دستور و کنترل دارای یک الگوی دسـتور-پاسخ[۳] می‌باشد و این روش با استفاده از این اصل، سعی در شناسایی Botها دارد.

---

[1] Human Interactive Proof
[2] Stateless
[3] Command-Response

فصل سوم: BotGrabber

Botnetها همواره در حال تکامل یافتن بوده و فوق‌العاده انعطاف‌پذیر هستند. ما مشاهده کرده‌ایم که کانال‌های دستور و کنترل از پروتکل‌های دیگری مانند وب به جای چت و ساختارهای نامتمرکز را به جای ساختارهای متمرکز استفاده می‌کنند. علاوه بر این، Botnet می‌تواند با استفاده از روش‌های Fast-Flux [۲۶,۲۵,۲۴] به طور پی در پی آدرس سرورهای دستور و کنترل را تغییر دهد.

در این فصل، ما یک سیستم جدید شناسایی Botnet به نام BotGrabber معرفی می‌کنیم که هدف آن شناسایی گروهی از کامپیوترهای آلوده به Bot متعلق به یک Botnet در شبکه در حال مانیتور است. خصوصیات این سیستم عبارتند از: مبتنی بر شبکه، مبتنی بر ناهنجاری، منفعل، فاز شناسایی: عملیاتی، شناسایی گروهی از Botها، عدم وابستگی به ساختارها و پروتکل‌های دستور و کنترل، داده مورد نیاز: کل بسته.

برای طراحی یک سیستم شناسایی که بتواند در مقابل تغییرات روش‌های دستور و کنترل مقاوم باشد، ما باید ویژگی‌های ذاتی فعالیت‌ها و ارتباطات Botnetها را مطالعه کنیم. بنابراین دوباره تعریف Botnet را ارائه می‌دهیم: گروهی هماهنگ از بدافزار(Bot)ها است که از طریق یک کانال دستور و کنترل تحت کنترل Botmaster می‌باشد. واژه بدافزار بدین معنی است که این Botها برای فعالیت‌های بدخواهانه استفاده می‌شوند. به عنوان مثال، مطابق [۳۶]، در حدود ۵۳٪ از فعالیت‌های Botnet مشاهده شده مربوط به اسکن کردن هستند (به قصد تکثیر شدن یا حملات DDoS) و در حدود ۱۴/۴٪ فعالیت‌ها مربوط به دانلود باینری می‌باشند [۳۸,۳۷] (برای به روز رسانی برنامه Bot). علاوه بر این، اکثر Botnetهای مبتنی بر وب و نظیر به نظیر برای ارسال اسپم استفاده می‌شوند [۹]. واژه تحت کنترل بدین معنی است که این Botها باید با سرورهای دستور و کنترل ارتباط برقرار کنند تا فرمان بگیرند و اجرا کنند. به عبارت دیگر، باید بین Botها و سرورهای دستور و کنترل تبادل اطلاعات صورت بگیرد. واژه گروه هماهنگ بدین معنی است که چند (حداقل دو) Bot در یک Botnet ارتباطات و فعالیت‌های شبیه و هماهنگ دارند.

اگر Botmaster به هر Bot به طور جداگانه دستور بدهد، Botها چیزی نیستند جز بدافزارهای جدا افتاده و نامرتبط. بنابراین این بدافزارها با توجه به تعریف ارائه شده به عنوان Botnet شناخته نمی‌شوند و در حوزه کار ما نمی‌باشند.

Botnetها جدای از ساختار و پروتکل کانال دستور و کنترل آن‌ها دارای رفتارهای ثابت و پایداری هستند. اول، Botها نیاز دارند تا به کانال دستور و کنترل وصل شوند تا بتوانند دستورات را دریافت کنند. آن‌ها ممکن است یک اتصال بلند مدت داشته باشند یا اینکه پی در پی اتصالات جدید برقرار کنند و اتصالات قبلی را قطع کنند. دوم، Botها باید بعد از دریافت دستورات، به آن‌ها واکنش نشان دهند. به طور کلی این واکنش‌ها را می‌توان به دو دسته کلی

تقسیم کرد:

۱. واکنش‌های دستور و کنترل: این نوع واکنش‌ها بسته‌هایی هستند که درون اتصال بین Bot و سرور دستور و کنترل، به سرور دستور و کنترل فرستاده می‌شوند. به عنوان مثال، Botmaster با ارسال دستوری خواستار گرفتن ورژن Bot یا اطلاعات سیستمی که Bot روی آن نصب است، باشد و در جواب Bot باید اطلاعات خواسته شده را از طریق کانال دستور و کنترل به Botmaster برساند.

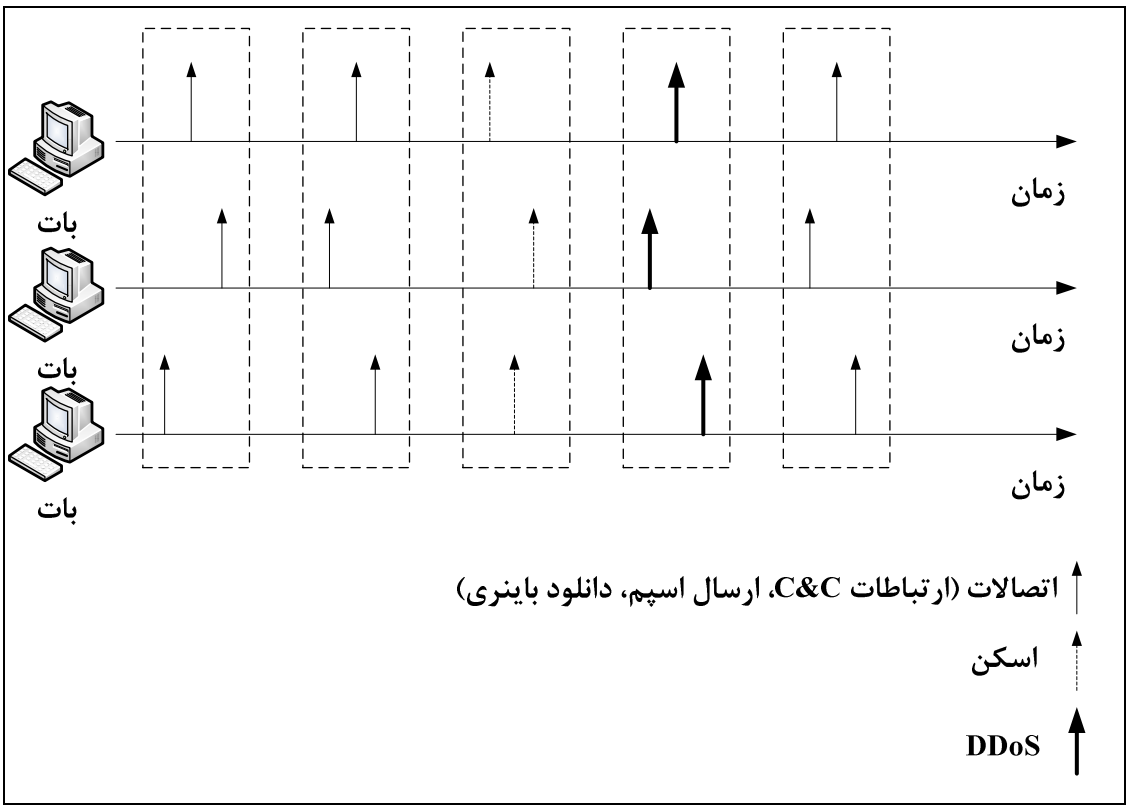
۲. واکنش‌های فعالیتی: این نوع واکنش‌ها، فعالیت‌های خرابکارانه مانند اسکن کردن، حملات DDoS، ارسال اسپم و دانلود باینری هستند. در این بین دو فعالیت اسکن کردن و حملات DDoS را می‌توان با روش‌های مبتنی بر ناهنجاری شناسایی کرد. فعالیت‌های ارسال اسپم و دانلود باینری باید با روش‌های مبتنی بر امضاء پیدا کرد. از آنجایی که هدف ما ارائه روشی کاملاً مبتنی بر ناهنجاری است، این سه فعالیت را از طریق شباهت‌های بین اتصالات ایجاد شده توسط Botها و ایجاد همبستگی بین آن‌ها شناسایی می‌کنیم که از دقت بیشتری برخوردار است.

اگر تعدادی Bot متعلق به یک Botnet در یک شبکه موجود باشند و به دستورات ارسالی از Botmaster واکنش نشان دهند، اکثر آن‌ها به شکل مشابه واکنش نشان می‌دهند. به عنوان مثال، Botها در زمانی یکسان، پیام‌های مشابه ارسال می‌کنند یا فعالیت‌های مشابه انجام می‌دهند. بنابراین ما شاهد یک حجمی از واکنش‌ها هستیم که توسط Botهای متعلق به یک Botnet صورت می‌گیرد. این چنین واکنش‌های گروهی هماهنگ در مورد تمام Botnetها مستقل از ساختار و پروتکل دستور و کنترل آن‌ها صادق است. از طرفی دیگر، برای سرویس‌های نرمالی مثل وب و چت، خیلی بعید است که تعدادی کامپیوتر به صورت متوالی، واکنش‌های مشابهی داشته باشند. بنابراین، Botها دارای هماهنگی و همزمانی پایدارتری نسبت به کاربران عادی هستند.

بر اساس مشاهدات فوق، رویکرد شناسایی Botnet در این پایان‌نامه، روی تشخیص واکنش‌های مشابه و همزمان Botها معطوف است. شکل ۳-۱ شباهت در واکنش‌های Botها را نمایش می‌دهد.

همانطور که در شکل ۳-۱ مشاهده می‌کنید، Botها به صورت متوالی دارای واکنش‌های مشابهی هستند. البته لزوماً این واکنش‌ها در یک زمان توسط همه Botها صورت نمی‌گیرد بلکه این واکنش‌ها با یک اختلاف زمانی صورت می‌گیرد. به همین خاطر ما یک پنجره زمانی در نظر می‌گیریم و تمام واکنش‌های موجود در آن پنجره زمانی را همزمان فرض می‌کنیم. هر چه طول پنجره زمانی بیشتر باشد، سیستم دارای دقت بیشتر بوده ولی از بلادرنگ بودن

سیستم کاسته می‌شود. بالعکس، هر چه طول پنجره زمانی کمتر باشد، دقت سیستم کاهش پیدا کرده و در عوض سیستم بلادرنگ‌تر خواهد بود. برای تعیین طول پنجره زمانی ما باید یک تعادل بین دقت و بلادرنگ بودن سیستم برقرار کنیم.



شکل ۳-۱) شباهت در واکنش‌های Botها

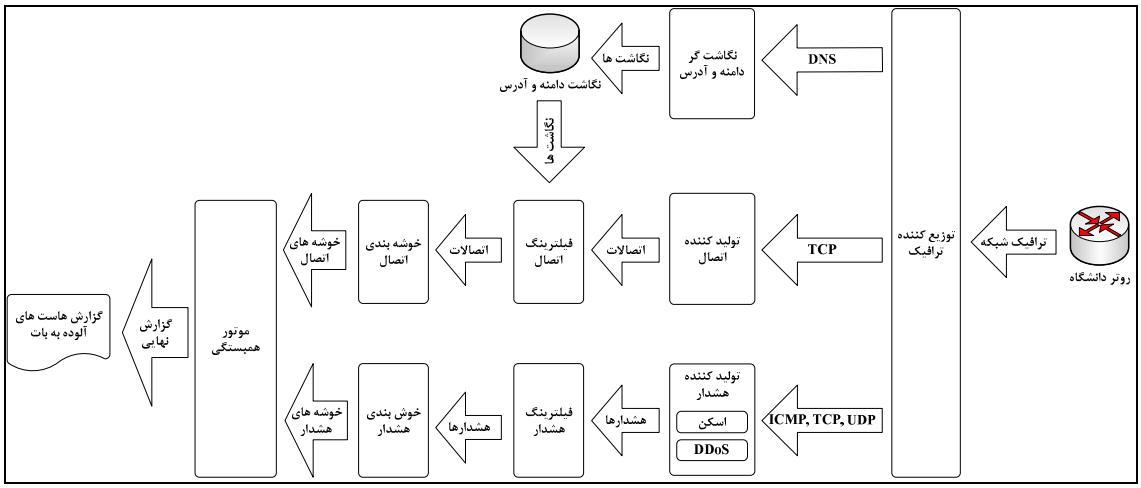
به طور خلاصه، هنگامی که چندین واکنش مشابه در پنجره‌های زمانی متوالی مشاهده کنیم، به کامپیوترهایی که دارای این واکنش مشابه بوده‌اند، یک مقدار امتیاز می‌دهیم و هنگامی که امتیاز هر کامپیوتر از یک آستانه بیشتر شود، آن کامپیوتر به عنوان Bot معرفی می‌شود. در ادامه معماری و مؤلفه‌های سیستم BotGrabber را با جزئیات بیشتری توضیح می‌دهیم.

۳-۱- معماری

شکل ۳-۲ معماری سیستم BotGrabber را نشان می‌دهد. سیستم BotGrabber دارای مؤلفه‌های مختلفی است که شامل توزیع‌کننده ترافیک[1]، نگاشت‌گر دامنه و آدرس[2]، تولیدکننده اتصال[3]، تولیدکننده هشدار[1]، فیلتر

[1] Traffic Dispatcher
[2] IP-DomainName Mapper
[3] Netflow Generator

اتصال[۲]، فیلتر هشدار[۳]، خوشه‌بندی اتصال[۴]، خوشه‌بندی هشدار[۵] و موتور همبستگی[۶] می‌باشد.



شکل ۳-۲) معماری BotGrabber

مؤلفه توزیع‌کننده ترافیک، ترافیک بین کامپیوترهای داخل و خارج شبکه را از روتر گرفته و ترافیک‌های مربوطه را به مؤلفه‌های نگاشت‌گر دامنه و آدرس، تولیدکننده اتصال و تولیدکننده هشدار می‌دهد. مؤلفه نگاشت‌گر دامنه و آدرس با مانیتور کردن ترافیک DNS، IPهای مربوط به دامنه‌هایی که مورد پرس و جو قرار گرفته‌اند را ذخیره می‌کند. تولیدکننده اتصال وظیفه تولید و ذخیره اتصالات شبکه برای تحلیل های آتی را دارد و تولیدکننده هشدار وظیفه شناسایی فعالیت‌های مشکوک و بدخواهانه (اسکن کردن و DDoS) را دارد. این سه مؤلفه به صورت موازی در حال اجرا هستند. مولفه های فیلتر اتصال و فیلتر هشدار وظیفه دارند اتصالات و هشدارهایی را که به پردازش نیازی ندارند، حذف کنند. خوشه‌بندی اتصال و خوشه‌بندی هشدار اطلاعات تولید شده توسط تولیدکننده اتصال و تولیدکننده هشدار را پردازش می‌کنند. هر دو مؤلفه با توجه به خصوصیات تعیین شده، از الگوریتم‌های خوشه‌بندی برای پیدا کردن گروهی از کامپیوترها که ارتباطات یا فعالیت‌های شبیه به هم دارند، استفاده می‌کنند. در نهایت موتور همبستگی با ایجاد همبستگی بین خوشه‌های تولید شده توسط مؤلفه‌های خوشه‌بندی اتصال و خوشه‌بندی هشدار، کامپیوترهای متعلق به Botnet را پیدا می‌کند.

در سیستم BotGrabber به دلیل لزوم کارایی بالا برای پردازش ترافیک بلادرنگ، تمام مؤلفه‌های ذکر شده با

---

[1] Alert Generator
[2] Netflow Filtering
[3] Alert Filtering
[4] Netflow Clustering
[5] Alert Clustering
[6] Correlation Engine

استفاده از زبان C++ پیاده‌سازی شده‌اند و ارتباط بین تمام مؤلفه‌ها به صورت برخط[1] می‌باشند. در بخش‌های آتی جزئیات طراحی و پیاده‌سازی هر یک از مؤلفه‌های ذکر شده ارائه خواهد شد.

۳-۱-۱- توزیع‌کننده ترافیک

از آنجایی که مؤلفه‌های نگاشت‌گر دامنه و آدرس، تولیدکننده اتصال و تولیدکننده هشدار ترافیک‌های مختلفی لازم دارند، این مؤلفه طراحی شده است. مؤلفه توزیع‌کننده ترافیک، ترافیک DNS (ترافیک UDP با پورت ۵۳) را به مؤلفه نگاشت‌گر دامنه و آدرس، ترافیک TCP را به مؤلفه تولیدکننده اتصال و ترافیک TCP، UDP و ICMP را به مؤلفه تولیدکننده هشدار می‌دهد. در این مؤلفه ترافیک‌های نامرتبط مانند ARP فیلتر می‌شوند تا کارایی سیستم افزایش یابد.

۳-۱-۲- نگاشت‌گر دامنه و آدرس

همانطور که قبلاً ذکر شد، Botها برای اتصال به سرور دستور و کنترل می‌توانند از IP یا آدرس DNSای که در فایل آن‌ها هاردکد[2] شده، استفاده کنند. از طرفی کامپیوترهای سالم هم برای وصل شدن به سایت‌های مورد نظرشان از DNS برای پیدا کردن IP سایت مربوطه استفاده می‌کنند. برای نگاشت برقرار کردن بین دامنه‌ها و IPها این مؤلفه طراحی شده است. این مؤلفه ابتدا ترافیک مربوط به DNS (ترافیک UDP با پورت ۵۳) را دریافت می‌کند. سپس با بررسی پاسخ‌های DNS، رکوردهای A را استخراج کرده و دامنه‌های پرس و جو شده و IPهای متناظر با آن‌ها را در یک پایگاه‌داده ذخیر می‌کند. این مؤلفه به طور موازی با مؤلفه‌های دیگر در حال اجرا می‌باشد.

۳-۱-۳- تولیدکننده اتصال

این مؤلفه ترافیک TCP را دریافت می‌کند و اتصالات بین کامپیوترها را بازسازی می‌کند. اکثر روترها مانند Cisco[3] و Juniper[4] توانایی تولید اتصالات را دارا هستند. ابزارهای متن باز[5] مانند Argus[6] نیز وجود دارند. از آنجایی که ابزارهای ذکر شده نیازمندی‌های ما را پاسخگو نیستند، ما ابزاری برای تولید اتصالات پیاده‌سازی کردیم. در حال حاضر، ما فقط اتصالات مربوط به TCP را مدنظر داریم. هر اتصال دارای اطلاعات ذیل می‌باشد: زمان شروع، زمان پایان، IP مبدأ، پورت مبدأ، IP مقصد، پورت مقصد، تعداد بسته‌های ارسالی/دریافتی و تعداد بایت‌های ارسالی/دریافتی. مهمترین برتری ابزار ما کارا بودن آن است و می‌تواند اطلاعات اتصالات را بهینه ذخیره کند. همچنین از آنجایی که

---

[1] Online
[2] Hard Code
[3] www.cisco.com
[4] www.juniper.net
[5] Open Source
[6] Audit Record Generation and Utilization System

ممکن است طول بعضی از اتصالات از طول یک پنجره زمانی بیشتر باشد، ابزار ما قادر است که اطلاعات اتصالات را در پنجره‌های زمانی مختلف بدهد. به این اتصالات اصطلاحاً اتصال نیمه‌کاره[1] گفته می‌شود.

۳-۱-۴- تولیدکننده هشدار

این مؤلفه ترافیک ICMP، TCP و UDP را دریافت می‌کند و فعالیت‌های بدخواهانه کامپیوترها را گزارش می‌دهد. این مؤلفه قادر است تا فعالیت‌های اسکن کردن (برای مقاصد تکثیر شدن) و حملات DDoS را شناسایی کند. اسکن کردن و DDoS، جزء مهمترین فعالیت‌های بدخواهانه‌ای هستند که توسط Botها صورت می‌گیرد [۳۶,۳۹,۱۳].

مؤلفه تولیدکننده هشدارها بر روی Snort که یک ابزار شناسایی نفوذ متن باز است، نوشته شده است. ما روش‌های شناسایی فعالیت‌های بدخواهانه را به عنوان پیش‌پردازنده‌هایی[2] برای Snort پیاده‌سازی کرده‌ایم. برای شناسایی اسکن کردن ما از پیش‌پردازنده موجود در Snort به نام sfPortscan استفاده کرده‌ایم. امروزه یکی از مهمترین ابزارهای اسکن کردن، Nmap است. در حال حاضر، این پیش‌پردازنده برای انواع اسکن‌های زیر هشدار تولید می‌کند:

- پورت اسکن یک به یک
- پورت اسکن مخفیانه[3]
- پورت اسکن توزیع‌شده[4]
- پورت اسکن‌های جاروبی[5]

پورت اسکن‌های یک به یک از اسکن‌های معروف می‌باشند. یک کامپیوتر چندین پورت کامپیوتر دیگر را اسکن می‌کند. بیشتر درخواست‌های ارسالی ناموفق خواهند بود چرا که اکثر کامپیوترها سرویس‌های کمی در حال اجرا دارند. پورت اسکن‌های مخفیانه مانند پورت اسکن‌های یک به یک هستند با این تفاوت که کامپیوترهای اسکن کننده علاوه بر بسته‌هایی که با آدرس خود ارسال می‌کنند، بسته‌های دیگری با آدرس‌های قلابی[6] تولید می‌کنند تا هویتشان آشکار نشود. پورت اسکن‌های توزیع‌شده، چند به یک هستند. بدین ترتیب که چند کامپیوتر پورت‌های یک کامپیوتر را اسکن

[1] Partial Netflow
[2] Preprocessor
[3] Decoy Portscan
[4] Distributed Portscan
[5] Port Sweep
[6] Spoofed

می‌کنند. با این روش تعداد اتصالات ناموفق بین کامپیوترهای اسکن کننده تقسیم می‌شود. پورت اسکن‌های جاروبی یک به چند هستند. بدین صورت که یک کامپیوتر یک پورت خاص را روی چند کامپیوتر اسکن می‌کند. این نوع اسکن‌ها برای پیدا کردن یک آسیب‌پذیری صورت می‌گیرد.

برای شناسایی فعالیت‌های مربوط به حملات DDoS، ما یک پیش‌پردازنده روی Snort پیاده‌سازی کرده‌ایم. این پیش‌پردازنده دارای سه ماژول است که قادر به شناسایی سه نوع حمله زیر می‌باشد:

- TCP SYN Flood
- UDP Flood
- ICMP Ping Flood

در حمله TCP SYN Flood تعداد زیادی بسته TCP SYN به پورت مورد نظر فرستاده می‌شود بدون اینکه اتصال به طور کامل برقرار شود. در این حالت حافظه کامپیوتر مورد حمله قرار گرفته پر شده و درخواست‌هایی که توسط کاربران واقعی سیستم فرستاده می‌شود بدون جواب گذاشته می‌شود و عملاً سرویس مورد نظر دچار اختلال می‌شود. در حمله UDP Flood، کامپیوتر مهاجم تعداد زیادی بسته UDP با حجم بالا به یک پورت خاص یک کامپیوتر ارسال می‌کند تا پهنای باند، حافظه و پردازنده کامپیوتر مورد نظر را مشغول کند و کامپیوتر مورد نظر نتواند به درخواست‌های کاربران معمولی پاسخ دهد. در حمله ICMP Ping Flood، کامپیوتر مهاجم تعداد زیادی بسته ICMP Ping با حجم بالا به یک کامپیوتر ارسال می‌کند تا پهنای باند، حافظه و پردازنده کامپیوتر مورد نظر را مشغول کند و کامپیوتر مورد نظر نتواند به درخواست‌های کاربران معمولی پاسخ دهد.

برای استفاده از این پیش‌پردازنده، کاربر یک پنجره زمانی و یک آستانه برای هر ماژول مشخص می‌کند. در طی هر پنجره زمانی هر ماژول جدول خود را تشکیل می‌دهد. جدول TCP SYN Flood دارای ستون‌های IP مبدأ، IP مقصد، پورت مقصد، تعداد بسته‌های TCP SYN که منجر به برقراری کامل اتصال نشده‌اند و انحراف از معیار زمان بین بسته‌های فرستاده شده، تشکیل شده است. جدول UDP Flood دارای ستون‌های IP مبدأ، IP مقصد، پورت مقصد، تعداد بسته‌های UDP فرستاده شده، متوسط تعداد بایت در هر بسته و انحراف از معیار زمان بین بسته‌های فرستاده شده، تشکیل شده است. جدول ICMP Ping Flood دارای ستون‌های IP مبدأ، IP مقصد، تعداد بسته‌های ICMP Ping فرستاده شده، متوسط تعداد بایت در هر بسته و انحراف از معیار زمان بین بسته‌های فرستاده شده، تشکیل شده است.

هر ماژول در انتهای پنجره زمانی تعریف شده، امتیازی برای هر یک از سطرهای جدولش حساب می‌کند و در

صورتی که امتیاز سطر مورد نظر از آستانه تعریف شده بیشتر باشد، IP مبدأ آن سطر به عنوان کامپیوتر مهاجم معرفی می‌شود. جدول ۳-۱ نمونه‌ای از جدول ساخته شده برای ماژول‌های فوق را نشان می‌دهد.

جدول ۳-۱) جدول اطلاعات DDoS

Proto	Src. IP	Dest. IP	Dest. Port	Pkts	Avg. BPP	Pkts Interval STD
TCP	192.168.20.1	194.225.100.4	80	70	-	0.006
…	…	…	…	…	-	…
UDP	192.168.20.2	194.225.100.5	53	500	400	0.04
…	…	…	…	…	…	…
ICMP	192.168.20.3	194.225.100.6	-	10	50	40.24
…	…	…	-	…	…	…

در حال حاضر ماژول تشخیص حملات DDoS به دلیل ناقص بودن در مؤلفه تولید کننده هشدار مورد استفاده قرار نگرفته است. از آنجایی که قصد داریم از روش‌های مبتنی بر ناهنجاری، فعالیت‌های بدخواهانه را شناسایی کنیم، فقط دو فعالیت اسکن کردن و حملات DDoS در این مؤلفه شناسایی می‌شوند. فعالیت‌های بدخواهانه دیگری نیز مانند ارسال اسپم و دانلود باینری Bot وجود دارند. در سیستم‌های قبلی، این فعالیت‌ها با روش‌های مبتنی بر امضاء تشخیص داده می‌شدند. از آنجایی که نمی‌توان با روش‌های مبتنی بر امضاء به طور کامل و دقیق این فعالیت‌ها را کشف کرد، این نوع فعالیت‌ها در مؤلفه موتور همبستگی شناسایی می‌شوند.

۳-۱-۵- فیلتر اتصال

از آنجایی که در شبکه‌های پر سرعت میزان ترافیک قابل پردازش خیلی زیاد است، در نتیجه تعداد اتصالات تولید شده توسط مؤلفه تولیدکننده اتصال بسیار زیاد است و پردازش این اتصالات توسط مؤلفه خوشه‌بندی اتصال بسیار زمانبر خواهد بود. به همین خاطر، برای افزایش کارایی باید تعدادی از اتصالات که احتیاجی به پردازش آن‌ها نیست را حذف کرد. مؤلفه فیلتر اتصال از قواعد زیر برای فیلتر کردن اتصالات استفاده می‌کند:

۱. اتصالاتی که از بیرون شبکه به داخل شبکه یا از داخل شبکه به داخل شبکه برقرار شده‌اند.
۲. اتصالاتی که داده ای ارسال نکرده‌اند. این اتصالات عموماً از اسکن کردن حاصل می‌شوند و فقط حاوی بسته‌های کنترلی هستند.
۳. اتصالاتی که میزان داده ارسالی آن‌ها از یک آستانه (BULKY_THRESHOD) بیشتر باشد. این اتصالات اصولاً مربوط به دانلود فایل‌های بزرگ هستند. بزرگترین اتصالات Botها شامل اتصالات دانلود فایل باینری Bot یا اتصالات ارسال اسپم که حاوی فایل باینری Bot هستند، می‌شوند که حداکثر به چند صد کیلو بایت می‌رسند.

۴. اتصالاتی که مقصد آن‌ها در لیست سفید قرار داشته باشند. لیست سفید شامل سرورهای شناخته شده (گوگل، یاهو و ...) است که به سختی بتوانند به عنوان سرورهای دستور و کنترل مورد استفاده قرار گیرند. این کار با استفاده از پایگاه‌داده‌ای که توسط مؤلفه نگاشت‌گر دامنه و آدرس تهیه شده، صورت می‌گیرد.

باید گفت که این فیلترینگ هیچ گونه تأثیری در دقت شناسایی سیستم نداشته و فقط به دلیل افزایش کارایی صورت می‌گیرد.

۳-۱-۶- فیلتر هشدار

این مؤلفه وظیفه حذف هشدارهایی را دارد که منشأ آن کامپیوترهای بیرونی هستند. این کار برای کاهش هشدارهای تولید شده و افزایش کارایی مؤلفه خوشه‌بندی هشدارها صورت می‌گیرد. باید در نظر داشت که این فیلترینگ هیچ گونه تأثیری در دقت شناسایی سیستم نداشته و فقط به دلیل افزایش کارایی صورت می‌گیرد.

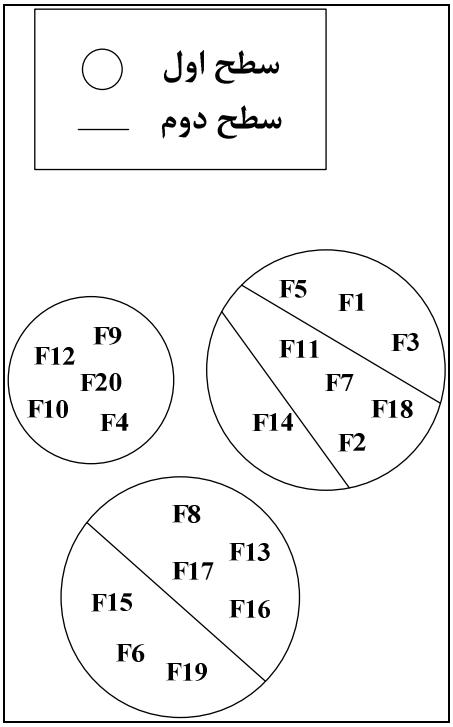
۳-۱-۷- خوشه‌بندی اتصال

هدف مؤلفه خوشه‌بندی اتصال این است که اتصالاتی که در یک پنجره زمانی شبیه به هم هستند را گروه‌بندی کند. الگوریتم‌های خوشه‌بندی یادگیری بدون نظارت[1] انجام می‌دهند. به طور کلی هدف الگوریتم خوش‌بندی، گروه‌بندی داده‌ها در قالب تعدادی خوشه است که در هر خوشه داده‌ها کمترین فاصله و از دیگر خوشه‌ها بیشترین فاصله را داشته باشد [۴۰]. از آنجایی که تعداد اتصالات تولید شده زیاد است، برای افزایش کارایی عملیات خوشه‌بندی در دو مرحله انجام می‌گیرد: خوشه‌بندی در سطح ویژگی‌های اتصال و خوشه‌بندی در سطح محتوای اتصال.

در خوشه‌بندی سطح اول لیستی از ویژگی‌ها[2] تهیه می‌کنیم. سپس برای هر اتصال ویژگی‌های آن را استخراج کرده و اتصالات را در خوشه‌های مختلف قرار می‌دهیم. از آنجایی که ممکن است اتصالات نامربوط با هم در یک خوشه قرار بگیرند، برای افزایش دقت، خوشه‌بندی سطح دوم را انجام می‌دهیم. این خوشه‌بندی بر روی اتصالات خوشه‌های تولید شده از مرحله قبل به طور جداگانه صورت می‌گیرد. در نتیجه خوشه‌های سطح اول به خوشه‌های جدیدتری شکسته می‌شوند. شکل ۳-۳ نمونه‌ای از خوشه‌بندی دو سطحی را نشان می‌دهد.

---

[1] Unsupervised Learning
[2] Feature



شکل ۳-۳) خوشه‌بندی دو سطحی اتصالات

ویژگی‌های استخراج شده، برای هر اتصال یک بردار تشکیل می‌دهند تا با استفاده از آن بتوان اتصالات را خوشه‌بندی کرد. این ویژگی‌ها عبارتند از:

۱. تعداد بسته‌های ارسالی: این ویژگی تعداد بسته‌های ارسالی از سمت مبدأ به مقصد می‌باشد.

۲. تعداد بایت‌های ارسالی: این ویژگی تعداد بایت‌های ارسالی از سمت مبدأ به مقصد می‌باشد.

۳. متوسط تعداد بایت در بسته‌های ارسالی: این ویژگی تقسیم تعداد بایت‌های ارسالی به تعداد بسته‌های ارسالی از سمت مبدأ به مقصد می‌باشد.

۴. متوسط تعداد بایت‌های ارسالی در ثانیه: این ویژگی تقسیم تعداد بایت‌های ارسالی به طول اتصال از سمت مبدأ به مقصد می‌باشد.

۵. تعداد بسته‌های دریافتی: این ویژگی تعداد بسته‌های ارسالی از سمت مقصد به مبدأ می‌باشد.

۶. تعداد بایت‌های دریافتی: این ویژگی تعداد بایت‌های ارسالی از سمت مقصد به مبدأ می‌باشد.

۷. متوسط تعداد بایت در بسته‌های دریافتی: این ویژگی تقسیم تعداد بایت‌های ارسالی به تعداد بسته‌های ارسالی از سمت مقصد به مبدأ می‌باشد.

۸. متوسط تعداد بایت‌های دریافتی در ثانیه: این ویژگی تقسیم تعداد بایت‌های ارسالی به طول اتصال از سمت

مقصد به مبدأ می‌باشد.

از آنجایی که Botهای متعلق به یک Botnet دارای رفتار مشابه در ارتباطات خود هستند، هدف ما پیدا کردن گروه‌های اتصالات شبیه به هم می‌باشد. بنابراین بردارهای اتصالات دارای رفتارهای مشابه نیز به هم نزدیک هستند. به عنوان مثال، فرض کنید دو Bot که متعلق به یک Botnet هستند به دو سرور دستور و کنترل متفاوت وصل باشند (بعضی از Botnetها از چند سرور دستور و کنترل استفاده می‌کنند). اگر چه این دو اتصال کاملاً جدا از هم هستند و هیچگونه اشتراکی از جهت آدرس و پورت مبدأ و مقصد ندارند، ولی خصوصیات مربوط به ترافیک اتصالات آن‌ها باید شبیه به هم باشد. بنابراین برای یافتن گروه‌هایی از کامپیوترها که دارای ارتباطات مشابه هستند، ما از الگوریتم‌های خوشه‌بندی برای خوشه‌بندی بردارهای متناظر اتصالات استفاده می‌کنیم.

برای خوشه‌بندی سطح اول، از الگوریتم خوشه‌بندی X-means [۴۱] استفاده کرده‌ایم. X-means یک الگوریتم خوشه‌بندی کارا و معروف است که بر اساس K-means است. برخلاف K-means، الگوریتم X-means احتیاجی به تعیین تعداد خوشه‌ها (K) ندارد. الگوریتم X-means چند دور الگوریتم K-means را اجرا می‌کند و از Bayesian Information Criterion [۴۱] برای ارزیابی و پیدا کردن بهترین مقدار K استفاده می‌کند. الگوریتم X-means یک الگوریتم سریع و مقیاس‌پذیر است.

خوشه‌بندی سطح دوم روی محتوای بسته‌های اتصالات انجام می‌گیرد. هر اتصال دارای دو محتوا است:

۱. محتوای ارسالی: این محتوا از به هم پیوستن بسته‌های ارسالی از مبدأ به سمت مقصد به دست می‌آید.

۲. محتوای دریافتی: این محتوا از به هم پیوستن بسته‌های ارسالی از مقصد به سمت مبدأ به دست می‌آید.

از آنجایی که Botهای متعلق به یک Botnet دارای محتوای مشابه در اتصالات خود هستند، هدف ما پیدا کردن گروه‌های اتصالات شبیه به هم می‌باشد. به عنوان مثال، فرض کنید دو Bot که متعلق به یک Botnet هستند به دو سرور دستور و کنترل متفاوت وصل باشند. اگر چه این دو اتصال کاملاً جدا از هم هستند و هیچگونه اشتراکی از جهت آدرس و پورت مبدأ و مقصد ندارند، ولی محتوای اتصالات آن‌ها باید شبیه به هم باشد. بنابراین برای یافتن گروه‌هایی از کامپیوترها که دارای محتواهای مشابه هستند، ما از الگوریتم‌های خوشه‌بندی برای خوشه‌بندی اتصالات با محتواهای مشابه استفاده می‌کنیم.

برای خوشه‌بندی سطح دوم، از الگوریتم خوشه‌بندی سلسله مراتبی[1] استفاده کرده‌ایم. الگوریتم خوشه‌بندی سلسله مراتبی یکی از الگوریتم‌های معروف و کارا در حوزه تحلیل خوشه است که هدف آن تولید سلسله مراتبی از

[1] Hierarchical Clustering

خوشه‌کامپیوتر که به صورت درختی سازمان یافته‌اند. در خوشه‌بندی سلسله مراتبی هر اتصال به عنوان یک خوشه در نظر گرفته می‌شود و سپس با یکپارچه کردن خوشه‌ها، سلسله مراتبی از خوشه‌ها شکل می‌گیرد. برای خوشه‌بندی اتصالات، از شباهت بین محتواهای ارسالی و دریافتی اتصالات استفاده می‌کنیم. برای اندازه‌گیری این شباهت از NCD[1] [۴۳,۴۲] استفاده می‌کنیم. برای محاسبه میزان فاصله دو اتصال از رابطه (۳-۱) استفاده می‌شود. همچنین رابطه‌های (۳-۲) و (۳-۳) نحوه محاسبه وزن‌های مورد نظر را نشان می‌دهند. روابط یاد شده تقریبی هستند.

$$\text{Distance}(F_i,F_j)=w_{send}\times NCD\left(SendPay_i, SendPay_j\right)+w_{recv}\times NCD\left(RecvPay_i, RecvPay_j\right) \quad (۳-۱)$$

$$w_{send}=\frac{|SendPay_i|+|SendPay_j|}{|SendPay_i|+|SendPay_j|+|RecvPay_i|+|RecvPay_j|} \quad (۳-۲)$$

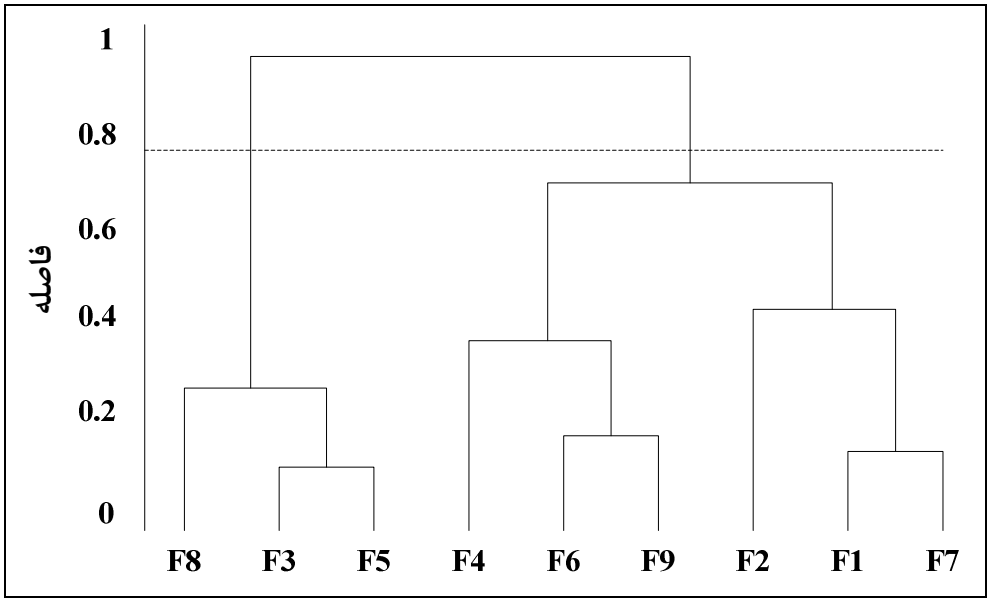
$$w_{recv}=\frac{|RecvPay_i|+|RecvPay_j|}{|SendPay_i|+|SendPay_j|+|RecvPay_i|+|RecvPay_j|} \quad (۳-۳)$$

در رابطه (۳-۱) ، $w_{send}+w_{recv}=1$ است و SendPay محتوای ارسالی اتصال مورد نظر و RecvPay محتوای دریافتی اتصال مورد نظر است. از آنجایی که طول محتوای ارسالی با محتوای دریافتی متفاوت است، از دو وزن w_{send} و w_{recv} برای افزایش دقت فاصله دو اتصال استفاده می‌کنیم. الگوریتم خوشه‌بندی سلسله مراتبی با استفاده از رابطه (۳-۱)، فاصله بین دو به دو اتصالات را محاسبه می‌کند و سپس سلسله مراتب اتصالات را تشکیل می‌دهد.

حال ما یک درختی از اتصالات داریم که باید خوشه‌هایی را از آن استخراج کنیم. تعداد خوشه‌های استخراج شده، بستگی به میزان حد برش[2] دارد. حد برش در حقیقت نشان‌دهنده میزان فاصله بین محتوای اتصالات است. هر چه مقدار حد برش کوچکتر باشد، تعداد خوشه‌های تولید شده بیشتر و دقت بالاتر است و هر چه مقدار حد برش بزرگتر باشد، تعداد خوشه‌های تولید شده کمتر و دقت پایین‌تر است. از آنجایی که محتوای اتصالات Botnet لزوماً به طور کامل شبیه به هم نیستند، اگر مقدار حد برش از یک حدی کوچکتر باشد، اشتباهاً اتصالات شبیه به هم در یک خوشه قرار نمی‌گیرند و باعث افزایش عدم شناسایی نادرست می‌شود.

شکل ۳-۴ نمونه‌ای از نمودار دندوگرام مربوط به خوشه‌بندی سلسله مراتبی است. خط چین در شکل ۳-۴ نشان دهنده مقدار حد برش است که در نهایت منجر به ایجاد دو خوشه {F8, F3, F5} و { F4, F6, F9, F2, F1, F7} شده است.

[1] Normalized Compression Distance
[2] Cut-off



شکل ۳-۴) نمودار دندروگرام

۳-۱-۸- خوشه‌بندی هشدار

در این مرحله، یک خوشه‌بندی چند سطحی روی هشدارها انجام می‌دهیم. برای تمام کامپیوترهایی که در طول پنجره زمانی مورد نظر حداقل یک فعالیت خرابکارانه انجام داده‌اند، در سطح اول خوشه‌بندی ابتدا آن‌ها را بر حسب نوع فعالیتشان (اسکن کردن، حملات DDoS و ...) خوشه‌بندی می‌کنیم. سپس در سطح‌های بعدی خوشه‌بندی، کامپیوترهای موجود در هر خوشه را با توجه به خصوصیات آن نوع فعالیت به خوشه‌های دیگر می شکنیم. برای فعالیت اسکن کردن، از خصوصیات زیر می‌توان برای خوشه‌بندی سطح دوم استفاده کرد:

۱. پورت‌های اسکن شده: بدین معنی که کامپیوترهایی که پورت‌های یکسانی را اسکن کرده‌اند، در یک خوشه قرار می‌گیرند.

۲. شبکه یا زیرشبکه اسکن شده: بدین معنی که کامپیوترهایی که کامپیوترهای یک شبکه یا زیر شبکه یکسانی را اسکن کرده‌اند، در یک خوشه قرار می‌گیرند.

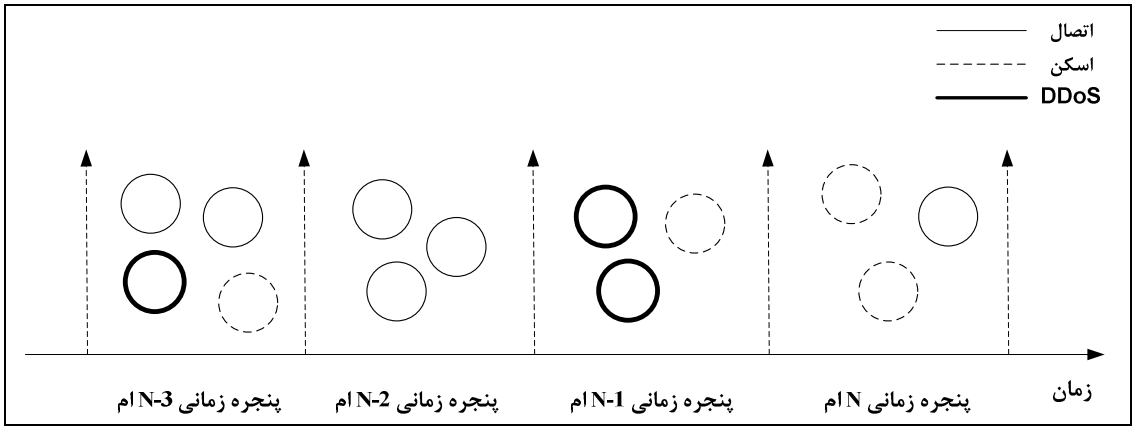
۳. نوع اسکن انجام شده: بدین معنی که کامپیوترهایی که یک نوع خاص اسکن را انجام داده‌اند (شماره هشداری که Snort تولید می‌کند)، در یک خوشه قرار می‌گیرند.

در سیستم پیاده‌سازی شده، فعالیت‌های اسکن کردن بر اساس نوع اسکن صورت گرفته خوشه‌بندی می‌شوند.

۳-۱-۹- موتور همبستگی

وقتی نتایج خوشه‌بندی از مراحل قبل مشخص شد، نوبت به ایجاد همبستگی بین خوشه‌ها می‌رسد. شکل ۳-۵

نمایی از خوشه‌های تولید شده در هر پنجره زمانی را نشان می‌دهد. همانطور که در شکل ۳-۵ مشاهده می‌کنید، مؤلفه‌های خوشه‌بندی اتصال و خوشه‌بندی هشدار در پایان هر پنجره زمانی، خوشه‌های مربوط به اتصالات و هشدارهای هر پنجره زمانی را درست می‌کنند و به مؤلفه موتور همبستگی می‌دهند. هر کدام از سه نوع خوشه ذکر شده شامل IP مبدأ اتصالات یا هشدارهای مورد نظر هستند. ایده اصلی ایجاد همبستگی این است که بین تمام خوشه‌های پنجره زمانی حال حاضر (N) و خوشه‌های پنجره‌های زمانی قبلی اشتراک‌گیری انجام دهیم و بر اساس میزان اشتراک به کامپیوترهای موجود در آن خوشه‌ها امتیاز اختصاص دهیم. برای انجام این کار، برای هر کامپیوتر یک امتیاز Score(h) محاسبه می‌شود و کامپیوترهایی که امتیازشان از یک آستانه θ بیشتر باشد، به عنوان Bot در نظر گرفته می‌شوند.



شکل ۳-۵) خوشه‌های تولید شده در پنجره‌های زمانی مختلف

حال نحوه امتیازدهی به کامپیوترها را توضیح می‌دهیم. فرض کنید H مجموعه تمام کامپیوترهایی هستند که تا به حال توسط سیستم ما مانیتور شده‌اند و $h \in H$ و N شماره پنجره زمانی جاری باشد. شکل ۳-۶ نحوه محاسبه امتیاز کسب شده برای کامپیوتر h در پنجره زمانی N ام را نشان می‌دهد. در شکل ۳-۶، Score(h, N) میزان امتیاز کسب شده برای کامپیوتر h در پنجره زمانی N ام است. نحوه امتیازدهی بدین صورت است که برای کامپیوتر h بین دو به دو خوشه‌های پنجره زمانی N ام با خوشه‌های پنجره زمانی N ام و خوشه‌های پنجره زمانی N ام با خوشه‌های پنجره‌های زمانی قبلی (حداکثر تا MAX_NUM_TW) همبستگی ایجاد می‌شود و بزرگترین امتیازی که توسط تابع TWCorrelation محاسبه می‌شود، به عنوان امتیاز کسب شده برای کامپیوتر h در نظر گرفته می‌شود. ولی اگر هیچگونه همبستگی ایجاد نشد، امتیازی منفی معادل فاصله پنجره زمانی N ام و آخرین پنجره زمانی‌ای که کامپیوتر h امتیازی کسب کرده ($lastCorrelatedTW_h$)، در نظر گرفته می‌شود.

```
function Score(h,N):
    maxScore=-1

    for i = 0 to MAX_NUM_TW:
        twScore = TWCorrelation(h, TW_N, TW_N-i)
        if twScore ≥ 0:
            if maxScore == -1 or twScore > maxScore:
                maxScore = twScore
            endif
        endif
    endfor

    if maxScore ≠ -1:
        return maxScore
    else:
        return -1 × (N - lastCorrelatedTW_h)
end function
```

شکل ۳-۶) امتیاز کسب شده برای کامپیوتر h در پنجره زمانی N ام

شکل ۳-۷ نحوه ایجاد همبستگی بین دو پنجره زمانی را نشان می‌دهد.

```
function TWCorrelation(h, TW_i, TW_j):
    clusterSet = {}
    twScore = 0
    for c_i in C^h_TW_i:
        for c_j in C^h_TW_j:
            if ClusterCorrelation(c_i , c_j, i-j) ≥ CORRELATION_THRESHOLD:
                clusterSet.insert(c_i )
                clusterSet.insert(c_j )
            endif
        endfor
    endfor

    for c in clusterSet:
        twScore += { 1,  Type(c) == Netflow
                   { 2,  Type(c) == {Scanning or DDoS}
    endfor

    return min(twScore, MAX_TW_SCORE)
end function
```

شکل ۳-۷) ایجاد همبستگی بین دو پنجره زمانی

در شکل ۳-۷، $C^{h}_{TW_i}$ مجموعه‌ای از خوشه‌های متعلق به پنجره زمانی i ام است که شامل کامپیوتر h نیز هستند.

همچنین $C^{h}_{TW_j}$ مجموعه‌ای از خوشه‌های متعلق به پنجره زمانی j ام است که شامل کامپیوتر h نیز هستند.

clusterSet مجموعه خوشه‌هایی هستند که شامل کامپیوتر h هستند و همچنین با هم همبسته شده‌اند و CORRELATION_THRESHOLD آستانه‌ای است که تعیین می‌کند میزان شباهت بین خوشه‌ها چقدر باشد تا آن‌ها همبسته شوند. Type(c) نوع خوشه مربوطه را مشخص می‌کند (اتصال، اسکن و DDoS). MAX_TW_SCORE حداکثر میزان امتیازی است که در یک پنجره زمانی به کامپیوتر h اختصاص داده می‌شود.

تابع ClusterCorrelation میزان شباهت بین دو خوشه را مشخص می‌کند. رابطه (۳-۴) نحوه محاسبه شباهت بین دو خوشه نشان داده شده است.

$$\text{ClusterCorrelation}\left(c_i, c_j, d\right) = 1 - e^{-\left(\frac{|c_i \cap c_j|}{|c_i \cup c_j|} \times |c_i \cap c_j| \times \frac{1}{d+1}\right)} \qquad (۳-۴)$$

در رابطه (۳-۴) سه عامل مهم در میزان امتیاز محاسبه شده نقش دارند: ۱) درصد اشتراک دو خوشه، ۲) تعداد اشتراک دو خوشه و ۳) فاصله دو پنجره زمانی‌ای که خوشه‌ها در آن‌ها قرار دارند. هر چه درصد اشتراک و تعداد اشتراک دو خوشه بالاتر و فاصله آن‌ها پایین‌تر باشد، مقدار امتیاز در نظر گرفته شده بیشتر خواهد بود. از آنجایی که مقدار محاسبه شده از ضرب این سه پارامتر دارای حد بالایی نیست، از فرمول $1 - e^{-x}$ برای محدود کردن این مقدار بین صفر و یک استفاده کرده‌ایم. البته رابطه (۳-۴) تقریبی بوده و برای افزایش کارایی چند تا از مقادیر از قبل محاسبه می‌شوند و در جدول مراجعه[1] قرار می‌گیرند.

برای امتیازدهی، هنگام ایجاد همبستگی بین پنجره‌های زمانی، خوشه‌هایی که با هم همبسته می‌شوند را در یک مجموعه نگهداری می‌کنیم. سپس بر حسب نوع خوشه‌های موجود در آن مجموعه امتیازدهی می‌کنیم. مقدار d فاصله بین دو پنجره زمانی‌ای است که می‌خواهیم خوشه‌های آن‌ها را همبسته کنیم. هر چه تعداد کامپیوترهای مشترک دو خوشه بیشتر باشد، میزان شباهت افزایش می‌یابد و هر چه مقدار d بزرگتر باشد، میزان شباهت کاهش می‌یابد.

کامپیوتر h امتیاز بالایی دریافت می‌کند اگر در پنجره‌های زمانی متوالی دارای واکنش باشد و تعداد کامپیوترهایی که همان واکنش‌ها را انجام داده‌اند، زیاد باشد. به عنوان مثال، فرض کنید کامپیوتر h در پنجره زمانی $N - 1$ ام فعالیت اسکن انجام داده است و سپس در پنجره زمانی N ام اقدام به حمله DDoS روی یک کامپیوتر که خارج از شبکه در حال مانیتور است، کرده باشد. فرض کنید C_1 خوشه کامپیوترهایی است که در پنجره زمانی $N - 1$ فعالیت اسکن انجام داده‌اند و کامپیوتر h هم در این خوشه قرار دارد. همچنین C_2 خوشه کامپیوترهایی است که در پنجره زمانی N حمله DDoS انجام داده‌اند و کامپیوتر h هم در این خوشه قرار دارد. اگر اشتراک وزن‌دار این دو خوشه از

[1] Lookup Table

یک آستانه بیشتر باشد، آنگاه چهار امتیاز به کامپیوتر h اضافه خواهد شد. در نتیجه اگر کامپیوتر h فعالیت‌های خرابکارانه انجام دهد، امتیاز بیشتری کسب می‌کند و سریعتر شناسایی می‌شود. در عین حال اگر کامپیوتر h هیچکدام از فعالیت‌های اسکن و DDoS را انجام ندهد، باز هم با توجه به شباهت‌هایی که بین اتصالاتش است، با سرعت کمتری شناسایی می‌شود. ولی اگر در پنجره‌های زمانی مختلف هیچگونه همبستگی‌ای برای کامپیوتر h ایجاد نشد، بسته به آخرین همبستگی قبلی، مقداری امتیاز منفی برای کامپیوتر h در نظر گرفته می‌شود. عملیات امتیازدهی در پایان هر پنجره زمانی صورت می‌گیرد و کامپیوترهایی که امتیازشان از یک آستانه (BOT_THRESHOLD) بیشتر باشد به عنوان Bot در نظر گرفته می‌شوند.

۳-۲- آزمایش‌ها

برای ارزیابی سیستم BotGrabber، ما از ترافیک‌های واقعی شبکه استفاده کرده‌ایم که شامل ترافیک نرمال جمع‌آوری شده از شبکه دانشگاه و ترافیک Botnetهای واقعی است. جدول ۳-۲ مقادیر پیش‌فرض هر یک از متغیرهای ذکر شده را تعیین می‌کند.

جدول ۳-۲) مقادیر پیش‌فرض متغیرها

متغیر	مقدار پیش‌فرض
طول پنجره زمانی	۲۰ دقیقه
MAX_NUM_TW	۳
حد برش خوشه‌بندی سطح دوم (CUT_OFF)	۰/۳۵
BULKY_THRESHOLD	۱ مگا بایت
MAX_TW_SCORE	۵
CORRELATION_THRESHOLD	۰/۶۵
BOT_THRESHOLD	۳۳

برای تعیین طول پنجره زمانی، Botnetهای مختلفی اعم از مبتنی بر چت و مبتنی بر وب را مورد مطالعه قرار دادیم و به این نتیجه رسیدیم که Botها در طول ۲۰ دقیقه حداقل یک واکنشی خواهند داشت و فاصله زمانی این واکنش‌ها حداکثر ۳ پنجره زمانی خواهد بود. برای تعیین حد برش خوشه‌بندی سطح دوم محتویات اتصالات

Botnetهای مختلفی را بررسی کردیم و مشاهده کردیم که حداکثر فاصله دو اتصال این Botnetها مقدار ۰/۳۵ است. همچنین پرحجم‌ترین اتصالات Botها مربوط به دانلود فایل باینری می‌باشد و از آنجایی که حجم فایل‌های باینری Botها حداکثر به چند صد کیلو بایت می‌رسد، مقدار متغیر BULKY_THRESHOLD را برابر یک مگا بایت قرار دادیم. برای کاهش شناسایی نادرست از پارامتر MAX_NUM_TW استفاده کردیم که با مطالعه Botnetهای مختلف به این نتیجه رسیدیم که در هر همبستگی حداکثر ۵ خوشه درگیر هستند. برای تعیین پارامتر CORRELATION_THRESHOLD نیز از میزان فعالیت Botها و نحوه انجام واکنش استفاده کردیم.

۳-۲-۱- جمع‌آوری داده و نحوه انجام آزمایش‌ها

ما برای جمع‌آوری ترافیک نرمال، از ترافیک یکی از شبکه‌های محلی مجازی[1] دانشگاه شهید بهشتی استفاده کرده‌ایم. ما فرض کرده‌ایم که این ترافیک تمیز بوده و دارای ترافیک‌های آلوده دیگر نیست. دلایل این فرض به شرح زیرند:

۱. کامپیوترهای شبکه محلی مجازی مورد نظر پشت NAT[2] هستند و مهاجم‌های بیرونی نمی‌توانند از آسیب‌پذیری‌های این کامپیوترها استفاده کرده و آن‌ها را آلوده کنند.

۲. کاربران کامپیوترهای شبکه محلی مجازی مورد نظر دارای دسترسی مدیر سیستم[3] نبوده و نمی‌توانند بدافزارهایی که از طریق ایمیل یا فلش به سیستم وارد کرده‌اند را اجرا کنند.

با مشاهده ترافیک می‌توان گفت که ترافیک نرمال جمع‌آوری شده گستره‌ای از کاربردها مانند HTTP ، DNS را دارا می‌باشد. این تنوع ترافیک باعث می‌شود تا بتوان بهتر میزان کارایی و شناسایی نادرست سیستم را سنجید.

همچنین در کل چهار ترافیک مختلف Botnet جمع کرده‌ایم که شامل دو Botnet مبتی بر چت و دو Botnet مبتنی بر وب می‌شود. برای جمع‌آوری ترافیک Botnet، یک شبکه کاملاً مجازی ایجاد کردیم که شامل ۱۷ ماشین مجازی با سیستم عامل ویندوز اکس پی است. هر کدام از Botهای فوق را روی چهار عدد از این ماشین‌ها نصب کردیم. همچنین روی یکی از کامپیوترهای مجازی برنامه سرور چت و برنامه سرور وب نصب کردیم.

Botهای مبتنی بر چت از کدهای تغییر یافته Botnetهای واقعی Spybot و SdBot [۵] استفاده می‌کنند. Botهای مبتنی بر چت به سرور چت کنترل شده متصل شده و به عضو کانال‌های تعریف شده می‌شوند، سپس منتظر دستورات Botmaster می‌مانند. برای فرمان دادن به Botهای مبتنی بر چت، ما یک نرم‌افزار Botmaster تعریف

[1] Virtual Local Area Network (VLAN)
[2] Network Address Translation (NAT)
[3] Administrator

کرده‌ایم که در زمان‌های مشخص، دستورات مشخصی را از طریق کانال‌های تعریف شده ارسال می‌کند.

Botهای HTTP-I و HTTP-II با توجه به توصیفات ارتباطات دستور و کنترل تحت وب که در [۴۴] ذکر شده، نوشته شده‌اند. این دو Bot با سرور وب کنترل شده ارتباطات برقرار می‌کنند و دستورات (ارسال اسپم) را انجام می‌دهند. HTTP-I Bot به صورت متناوب به سرور وصل می‌شود (هر ۵ دقیقه یک بار). HTTP-II Bot دارای ارتباطات منظمی نبوده و قبل از وصل شدن به سرور یک زمان تصادفی بین ۰ و ۱۰ دقیقه صبر می‌کند. همچنین فایل‌هایی از قبل ساخته شده‌اند که دارای متن ایمیل‌هایی هستند که باید به عنوان اسپم توسط این دو Bot مبتنی بر وب فرستاده شوند. این فایل روی سرور وب تحت کنترلی گذاشته شده‌اند و توسط Botها دانلود می‌شوند. جدول ۳-۳ آمارهای ترافیک‌های جمع‌آوری شده برای ارزیابی سیستم را نشان می‌دهد.

جدول ۳-۳) اطلاعات ترافیک‌های جمع‌آوری شده

ترافیک	اندازه (بایت)	مدت (ساعت)	بسته‌ها				اتصالات TCP
			کل	ICMP	TCP	UDP	
نرمال	۴۳/۸ گیگا	۲۱۵/۸	۷۴,۹۷۴,۶۹۲	۶۵۸,۷۴۷	۶۱,۷۵۲,۶۰۹	۱۲,۵۶۳,۳۳۶	۸۱۸,۳۶۲
SdBot	۲/۳ مگا	۲۲/۹	۱۸,۶۸۵	۰	۱۷,۰۷۲	۱,۶۱۳	۱۶
SpyBot	۳۷/۹ مگا	۲۲/۹	۴۶۸,۰۲۶	۲,۷۰۸	۴۶۴,۰۵۴	۱,۲۶۴	۴۲۱,۸۷۰
HTTP-I	۶/۲ مگا	۲۲/۹	۱۶,۹۲۶	۰	۱۵,۲۱۴	۱,۷۱۲	۱,۱۶۴
HTTP-II	۵/۳ مگا	۲۲/۹	۱۴,۵۹۴	۰	۱۴,۰۵۴	۵۴۰	۱,۰۶۹

۳-۲-۲- نتایج ارزیابی

برای ارزیابی دقت شناسایی سیستم BotGrabber، ما ترافیک‌های Botnet را با ترافیک نرمال تلفیق کردیم. فرض کنید تعداد کامپیوترهای موجود در ترافیک Botnet n باشد. ما به طور تصادفی n آدرس IP که در ترافیک نرمال استفاده نشده را به n آدرس Bot IPهای موجود در ترافیک Botnet نگاشت کردیم و شروع ترافیک Botnet را به طور تصادفی در بازه زمانی ترافیک نرمال قرار دادیم. بدین طریق یک ترافیک تلفیق شده از ترافیک نرمال و ترافیک Botnet ساخته ایم. همچنین برای بررسی دقیقتر کارایی سیستم از فیلترینگ لیست سفید استفاده نکرده‌ایم.

جدول ۳-۴ نتایج شناسایی را نشان می‌دهد.

جدول ۳-۴) نتایج شناسایی سیستم BotGrabber

ترافیک	تعداد کامپیوتر	تعداد کامپیوتر شناسایی شده به عنوان Bot
نرمال	۲۸	۲ (۲ شناسایی نادرست)
IRC-SdBot	۴	۴
IRC-Spybot	۴	۴
HTTP-I	۴	۴
HTTP-II	۴	۲ (۲ عدم شناسایی نادرست)

همانطور که مشاهده می کنیم، سیستم دارای ۲ عدم شناسایی نادرست و ۲ شناسایی نادرست است و سیستم BotGrabber دارای قدرت شناسایی بالا با تعداد نسبتاً کم شناسایی نادرست روی ترافیک نرمال است. با بررسی‌های انجام شده به این نتیجه رسیدیم که با اعمال فیلترینگ لیست سفید، شناسایی‌های نادرست از بین می‌روند.

فصل چهارم: نتیجه‌گیری و کارهای آینده

در این فصل ابتدا یک نتیجه‌گیری از بحث‌های صورت گرفته انجام می‌گیرد و سپس کارهای آینده معرفی می‌شوند.

۴-۱- نتیجه‌گیری

امروز Botnet به عنوان بزرگترین تهدید امنیتی در اینترنت شناخته می‌شود. میلیون‌ها کامپیوتر در سراسر دنیا آلوده شده‌اند و می‌توانند تحت کنترل Botmaster قرار گیرند تا فعالیت‌های بدخواهانه انجام دهند. بنابراین ما احتیاج به راهکاری برای شناسایی Botnetها داریم تا بتوانیم به خوبی با آن‌ها مقابله کنیم.

در این پایان‌نامه، ما یک روش کاملاً مبتنی بر ناهنجاری برای شناسایی Botnet ارائه دادیم و سیستم BotGrabber را بر اساس آن پیاده‌سازی کردیم. سیستم شناسایی Botnet سعی کرده اهداف اشاره شده در فصل اول را فراهم کند. اهداف مورد نظر عبارتند از:

- سیستم BotGrabber رفتارهای بنیادی و تغییر ناپذیر Botnetها را شناسایی می‌کند.
- سیستم BotGrabber فراگیر است و مختص یک نوع Botnet خاص نیست و همچنین وابسته به یک ساختار یا پروتکل خاص دستور و کنترل نمی‌باشد.
- سیستم BotGrabber یک سیستم شناسایی عملی است که می‌تواند در شبکه‌های واقعی کار کند. همچنین سیستم BotGrabber می‌تواند با دقت بالایی Botnetهای واقعی را شناسایی کند و دارای شناسایی نادرست کمی روی ترافیک نرمال است و همچنین دارای مصرف منابع معقولی می‌باشد.

۴-۲-کارهای آینده

در آینده قصد داریم تا موارد زیر را مورد مطالعه قرار دهیم:

- بهینه‌سازی کارایی مؤلفه‌های سیستم BotGrabber: ما قصد داریم روش‌های جدیدی برای افزایش کارایی مؤلفه‌های مانیتورینگ، خوشه‌بندی و همبستگی به دست آوریم تا میزان محدودیت‌های ذکر شده را کاهش دهیم.
- شناسایی Botnet در شبکه‌های بزرگ و پر سرعت: ما قصد داریم سیستمی توزیع‌شده طراحی کنیم که درجه بلادرنگی و برخطی بهتری داشته و در شبکه‌های بزرگ و پر سرعت قابل استفاده باشد.
- استفاده از روش‌های فعال برای شناسایی Botnet: ما قصد داریم روش‌های فعال را مورد مطالعه قرار دهیم تا

بتوانیم از آن به عنوان مکمل برای سیستم منفعل BotGrabber استفاده کنیم.

- استفاده از روش‌های مبتنی بر میزبان: ما قصد داریم از روش‌های شناسایی مبتنی بر میزبان به عنوان مکملی برای روش‌های مبتنی بر شبکه استفاده کنیم. روش‌های مبتنی بر میزبان می‌توانند اطلاعاتی را فراهم کنند که روش‌های مبتنی بر شبکه قادر به آن نیستند. ترکیب این دو روش می‌تواند دقت شناسایی را بالاتر ببرد.
- ایجاد لیست سفید پویا: در حال حاضر لیست سفید مورد استفاده ایستا بوده و باید به صورت دستی تنظیم شود. برای افزایش دقت و همچنین خودکار شدن فرآیند لیست سفید، برای هر سرور موجود در لیست سفید یک برچسب زمانی در نظر خواهیم گرفت.
- تکمیل ماژول تشخیص حملات DDoS: رابطه‌های (۴-۱)، (۴-۲) و (۴-۳) به ترتیب برای محاسبه امتیاز سطرهای جداول TCP SYN Flood، UDP Flood و ICMP Ping Flood استفاده می‌شوند.

S (۴-۱)

S (۴-۲)

S (۴-۳)

در رابطه‌های (۴-۱)، (۴-۲) و (۴-۳) pkts تعداد بسته‌های فرستاده شده، avgBpp متوسط تعداد بایت در بسته و std انحراف از معیار زمان بین بسته‌های فرستاده شده، هستند. ضرایب α، β و γ با استفاده از روش‌های کلاسه‌بندی و آموزش مدل کلاسه‌بندی توسط ترافیک‌های نرمال و حملات مربوطه بدست آورد. برای خوشه‌بندی فعالیت‌های DDoS، کامپیوترها را به سه خوشه تقسیم می‌کنیم:

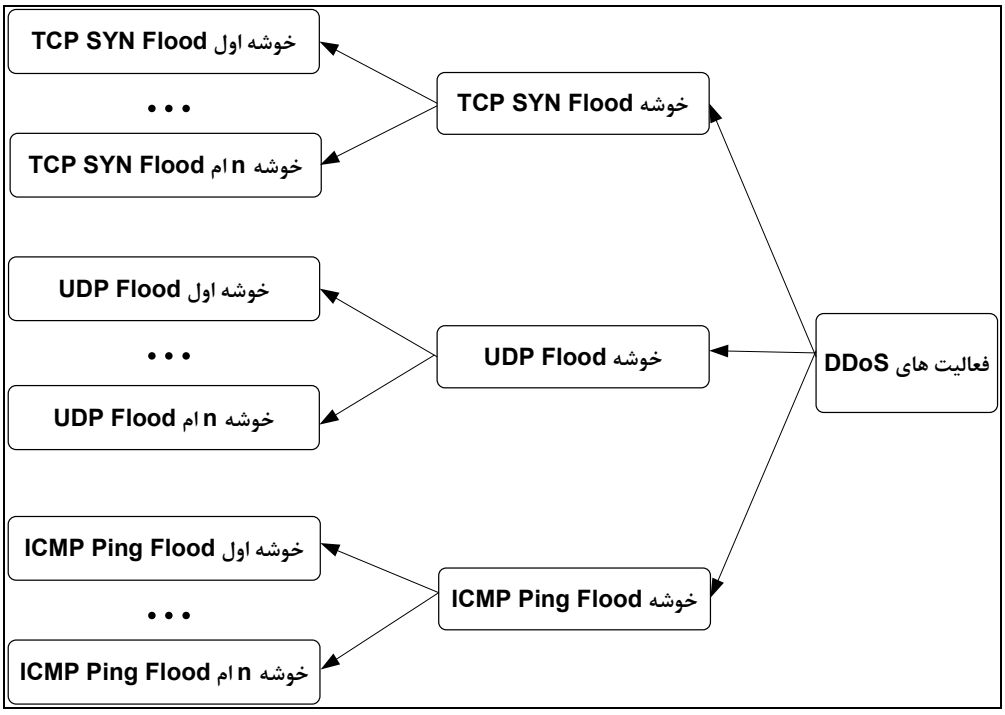
۱. TCP SYN Flood: در این خوشه، کامپیوترها را می‌توان با معیارهای زیر خوشه‌بندی کرد:
   أ. پورت‌های مورد حمله: بدین معنی که کامپیوترهایی که پورت‌های یکسانی را مورد حمله قرار می‌دهند، در یک خوشه قرار می‌گیرند.
   ب. شبکه یا زیرشبکه مورد حمله: بدین معنی که کامپیوترهایی که کامپیوترهای یک شبکه یا زیر شبکه یکسانی را مورد حمله قرار داده‌اند، در یک خوشه قرار می‌گیرند.
۲. UDP Flood: در این خوشه، کامپیوترها را می‌توان با معیارهای زیر خوشه‌بندی کرد:
   أ. پورت‌های مورد حمله: بدین معنی که کامپیوترهایی که پورت‌های یکسانی را مورد حمله قرار می‌دهند، در یک خوشه قرار می‌گیرند.
   ب. شبکه یا زیرشبکه مورد حمله: بدین معنی که کامپیوترهایی که کامپیوترهای یک شبکه یا زیر

شبکه یکسانی را مورد حمله قرار داده‌اند، در یک خوشه قرار می‌گیرند.

۳. ICMP Ping Flood: در این خوشه، کامپیوترها را می‌توان با معیارهای زیر خوشه‌بندی کرد:

أ. شبکه یا زیرشبکه مورد حمله: بدین معنی که کامپیوترهایی که کامپیوترهای یک شبکه یا زیر شبکه یکسانی را مورد حمله قرار داده‌اند، در یک خوشه قرار می‌گیرند.

شکل ۴-۱ شیوه خوشه‌بندی هشدارهای DDoS را نشان می‌دهد.



شکل ۴-۱) خوشه‌بندی چند سطحی هشدارها

پیوست ۱: فرهنگ واژگان فارسی به انگلیسی

الف

ابزار پایه‌ای	Rootkit
اتصال	Netflow
اسکن کردن	Scanning
اسم خاص	Nickname
اشتراک‌گذاری فایل	File Sharing
اعلام همگانی	Broadcast
امضاء	Signature
ایستا	Static
آستانه	Threshold
آسیب‌پذیری	Vulnerability
آماده‌سازی	Preparation

ب

بازپخش	Replay
بدافزار	Malware
بدخواه	Malicious
بدون حالت	Stateless
برچسب زمانی	Time Stamp
برخط	Online
برون‌سو	Out-bound
بلادرنگ	Real-time

پ

پراکندگی	Diversity
پنجره زمانی	Time Window
پویا	Dynamic

پیش‌پردازش	Preprocess

ت

تراکم	Density
تروجان	Trojan Horse
تصادفی	Random
تطبیق دادن	Matching
تقلیدی	Mimicry
تناوبی	Periodic
توزیع بایت	Byte Distribution
توزیع‌شده	Distributed
توزیع‌کننده	Dispatcher
تولیدکننده	Generator

ث

ثبت	Log
ثبت‌کننده کلید	Keylogger

ج

جاروب	Sweep
جاسوس‌افزار	Spyware
جدول مراجعه	Lookup Table

چ

چت	Chat

ح

حد برش	Cut-off
حریصانه	Heuristic

خ

خرابی در یک قسمت	Single Point of Failure

خود همبستگی	Self-Correlation
خوشه‌بندی	Clustering

د

درگاه	Gateway
درون‌سو	In-bound

ز

زمان بین	Interval Time
زمانی	Temporal
زیررشته	Substring

س

سایت‌های کلاهبرداری	Phishing Sites
سربار	Header
سرور وب	Web Server
سکو	Platform
سلسله مراتبی	Hierarchical
سوء استفاده	Exploit
سیستم تشخیص نفوذ	Intrusion Detection System (IDS)

ش

شبکه محلی	Local Area Network (LAN)
شناسایی نادرست	False Positive
شیادانه	Fraudulent

ع

عبارت منظم	Regular Expression
عدم شناسایی نادرست	False Negative
عملیاتی	Operation

ف

فراخوانی سیستمی	System Call
فشاری	Push
فضایی	Spatial
فعال	Active

ق

قابلیت	Functionality
قلابی	Spoofed

ک

کارت اعتباری	Credit Card
کانال دستور و کنترل	Command and Control Channel (C&C)
کانال نهفته	Covert Channel
کرم	Worm
کششی	Pull

گ

گردآوری و ساماندهی	Rallying
گفتگو	Conversation

ل

لیست سفید	Whitelist
لیست سیاه	Blacklist
لیست مراقب	Watchlist

م

مبتنی بر امضاء	Signature-based
مبتنی بر شبکه	Network-based
مبتنی بر میزبان	Host-based
مبتنی بر ناهنجاری	Anomaly-based

مبدأ	Source
متمرکز	Centralized
متن باز	Open Source
مجازی	Virtual
مجموعه آموزش	Training Set
مدیر سیستم	Administrator
مسموم کردن	Poisoning
مقصد	Destination
منفعل	Passive
مهندسی اجتماعی	Social Engineering
مورد سوء استفاده قرار گرفته شده	Compromised
میزبان	Host

ن

ناهنجاری	Anomaly
نظیر به نظیر	Peer-to-Peer (P2P)
نگاشت‌گر	Mapper
نیمه‌کاره	Partial

ه

هشدار	Alert
هماهنگی	Synchronization
همبستگی	Correlation

و

واکنش	Response
ویروس	Virus
ویژگی	Feature

ی

یادگیری با نظارت	Supervised Learning
یادگیری بدون نظارت	Unsupervised Learning
یادگیری ماشین	Machine Learning

پیوست ۲: فرهنگ واژگان انگلیسی به فارسی

A

Active	فعال
Administrator	مدیر سیستم
Alert	هشدار
Anomaly	ناهنجاری
Anomaly-based	مبتنی بر ناهنجاری

B

Blacklist	لیست سیاه
Broadcast	اعلام همگانی
Byte Distribution	توزیع بایت

C

Centralized	متمرکز
Chat	چت
Clustering	خوشه‌بندی
Command and Control Channel (C&C)	کانال دستور و کنترل
Compromised	مورد سوء استفاده قرار گرفته شده
Conversation	گفتگو
Correlation	همبستگی
Covert Channel	کانال نهفته
Credit Card	کارت اعتباری
Cut-off	حد برش

D

Density	تراکم
Destination	مقصد
Dispatcher	توزیع‌کننده
Distributed	توزیع‌شده

Diversity	پراکندگی
Dynamic	پویا

E

Exploit	سوء استفاده

F

False Negative	عدم شناسایی نادرست
False Positive	شناسایی نادرست
Feature	ویژگی
File Sharing	اشتراک‌گذاری فایل
Fraudulent	شیادانه
Functionality	قابلیت

G

Gateway	درگاه
Generator	تولیدکننده

H

Header	سربار
Heuristic	حریصانه
Hierarchical	سلسله مراتبی
Host	میزبان
Host-based	مبتنی بر میزبان

I

In-bound	درون‌سو
Interval Time	زمان بین
Intrusion Detection System (IDS)	سیستم تشخیص نفوذ

K

Keylogger	ثبت‌کننده کلید

L

Local Area Network (LAN)	شبکه محلی
Log	ثبت
Lookup Table	جدول مراجعه

M

Machine Learning	یادگیری ماشین
Malicious	بدخواه
Malware	بدافزار
Mapper	نگاشت‌گر
Matching	تطبیق دادن
Mimicry	تقلیدی

N

Netflow	اتصال
Network-based	مبتنی بر شبکه
Nickname	اسم خاص

O

Online	برخط
Open Source	متن باز
Operation	عملیاتی
Out-bound	برون‌سو

P

Partial	نیمه‌کاره
Passive	منفعل
Peer-to-Peer (P2P)	نظیر به نظیر
Periodic	تناوبی
Phishing Sites	سایت‌های کلاهبرداری
Platform	سکو
Poisoning	مسموم کردن

Preparation	آماده‌سازی
Preprocess	پیش‌پردازش
Pull	کششی
Push	فشاری

R

Rallying	گردآوری و ساماندهی
Random	تصادفی
Real-time	بلادرنگ
Regular Expression	عبارت منظم
Replay	بازپخش
Response	واکنش
Rootkit	ابزار پایه‌ای

S

Scanning	اسکن کردن
Self-Correlation	خود همبستگی
Signature	امضاء
Signature-based	مبتنی بر امضاء
Single Point of Failure	خرابی در یک قسمت
Social Engineering	مهندسی اجتماعی
Source	مبدأ
Spatial	فضایی
Spoofed	قلابی
Spyware	جاسوس‌افزار
Stateless	بدون حالت
Static	ایستا
Substring	زیررشته
Supervised Learning	یادگیری با نظارت

Sweep	جاروب
Synchronization	هماهنگی
System Call	فراخوانی سیستمی

T

Temporal	زمانی
Threshold	آستانه
Time Window	پنجره زمانی
Time Stamp	برچسب زمانی
Training Set	مجموعه آموزش
Trojan Horse	تروجان

U

Unsupervised Learning	یادگیری بدون نظارت

V

Virtual	مجازی
Virus	ویروس
Vulnerability	آسیب‌پذیری

W

Watchlist	لیست مراقب
Web Server	سرور وب
Whitelist	لیست سفید
Worm	کرم

مراجع

[1] Staniford, S., Hoagland, J. and McAlerney, J., "Practical automated detection of stealthy portscans", *Journal of Computer Security*, vol. 10, no. 1, p. 136, 105 2002.

[2] Moore, D., Voelker, G. and Savage, S., "Inferring internet denial-of-service activity", in *Proceedings of USENIX Security Symposium*, 2001.

[3] Cooke, E., Jahanian, F. and McPherson, D., "The zombie roundup: Understanding, detecting, and disrupting botnets", in *Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2005.

[4] Ramachandran, A., Feamster, N. and Dagon, D., "Revealing botnet membership using DNSBL counter-intelligence", in *Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2006.

[5] Barford, P. and Yegneswaran, V., "An inside look at botnets", in *Proceedings of Special Workshop on Malware Detection*, 2006.

[6] Goebel, J. and Holz, T., "Rishi: Identify bot contaminated hosts by IRC nickname evaluation", in *Proceedings of USENIX Hot Topics in Understanding Botnets (HotBots)*, 2007.

[7] Provos, N., Navrommatis, P., Rajab, M. and Monrose, F., "All your iframes point to us", in *Proceedings of USENIX Security Symposium*, 2008.

[8] Dagon, D., Zou, C. and Lee, W., "Modeling botnet propagation using time zones", in *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2006.

[9] Grizzard, B., Sharma, V., Nunnery, C., Kang, B. and Dagon, D., "Peer-to-peer botnets: Overview and case study", in *Proceedings of USENIX Hot Topics in Understanding Botnets (HotBots)*, 2007.

[10] Holz, T., Steiner, M., Dahl, F., Biersack, E. and Freiling, F., "Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm", in *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[11] Vogt, R., Aycock, J. and Jacobson, M., "Army of botnets", in *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2007.

[12] Wang, P., Sparks, S. and Zou, C., "An advanced hybrid peer-to-peer botnet", in *Proceedings of USENIX Hot Topics in Understanding Botnets (HotBots)*, 2007.

[13] Ramachandran, A. and Feamster, N., "Understanding the network-level behavior of spammers", in *Proceedings of ACM SIGCOMM*, 2006.

[14] Zhao, Y. et al., "BotGraph: Large scale spamming botnet detection", in *Proceedings of USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[15] Ehrlich, W., Karasaridis, A., Liu, D. and Hoeflin, D., "Detection of spam hosts and spam bots using network flow traffic modeling", in *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2010.

[16] Hu, X., Knysz, M. and Shin, K., "RB-Seeker: Auto-detection of redirection botnets", in *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2009.

[17] Gu, G., Perdisci, R., Zhang, J. and Lee, W., "BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection", in *Proceedings of USENIX Security Symposium*, 2008.

[18] Gu, G., Zhang, J. and Lee, W., "BotSniffer: Detecting botnet command and control channels in network traffic", in *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2008.

[19] Masud, M., Al-khateeb, T., Khan, L., Thuraisingham, B. and Hamlen, K., "Flow-based identification of botnet traffic by mining multiple log file", in *Proceedings of International Conference on Distributed Frameworks & Applications (DFMA)*, 2008.

[20] Yen, T. and Reiter, M., "TAMD: Traffic aggregation for malware detection", in *Proceedings of International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2008.

[21] Dagon, D., "Botnet detection and response", in *OARC Workshop*, 2005.

[22] Binkley, J. and Singh, S., "An algorithm for anomaly-based botnet detection", in *Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2006.

[23] Tu, H., Li, Z. and Liu, B., "Detecting botnets by analyzing DNS traffic", in *Proceedings of Pacific Asia conference on Intelligence and Security Informatics*

(PAISI), 2007.

[24] Holz, T., Gorecki, C., Rieck, K. and Freiling, F., "Detection and mitigation of fast-flux service networks", in *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2008.

[25] Scharrenberg, P., "Analyzing fast-flux service networks", Diploma Thesis, University of Mannheim, 2008.

[26] Hsu, C., Huang, C. and Chen, K., "Fast-flux bot detection in real time", in *Proceedings of International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2010.

[27] Passerini, E., Paleari, R., Martignoni, L. and Bruschi, D., "FluXOR: Detecting and monitoring fast-flux service networks", in *Proceedings of International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2008.

[28] Choi, H., Lee, H., Lee, H. and Hyogon, K., "Botnet detection by monitoring group activities in DNS traffic", in *Proceedings of IEEE International Conference on Computer and Information Technology (CIT)*, 2007.

[29] Villamarín-Salomón, R. and Brustoloni, J., "Bayesian bot detection based on DNS traffic similarity", in *Proceedings of ACM Symposium on Applied Computing (SAC)*, 2009.

[30] Livadas, C., Walsh, R., Lapsley, D. and Strayer, W., "Using machine learning techniques to identify botnet traffic", in *Proceedings of IEEE Conference on Local Computer Networks (LCN)*, 2006.

[31] Strayer, W., Walsh, R., Livadas, C. and Lapsley, D., "Detecting botnets with tight command and control", in *Proceedings of IEEE Conference on Local Computer Networks (LCN)*, 2006.

[32] Karasaridis, A., Rexroad, B. and Hoeflin, D., "Wide-scale botnet detection and characterization", in *Proceedings of USENIX Hot Topics in Understanding Botnets (HotBots)*, 2007.

[33] Gu, G., Porras, P., Yegneswaran, V., Fong, M. and Lee, W., "BotHunter: Detecting malware infection through IDS-driven dialog correlation", in *Proceedings of*

USENIX Security Symposium, 2007.

[34] Gianvecchio, S., Xie, M., Wu, Z. and Wang, H., "Measurement and classification of humans and bots in internet chat", in *Proceedings of USENIX Security Symposium*, 2008.

[35] Gu, G., Yegneswaran, V., Porras, P., Stoll, J. and Lee, W., "Active botnet probing to identify obscure command and control channels", in *Proceedings of Annual Computer Security Applications Conference (ACSAC)*, 2009.

[36] Zhuge, J., Holz, T., Han, X., Guo, J. and Zou, W., "Characterizing the IRC-based botnet phenomenon", Technical Report, University of Mannheim, 2007.

[37] Strayer, W., Lapsley, D., Walsh, B. and Livadas, C., "Botnet detection based on network behavior", in *Botnet Detection*, Lee, W., Wang, C. and Dagon, D., Springer, pp. 1-24, 2008.

[38] Seewald, A. and Gansterer, W., "On the detection and identification of botnets", *Computers & Security*, vol. 29, no. 1, pp. 45-58, 2010.

[39] Collins, M. et al., "Using uncleanliness to predict future botnet addresses", in *Proceedings of ACM on Internet Measurement Conference (IMC)*, 2007.

[40] Jain, A., Murty, M. and Flynn, P., "Data clustering: A review", *ACM Computer Survey*, vol. 31, no. 3, pp. 264–323, 1999.

[41] Pelleg, D. and Moore, A., "X-means: Extending k-means with efficient estimation of the number of clusters", in *Proceedings of International Conference on Machine Learning (ICML)*, 2000.

[42] Bailey, M. et al., "Automated classification and analysis of internet malware", in *Proceedings of International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2007.

[43] Wehner, S., "Analyzing worms and network traffic using compression", *Journal of Computer Security*, vol. 15, no. 3, pp. 303–320, 2007.

[44] Ianelli, N. and Hackworth, A., "Botnets as a vehicle for online crime", in *Proceedings of FIRST conference*, 2006.

Abstract

Most of the attacks and fraudulent activities on the Internet are carried out by malware. In particular, botnets have become a primary "platform" for attacks on the Internet. A botnet is a network of compromised computers (bots) that are under the control of a botmaster through some command and control (C&C) channels. It typically contains tens to hundreds of thousands of bots, but some even had several millions of bots. Botnets are now used for distributed denial-of-service (DDoS) attacks, spamming, phishing, information theft, etc. With the magnitude and the potency of attacks afforded by their combined bandwidth and processing power, botnets are now considered as the largest threat to the internet security.

Counteracting this emerging threat requires better detection techniques. In this thesis, we focus on addressing the botnet detection problem in an enterprise network environment. We propose a pure anomaly-based approach that requires no a priori knowledge of bot signatures, botnet C&C protocols, and C&C server addresses. We start from inherent characteristics of botnets. Bots have to respond to received commands and the bots which belong to the same botnet respond to the received commands in a similar fashion. Our method divides bots into clusters with similar behaviors in different time windows and performs correlation analysis between clusters to identify infected hosts with bots. We develop BotGrabber prototype system and evaluate it with real-world traces including normal traffic and several real-world botnet traces. The results show that BotGrabber has achieved high detection accuracy and a low False Positive.

Keywords: Botnet, Botmaster, C&C Channel, Anomaly, Netflow, Clustering.

Shahid Beheshti University
Department of Electrical & Computer Engineering

Botnet Detection based on Anomaly Behaviors in Network Traffic

Supervisor:
Dr. Maghsoud Abbaspour

A THESIS SUBMITTED
FOR THE DEGREE OF
MASTER OF SCIENCE

For Partial Fulfillment in MSc Degree in Shahid Beheshti University

By
Sajjad Arshad

Advisor:
Dr. Mehdi Kharrazi

Winter 2011