به نام خدا

پروژه نهایی درس امنیت شبکه‌های کامپیوتری

عنوان پروژه: تشخیص بات‌نت‌های P2P با کشف وابستگی جریان در ترافیک فرمان و کنترل (C&C)

این پروژه نهایی برای دانشجویان تحصیلات تکمیلی در حوزه امنیت شبکه های کامپیوتری، بر اساس مقاله " Detecting P2P botnets by discovering flow dependency in C&C traffic" نوشته Hongling Jiang و Xiuli Shao طراحی شده است. هدف این پروژه، پیاده سازی رویکردی برای شناسایی بات‌نت‌های P2P با تمرکز بر روی ترافیک فرمان و کنترل (C&C) آن‌ها است.

مقدمه

بات‌نت‌ها به یکی از تهدیدات اصلی اینترنت تبدیل شده‌اند. مهاجمان از بات‌نت‌ها به عنوان زیرساخت برای انواع جرایم سایبری، از جمله حملات محروم‌سازی از سرویس توزیع‌شده (DDoS)، ارسال هرزنامه، کلاهبرداری کلیکی، فیشینگ و ثبت کلیدها، استفاده می‌کنند. با پیشرفت سرویس‌های ابری، مهاجمان آینده ممکن است از این سرویس‌ها برای ایجاد بات‌نت و راه‌اندازی حملات بهره ببرند.

بات‌نت‌ها از ساختارهای متمرکز به ساختارهای توزیع‌شده (Peer-to-Peer یا P2P) تکامل یافته‌اند. در یک بات‌نت P2P، برخلاف ساختارهای متمرکز ، هیچ نقطه مرکزی برای سرور C&C وجود ندارد. هر بات هم به عنوان کلاینت و هم به عنوان سرور عمل می‌کند. این ساختار باعث می‌شود که حتی در صورت آفلاین شدن برخی نودها، بات‌نت P2P به فعالیت خود ادامه دهد و در برابر تغییرات پویا مقاوم باشد. Storm ، Nugache و Waledac از جمله محبوب‌ترین بات‌نت‌های P2P هستند.

اکثر بات‌های P2P مدرن حملات خود را به صورت مخفیانه انجام می‌دهند و رویکردهای تشخیص مبتنی بر ترافیک مخرب بات‌ها ناکارآمد هستند. تشخیص بات‌نت‌های P2P با چالش‌های متعددی روبروست:

1. ترافیک بات‌نت‌های P2P به ترافیک شبکه‌های P2P قانونی شباهت دارد و در ترافیک عادی پنهان می‌شود.
2. بسیاری از بات‌نت‌های P2P مانند Nugache، Storm، Waledac و Conficker از مکانیزم‌های رمزنگاری استفاده می‌کنند که رویکردهای مبتنی بر محتوای بسته را بی‌اثر می‌سازد.

3. باتهای P2P فعالیت‌های مخرب خود را به صورت مخفیانه انجام می‌دهند.
4. در باتنت‌های P2P سرور مرکزی وجود ندارد.
5. باتها از طریق پورت‌های تصادفی با یکدیگر ارتباط برقرار می‌کنند.

این پروژه بر اساس مقاله‌ی "C traffic&Detecting P2P botnets by discovering flow dependency in C" ، یک رویکرد تشخیص باتنت P2P را پیشنهاد می‌کند که با تمرکز بر روی ویژگی‌های ذاتی ارتباطات فرمان و کنترل (C&C) آن‌ها، باتهای P2P را شناسایی می‌کند. این رویکرد نیازی به بررسی ترافیک مخرب یا اطلاعاتی از باتها که توسط سیستم‌های خارجی (مانند Honeypot یا IDS) ارائه شده، ندارد. این روش باتهای P2P را با کشف وابستگی جریان در ترافیک C&C شناسایی می‌کند. پس از کشف این وابستگی‌ها، رویکرد پیشنهادی با استفاده از تکنیک خوشه‌بندی، باتهای P2P را از میزبان‌های عادی متمایز می‌کند.

معماری و پیاده‌سازی رویکرد تشخیص

همانطور که در شکل 2 مقاله مشخص است ، این رویکرد از سه مؤلفه اصلی تشکیل شده است:

1. **Flow Capture**
2. **Flow Dependency Extractor**
3. **Bots Detector**

بخش اول) Flow Capture (۲۵ درصد نمره)

این مرحله وظیفه دریافت جریان‌های کاندید را بر عهده دارد و شامل دو ماژول است:

- **Flow Generation** (۱۰ درصد نمره)

برای تشخیص وابستگی جریان‌ها، بسته‌های خام باید به جریان تبدیل شوند. یک جریان به عنوان دنباله‌ای از بسته‌ها بین یک مبدأ و یک مقصد در یک اتصال تعریف می‌شود. جریان‌ها بر اساس اطلاعات پنج‌تایی شامل آدرس‌های IP، پورت‌ها و پروتکل (localIP, localPort, remoteIP, remotePort, protocol) تولید می‌شوند . localIP و localPort آدرس IP و پورت میزبان داخل شبکه نظارت شده هستند، در حالی که remoteIP و remotePort آدرس IP و پورت میزبان خارج از شبکه نظارت شده می‌باشند. جریان‌ها دوطرفه هستند، به این معنی که آدرس IP پورت مبدأ و آدرس IP پورت مقصد قابل جابجایی هستند. زمان بین هر دو بسته متوالی در یک جریان باید کمتر از آستانه Timeout باشد.

از آنجایی که اکثر بات‌نت‌های P2P از پروتکل UDP یا TCP برای ارتباط استفاده می‌کنند ، ما بر روی جریان‌های UDP و TCP تمرکز می‌کنیم. برای TCP، بسته‌های SYN, SYN-ACK, ACK نشان‌دهنده‌ی شروع جریان و FIN, ACK, FIN, ACK یا RST نشان‌دهنده‌ی پایان جریان هستند. فقط جریان‌های TCP موفق با handshake کامل تحلیل می‌شوند. برای UDP، بسته‌های با همان پنج‌تایی در یک جریان UDP تجمیع می‌شوند و زمان بین هر دو بسته متوالی باید کمتر از آستانه Timeout باشد.

برای کشف وابستگی جریان‌ها، از هدرهای بسته‌های TCP/UDP و اطلاعات زمانی استفاده می‌شود. این اطلاعات به راحتی قابل دسترسی هستند و مستقل از محتوای بسته می‌باشند.

اطلاعات جریان به صورت زیر ثبت می‌شود:

1. **پنج‌تایی (five-tuple)**
2. **زمان شروع (start time): زمان رسیدن اولین بسته در یک جریان**
3. **زمان پایان (end time): زمان رسیدن آخرین بسته در یک جریان**
4. **مدت زمان جریان (duration time): تفاوت بین زمان پایان و زمان شروع**
5. **تعداد بسته‌ها در یک جریان (number of packets)**
6. **حجم بایت‌های یک جریان (bytes of a flow): مجموع بایت‌های بسته‌ها در یک جریان**

راهنمایی برای پیاده‌سازی:

- **برای خواندن پکت‌ها از فایل‌های**

pcap، از کتابخانه scapy در پایتون استفاده کنید. تابع sniff در scapy امکان خواندن بسته‌ها از یک فایل pcap و ارسال اطلاعات هر بسته به یک تابع تعریف‌شده توسط کاربر را فراهم می‌کند.

- **تابع**

create_flow_using_sniff (یا نام مشابه) شما باید بسته‌ها را به عنوان ورودی بگیرد و جریان‌های مربوطه را تولید کند. مثال استفاده:

```
sniff(offline=pcap_file, prn=create_flow_using_sniff)
```

برای هر پکت، اطلاعات زیر را استخراج کنید : آدرس و پورت مبدأ، آدرس و پورت مقصد، حجم اطلاعات بدنه بسته، و زمان رسیدن بسته.

- با استفاده از این اطلاعات، همه جریان‌هایی که تا به حال ساخته شده‌اند را بررسی کنید. اگر شروط ذکر شده در مقاله (یکی بودن مبدا و مقصد یا یکی بودن مبدا پکت با مقصد جریان و بالعکس، و رسیدن پکت در بازه زمانی مشخص از آخرین پکت جریان) برای پکت مورد بررسی و جریان پیدا شده برقرار بود، این پکت را به جریان پیدا شده اضافه کنید. در غیر این صورت، با استفاده از این پکت یک جریان جدید بسازید.

- **Filter** (۱۰ درصد نمره اضافی - اختیاری)

برای کاهش حجم و نویز در ردیابی‌های شبکه، ترافیکی که احتمالاً به باتنت‌های P2P مرتبط نیست، فیلتر می‌شود. این فیلتر برای رویکرد تشخیص حیاتی نیست، اما به کاهش ترافیک و افزایش کارایی تشخیص کمک می‌کند.

فیلتر در سه مرحله انجام می‌شود:

1. فیلتر کردن جریان‌های با حجم کوچک: جریان‌های C&C باتنت‌ها معمولاً از بسته‌های کوچک تشکیل شده‌اند و حجم کل بایت‌های آنها معمولاً کم است. در مقابل، بسیاری از جریان‌های قانونی (مانند جریان‌های چندرسانه‌ای) ممکن است حجم زیادی داشته باشند.
2. فیلتر کردن جریان‌های طولانی: یکی دیگر از ویژگی‌های جریان‌های C&C این است که مدت زمان آنها معمولاً کوتاه است. جریان‌های طولانی (مانند جلسات SSH یا اتصالات دسکتاپ از راه دور که ممکن است ساعت‌ها فعال بمانند) در شبکه رایج هستند و می‌توانند نویز ایجاد کنند. این جریان‌ها نادیده گرفته می‌شوند.
3. فیلتر کردن جریان‌هایی که به ندرت اتفاق می‌افتند: باتهای P2P به طور مکرر به همتایان خود در لیست‌های همتایانشان متصل می‌شوند. بنابراین، جریان‌های بین باتها و همتایانشان به طور مکرر اتفاق می‌افتند. جریان‌هایی که بین دو میزبان به ندرت اتفاق می‌افتند، ممکن است به باتنت‌ها مرتبط نباشند.

- نکته: از آنجایی که باتهای P2P معمولاً از پورت‌های تصادفی برای ارتباط استفاده می‌کنند (مانند Nugache)، شماره پورت نادیده گرفته شده و فقط آدرس‌های IP و پروتکل یک جریان در نظر گرفته می‌شوند.

بخش دوم) Flow Dependency Extractor (۶۵ درصد نمره)

این مؤلفه وابستگی‌های جریان را کشف می‌کند و میزبان‌هایی را که دارای وابستگی جریان هستند، به عنوان میزبان‌های کاندید در نظر می‌گیرد. این ماژول ابتدا وابستگی‌های جریان دو سطحی را کشف کرده و سپس بر اساس آن‌ها، وابستگی‌های جریان چند سطحی را پیدا می‌کند. جریان‌های وابستگی با امتیاز بالاتر از S_dep_th به عنوان وابستگی‌های جریان واقعی در نظر گرفته می‌شوند.

- **محاسبه تعداد رخداد (۱۵ درصد نمره)**

برای تشخیص وابستگی جریان‌ها، لازم است قبل از شروع تشخیص، تعداد رخداد هر جریان را شمرده و آن را برای همه جریان‌ها ذخیره کنید.

راهنمایی: تابعی بنویسید که همه جریان‌های مورد نظر را بگیرد. برای هر جریان، اگر تعداد تکرارش قبلاً حساب شده باشد، آن را دوباره بررسی نکند. اما اگر تعداد تکرارش حساب نشده باشد، تعداد تکرار را حساب کرده و مقدار آن را برای همه جریان‌های یکسان ذخیره کند.

- **ساخت وابستگی جریان دولایه‌ای (۳۰ درصد نمره)**

چگونه وابستگی‌های جریان دو سطحی را در یک ردیابی معین استخراج کنیم؟ بینش کلیدی ما این است که اگر یک جفت جریان به طور مداوم با هم رخ دهند، ممکن است رابطه وابستگی داشته باشند. ما وابستگی جریان‌ها را با جستجو برای همبستگی زمانی جریان‌ها کشف می‌کنیم. به عنوان مثال، اگر جریان B بلافاصله پس از جریان A مشاهده شود، می‌توان فرض کرد که جریان B به جریان A وابسته است. در حالی که همبستگی زمانی ممکن است همیشه نشان‌دهنده‌ی یک وابستگی واقعی نباشد، برای کاهش موارد مثبت کاذب به تعداد زیادی نمونه تکیه می‌کنیم. به عبارت دیگر، ایده ما استخراج جفت جریان‌هایی است که به طور مکرر با هم رخ می‌دهند و به طور مداوم در طول زمان تکرار می‌شوند.

با این حال، ممکن است تعداد زیادی جریان در چند ساعت وجود داشته باشد. بررسی هر جفت جریان پرهزینه، غیرقابل مقیاس‌بندی و ناکارآمد است. برای حل این مشکل، رویکرد ما دو اکتشافی را به کار می‌برد:

1. با توجه به یک جریان فعلی، فقط جریان‌هایی را تحلیل می‌کنیم که پس از یک زمان کوتاه از جریان فعلی شروع می‌شوند، زیرا یک جریان معمولاً به جریانی وابسته است که اخیراً رخ داده است.
2. فقط جریان‌هایی را تحلیل می‌کنیم که تقریباً تعداد دفعات یکسانی رخ می‌دهند. اگر دو جریان همیشه با هم رخ دهند، تعداد رخدادهای این جریان‌ها در یک دوره زمانی تقریباً با هم برابر خواهد بود.

برای پیاده‌سازی این روش:

- برای هر میزبان محلی h در مجموعه F (جریان‌های کاندید در یک دوره)، مجموعه جریان‌های G_h مربوط به آن میزبان را پیدا کنید.
- جریان‌ها را در G_h بر اساس زمان شروعشان مرتب کنید.
- برای هر جریان f_i در G_h، جریان‌های f_j را پیدا کنید که در بازه زمانی T_dep پس از f_i شروع می‌شوند.
- اگر چنین جریان‌هایی وجود داشت، بررسی کنید که تفاوت در تعداد رخداد f_i و f_j (یعنی $|N_j-N_i|$) کمتر از N_dep باشد.
- اگر این شرط هم برای دو جریان مورد نظر برقرار بود، آنگاه $f_i \leftarrow f_j$ به عنوان یک وابستگی جریان کاندید در نظر گرفته می‌شود. متغیر T_{ij} نشان‌دهنده تعداد دفعاتی است که f_j بلافاصله پس از f_i رخ می‌دهد و یک واحد به آن اضافه می‌شود.
- برای محاسبه امتیاز وابستگی دو جریان از فرمول زیر استفاده کنید:

$$S_{dep}(f_i \rightarrow f_j) = \frac{Tij}{\sqrt{Ni \cdot Nj}}$$

که در آن N_i تعداد رخدادهای جریان f_i است. هر چه این امتیاز بیشتر باشد، احتمال وابستگی دو جریان بیشتر است. اگر امتیاز یک وابستگی جریان از S_dep_th بزرگتر باشد، آن را به عنوان یک وابستگی جریان واقعی در نظر می‌گیریم. الگوریتم کامل پیاده‌سازی این قسمت طبق مقاله (Algorithm 1) به صورت زیر می‌باشد:

```
Algorithm 1 Extracting two-level flow dependencies
Input: F-the set of candidate flows during an epoch
Output: Two-level flow dependencies

Compute the number of occurrences
   for each flow f in F, denoted as N_i
foreach local host h in F do
    Find the set of flows G_h of h
    Sort the flows in G_h in the order of start time
    foreach flow f_i in G_h do
       Find flows F_h start after f_i within T_dep
       foreach flow f_j in F_h do
          if |N_i - N_j| < N_dep then
```

```
            if f_i -> f_j in candidate flow dependencies set D then
              T_ij = T_ij + 1
          else
              Insert f_i -> f_j to D
foreach f_i -> f_j in D do
    Compute S_dep(f_i -> f_j(
    if S_dep(f_i -> f_j) > S_dep_th then
       Label f_i -> f_j as true flow dependency
```

- **ساخت وابستگی جریان چند لایه‌ای (۲۰ درصد نمره)**

تاکنون، وابستگی‌های جریان دو سطحی کشف شده‌اند. با این حال، وابستگی‌های جریان دو سطحی در ترافیک عادی نیز وجود دارند. به عنوان مثال، یک جریان به سرور وب اغلب وابسته به یک جریان به سرور DNS است. اما وابستگی‌های جریان بات‌های P2P معمولاً شامل بیش از دو جریان می‌شوند، زیرا تعداد همتایان در لیست همتایان یک بات P2P معمولاً بیش از دو است. برای شناسایی بات‌های P2P، وابستگی‌های جریان چند سطحی باید استخراج شوند. وابستگی‌های جریان چند سطحی با ترکیب وابستگی‌های جریان سطح پایین‌تر استخراج می‌شوند. به عنوان مثال، اگر وابستگی‌های جریان دو سطحی $f2 \rightarrow f1$ و $f3 \rightarrow f2$ وجود داشته باشند، این ترکیب یک وابستگی جریان سه‌سطحی $f3 \rightarrow f2 \rightarrow f1$ را نتیجه می‌دهد. به همین ترتیب، اگر $f4 \rightarrow f3$ یک وابستگی جریان واقعی باشد، می‌تواند یک وابستگی جریان چهارسطحی $f4 \rightarrow f3 \rightarrow f2 \rightarrow f1$ را نتیجه دهد.

امتیاز یک وابستگی جریان k-سطحی به صورت زیر محاسبه می‌شود:

$$S(f1 \rightarrow f2 \rightarrow \ldots \rightarrow fk) = \left(\prod_{i=1}^{k-1} S(fi \rightarrow fi+1)\right)$$

هنگام کشف وابستگی‌های جریان چند سطحی، فقط وابستگی‌های جریان واقعی را در نظر می‌گیریم که امتیاز آنها از S_dep_th بیشتر باشد. میزبان‌های محلی که دارای وابستگی‌های جریان دو سطحی یا چند سطحی هستند، به احتمال زیاد بات‌های P2P هستند و به عنوان میزبان‌های کاندید در نظر گرفته می‌شوند.

راهنمایی: برای تشخیص وابستگی‌های چند لایه‌ای، در ابتدا باید همه وابستگی‌های دولایه‌ای را با خودشان بررسی کنیم و به صورت بازگشتی یا تکراری وابستگی‌های با سطوح بالاتر را بسازیم.

بخش سوم) Bots Detector (۲۵ درصد نمره - اضافی)

پس از کشف وابستگی‌های جریان، ماژول Bots Detector برای شناسایی بات‌های P2P اعمال می‌شود.

- **محاسبه فاصله (۱۵ درصد نمره اضافی)**

در مرحله اول، فاصله بین هر جفت از میزبان‌های کاندید محاسبه می‌شود. همانطور که قبلاً ذکر شد، بات‌های یک باتنت P2P با همتایان موجود در لیست‌های خود برای دریافت دستورات یا بروزرسانی‌ها ارتباط برقرار می‌کنند. اگرچه بات‌های مختلف به همتایان مختلفی در لیست‌های خود متصل می‌شوند، اما احتمال دارد که هر جفت از بات‌های P2P حداقل به یک همتای مشترک متصل شوند، زیرا بات‌های یک باتنت از پروتکل P2P و شبکه یکسانی استفاده می‌کنند. بر اساس این مشاهده، فاصله بین میزبان‌های کاندید با توجه به همپوشانی میزبان‌های مشترک در وابستگی‌های جریان آن‌ها اندازه‌گیری می‌شود.

با داشتن مجموعه‌ای از وابستگی‌های جریان k-سطحی، از ضریب شباهت جاکارد برای اندازه‌گیری شباهت k-ام بین دو میزبان H_a و H_b استفاده می‌کنیم. شباهت k-ام به صورت زیر تعریف می‌شود:

$$Sim^{(k)}(H_a, H_b) = \begin{cases} \dfrac{\left|DEP_a^{(k)} \cap DEP_b^{(k)}\right|}{\left|DEP_a^{(k)} \cup DEP_b^{(k)}\right|} & if \left|DEP_a^{(k)} \cap DEP_b^{(k)}\right| \neq 0, \left|DEP_a^{(k)} \cup DEP_b^{(k)}\right| \neq 0 \\ 0 \quad otherwise \end{cases}$$

که در آن $DEP_a^{(k)}$ و $DEP_b^{(k)}$ به ترتیب مجموعه‌های میزبان‌های راه دور در وابستگی‌های جریان k-سطحی میزبان H_a و H_b هستند.

سپس فاصله بین دو میزبان H_a و H_b به صورت زیر تعریف می‌شود:

$$Dist(H_a,\ H_b) = \sum_{k=2}^{n} w_k * \left(1 - Sim^{(k)}(H_a, H_b)\right), w_k = \frac{k}{\sum_{k=2}^{n} k}$$

w_k وزن k-ام فاصله است و با تغییر k، w_k نیز تغییر می‌کند. همانطور که قبلاً دیدیم، میزبان‌های عادی ممکن است وابستگی جریان داشته باشند، اما سطوح آنها معمولاً پایین‌تر از بات‌های P2P است. برای شناسایی بات‌های P2P، w_k با افزایش k افزایش می‌یابد.

راهنمایی:

- ابتدا لازم است میزبان‌هایی که برای آن‌ها وابستگی جریان پیدا شده است را در لیستی نگه دارید.

- برای محاسبه فرمول‌های فوق، نیاز دارید که برای هر میزبان (IP میزبان) و لیستی از وابستگی‌های میزبان را داشته باشید؛ به این صورت که هر عضو از لیست دارای دو مقدار باشد: عدد لایه وابستگی و تمام مقصدهایی که در این وابستگی بوده‌اند. پس نیاز به تابعی دارید که بتواند این اطلاعات را به دست آورد.
- سپس طبق فرمول ارائه شده در مقاله فاصله را محاسبه کنید.
- در نهایت یک ماتریس فاصله (Distance Matrix)

$D = d_ab * a, b = 1, \ldots, n$ شامل فاصله $d * ab = dist(Ha, Hb)$ بین هر جفت از میزبان‌ها ساخته می‌شود، که n تعداد میزبان‌های کاندید است.

- **خوشه‌بندی میزبان‌ها (۱۰ درصد نمره - اضافی)**

پس از محاسبه فاصله، یک الگوریتم خوشه‌بندی سلسله‌مراتبی تک-پیوندی (single-linkage hierarchical clustering algorithm) بر روی ماتریس فاصله D اعمال می‌شود. این الگوریتم D را به عنوان ورودی می‌گیرد و یک دندروگرام (dendrogram) تولید می‌کند که یک گراف شبیه درخت است (شکل 7 مقاله را ببینید). در این دندروگرام، برگ‌ها نشان‌دهنده میزبان‌های اصلی و طول یال‌ها نشان‌دهنده فواصل بین خوشه‌ها هستند.

دندروگرام رابطه بین میزبان‌ها را نشان می‌دهد. هر چه فاصله بین دو میزبان کمتر باشد، یال اتصال آن‌ها در دندروگرام کوتاه‌تر است. برای گروه‌بندی میزبان‌ها به خوشه‌ها، پارامتری به نام dist_th برای برش دندروگرام در ارتفاع مشخصی استفاده می‌شود.

dist_th بر اساس توزیع فواصل میان میزبان‌ها تنظیم می‌شود. اگر برخی میزبان‌ها فواصل کمی با یکدیگر داشته باشند و یک خوشه متراکم تشکیل دهند و میزبان‌های دیگر فواصل بزرگ‌تری داشته باشند، dist_th در میانه فواصل کوچک و بزرگ تنظیم می‌شود .

dist_th بسیار بزرگ ممکن است منجر به موارد مثبت کاذب شود (میزبان‌های عادی به خوشه‌های باتنت خوشه‌بندی شوند)، در حالی که dist_th بسیار کوچک ممکن است منجر به موارد منفی کاذب شود (برخی باتها تشخیص داده نشوند). بنابراین، dist_th باید بر اساس توزیع فواصل تنظیم شود. خوشه‌هایی که زیر dist_th قرار می‌گیرند به عنوان خوشه‌های بات طبقه‌بندی می‌شوند.

4. گزارش (۱۰ درصد نمره)

یک گزارش جامع از روند کار، شامل توضیحات پیاده‌سازی هر بخش، چالش‌های پیش رو و راه‌حل‌های اتخاذ شده، به همراه فایل کدها ارائه دهید. حتی در صورت عدم موفقیت در پیاده‌سازی کامل، ارائه گزارشی شامل الگوریتم‌های مقاله و روش کارتان، نمره مربوطه را به دنبال خواهد داشت.

نکته: برای ارزیابی عملکرد پیاده‌سازی خود، می‌توانید از معیارهای Detection Rate (DR) و False Positive Rate (FPR) که در مقاله (بند 4.2، صفحه 327) تعریف شده‌اند، استفاده کنید.

نکات مهم و راهنمایی‌ها برای دانشجویان:

- زبان برنامه‌نویسی: پیشنهاد می‌شود که از زبان برنامه‌نویسی پایتون استفاده کنید ، زیرا این زبان کتابخانه‌های بسیار مفیدی (مانند scapy برای خواندن اطلاعات شبکه) برای پیاده‌سازی ساده‌تر در اختیار شما قرار می‌دهد.
- دیتاست: برای دیتاست، فایل‌های pcap به صورت آماده در اینترنت هستند و می‌توانید از آن‌ها استفاده کنید. این فایل‌ها شامل پکت‌های مشاهده شده بر روی یک شبکه هستند. (یک نمونه ازpcap. در فایل قرار داده شده است.)
- منابع: مقاله اصلی را با دقت مطالعه کنید؛ جزئیات پیاده‌سازی به طور کامل در آن ذکر شده است.
- زمان تحویل پروژه : **ساعت ۱۱:۵۹ شب ۲۸ مرداد ماه ۱۴۰۴ - زمان تحویبل آنلاین متعاقبا اعلام خواهد شد.**
- ارزیابی: در موقع تحویل آنلاین، تسلط کامل بر پروژه ملاک عمل است و صرفاً بارگذاری پروژه کافی نیست.
- ارتباط: سوالات خود را می‌توانید به آدرس ایمیل **fatemeh.abbasi4@webmail.sbu.ac.ir** ارسال کنید.

معیارهای ارزیابی عملکرد

عملکرد رویکرد با Detection Rate (DR) و False Positive Rate (FPR) اندازه‌گیری می‌شود.

- Detection Rate (DR): نرخ بات‌هایی است که به درستی شناسایی شده‌اند. با فرمول زیر تعریف می‌شود:

$$DR = TP + FNTP$$

TP تعداد موارد مثبت واقعی (بات‌هایی که به عنوان بات طبقه‌بندی شده‌اند) و FN تعداد موارد منفی کاذب (بات‌هایی که به عنوان غیربات طبقه‌بندی شده‌اند) است.

- False Positive Rate (FPR): نرخ میزبان‌های عادی است که به اشتباه به عنوان بات طبقه‌بندی شده‌اند. با فرمول زیر تعریف می‌شود:

$$FPR = FP + TNFP$$

FP تعداد موارد مثبت کاذب (غیربات‌هایی که به عنوان بات طبقه‌بندی شده‌اند) و TN تعداد موارد منفی واقعی (غیربات‌هایی که به عنوان غیربات طبقه‌بندی شده‌اند) است.

بحث و تحلیل (جهت راهنمایی دانشجو برای بخش گزارش)

روش پیشنهادی بات‌نت‌های P2P را با کشف وابستگی‌های جریان شناسایی می‌کند. جریان‌های وابسته باید به طور مکرر با هم اتفاق بیفتند. اگر این جریان‌ها به ندرت اتفاق بیفتند، روش ممکن است در کشف وابستگی جریان دچار مشکل شود. این محدودیت در تکنیک‌های کشف وابستگی مبتنی بر شبکه رایج است. این محدودیت را می‌توان با افزایش دوره زمانی داده‌های جمع‌آوری شده کاهش داد.

میزبان‌های P2P قانونی نیز ممکن است به طور مکرر به همتایان خود در لیست‌های همتایانشان متصل شوند تا فایل‌ها را جستجو کنند یا شبکه P2P را حفظ کنند. آنها نیز ممکن است وابستگی جریان داشته باشند. در این روش، دو راه برای طبقه‌بندی میزبان‌های P2P قانونی و بات‌های P2P وجود دارد:

1. بر اساس زمان فعالیت: زمان فعالیت میزبان‌های P2P قانونی ممکن است کوتاه باشد و معمولاً توسط کاربران تعیین می‌شود. مثلاً کاربران ممکن است پس از دانلود فایل‌ها، برنامه P2P را ببندند. در حالی که زمان فعالیت بات‌های P2P طولانی است، زیرا تا زمانی که سیستم زیرین فعال باشد، اجرا می‌شوند. بنابراین، تعداد نمونه‌های وابستگی جریان میزبان‌های P2P قانونی ممکن است کمتر از بات‌های P2P باشد و مقدار T_{ij} جریان‌های قانونی ممکن است کوچکتر باشد.
2. بر اساس خوشه‌بندی: فرآیند خوشه‌بندی می‌تواند میزبان‌های P2P قانونی را حذف کند. از آنجایی که میزبان‌های P2P قانونی مختلف تمایل دارند فایل‌های مختلفی را جستجو کنند و همتایانی که به آنها متصل می‌شوند احتمال همپوشانی کمی دارند، فاصله میزبان‌های P2P قانونی بزرگ است و آنها نمی‌توانند یک خوشه متراکم تشکیل دهند. از سوی دیگر، بات‌های P2P در یک بات‌نت معمولاً به همتایان مشترک متصل می‌شوند و فواصل آنها کوچک است، بنابراین می‌توانند یک خوشه متراکم تشکیل داده و شناسایی شوند.