گزارش پروژه درس اندازه گیری اینترنت

حسین بیگی ۸۱۰۱۹۹۰۲۰

حسن خالقی راد ۸۱۰۱۹۹۰۳۱

محمد وطن دوست ۸۱۰۱۹۸۲۷۶

نادیا یزدانی ۸۱۰۱۹۹۳۰۹

۳ مرداد ۱۴۰۰

فهرست مطالب

معرفی ۲

مانیتورینگ شبکه ۳

حمله DoS ۴

برنامه رابط کاربری IDS ۶

ایجاد گراف شبکه ۸

تشخیص حملات با کمک یکریختی(Isomorphim) در گراف TDG ۹

تشخیص ناهنجاری ۱۰

منابع ۱۱

معرفی

حمله DoS (Denial of Service) برای جلوگیری از دسترسی کاربران قانونی به منابع شبکه یا سیستم رایانه ای طراحی شده است. طبقه بندی حمله DoS باعث کاهش زمان کشف الگو برای اپراتورهای شبکه می شود. پروژه ما با هدف تجزیه و تحلیل تفاوت بین گراف های پراکندگی ترافیک (TDG) در سری های زمانی برای شناسایی فعالیت های مخرب و استفاده از الگوریتم های همسان سازی برای شناسایی الگوهای حمله در ترافیک غیر عادی شبکه است.

این پروژه از دو بخش ذخیره log شبکه و تجزیه و تحلیل log برای شناسایی ترافیک حمله تشکیل شده است.

مانیتورینگ شبکه

برای خواندن داده ها از کتابخانه libpcap استفاده شده است. Libpacp یک کتابخانه cross-platform است، که در سطح User اجرا می شود. با استفاده از این کتابخانه به راحتی می توان با رابط شبکه هسته سیستم عامل ارتباط برقرار و بسته های شبکه را دریافت کرد. ابتدا لیستی از هایinterface شبکه سیستم تهیه می کنیم. سپس با اولین interface ارتباط برقرار می کنیم. تابع packetHandler را به عنوان callback ثبت می کنیم، به ازای هر بسته، یک باز این تابع فراخوانی می شود. ما بسته ها را در ۱۰۰ بازه زمانی دریافت می کنیم. بسته های هر بازه را داخل فایل اکسل، برای تحلیل ذخیره می کنیم.

```
dev = pcap_lookupdev(errbuf);
if (dev == NULL) {
    cout << "pcap_lookupdev() failed: " << errbuf << endl;
    return 1;
}

descr = pcap_open_live(dev, BUFSIZ, 0, -1, errbuf);
if (descr == NULL) {
    cout << "pcap_open_live() failed: " << errbuf << endl;
    return 1;
}
for(int i=0; i< number_of_time_slots; i++) {
    cout<<"----- time slot :"<< i << "--------------" << endl;
     outputFile = new std::ofstream ("packetsData_"+std::to_string(i)+".csv");
 (*outputFile) << "Time(uS) , Destination Address , Source Address, Protocol, IP Version, IP Header Length , "<<
            " Type Of Service , IP Total Length , Identification, TTL, Protocol, Checksum, Source IP, Destination IP " <<  std::endl;
if (pcap_loop(descr, 100, packetHandler, NULL) < 0) {
    cout << "pcap_loop() failed: " << pcap_geterr(descr);
    return 1;
}
 outputFile->close();
}
```

حمله DoS

DoS به حملاتی گفته می شود که نفوذگر با ارسال درخواست های بسیار به یک سرور یا کامپیوتر، باعث استفاده بیش از حد از منابع آن مانند پردازنده سرور، بانک اطلاعاتی، پهنای باند و ... میشود به صورتی که سرور مجازی یا اختصاصی میزبان سایت دچار کندی شده که به دلیل حجم بالای پردازش سیستم دچار وقفه و اختلال و یا حتی قطعی کامل شده و از دسترس خارج شود.

server.py برنامه یک سرور UDP(User Datagram Protocol) و TCP (Transmission Control Protocol)است. TCP پروتکل connection-oriented برای انتقال داده ها بین دو ماشین با ضریب اعتماد بالا و UDP پروتکل connection-less می باشد. ابتدا برای پروتکل UDP در سرور یک شی از سوکت به نام `udp_server_socket` می سازیم، در هنگام تعریف شی دو آرگومان را مشخص می کنیم که آرگومان اول نوع IP است که چون ما قصد استفاده از IPV۴ را داریم، گزینه AF_INET را وارد می کنیم. آرگومان دوم نیز پروتکل را تعیین می کند که چون در این بخش از پروتکل UDP استفاده کردیم، گزینه SOCK_DGRAM را انتخاب می کنیم و برای پروتکل TCP نیز SOCK_STRAM را انتخاب می کنیم. برای در دسترس بودن سوکت ایجاد شده، در متد bind دو آرگومان IP و port را وارد می کنیم که ما در اینجا از آی پی local host و شماره پورت ۵۳۶۷ استفاده کرده ایم. سپس در یک حلقه، سرور درخواست های کلاینت را دریافت می کند و به آن ها پاسخ می دهد.

برای پروتکل TCP هم به صورت مشابه عمل می کنیم برای اینکه سرور بتواند درخواست کلاینت را متوجه شود، از متد listen استفاده کردیم که آرگومان ورودی آن، تعداد کلاینت هایی که همزمان می توانند به سرور متصل شوند را نمایش می دهد. سپس با استفاده از متد accept می تواند درخواست اتصال کلاینت را بپذیرد.

dos_attack_udp.py و dos_attack_tcp.py برنامه های UDP و TCP برای حمله به سرور است. حمله به مدت ۵۰ میلی ثانیه طول می کشد و پکت های ۱۰۲۴ بایتی که به صورت رندوم تولید شده اند به سرور ارسال می شوند. اگر پکت ها بیشتر از این اندازه باشند، ممکن است به وسیله روتر فیلتر شوند. در طول زمان حمله، با استفاده از متد های sendto و send که به ترتیب برای UDP و TCP است، پکت ها ارسال می شوند.برای پروتکل TCP از متد connect برای ارسال درخواست اتصال به سرور استفاده می کنیم. این برنامه برای حمله DoS استفاده می شود، حمله DDoS را می توانیم با استفاده از همین برنامه، با سیستم های مختلف انجام دهیم. برنامه حمله به سرور، همزمان با برنامه مانیتورینگ شبکه اجرا می شود تا ترافیک حمله را ذخیره کنیم و تحلیل های لازم را برای شناسایی حمله انجام دهیم.

```
send 11549 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
send 11550 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
send 11551 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
send 11552 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
send 11553 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
send 11554 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
send 11555 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
send 11556 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
send 11557 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
send 11558 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
send 11559 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
send 11560 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
send 11561 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
send 11562 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
send 11563 packet to 127.0.0.1 throught port 20002 on a TCP connection.
message from server:Hello TCP Client
```

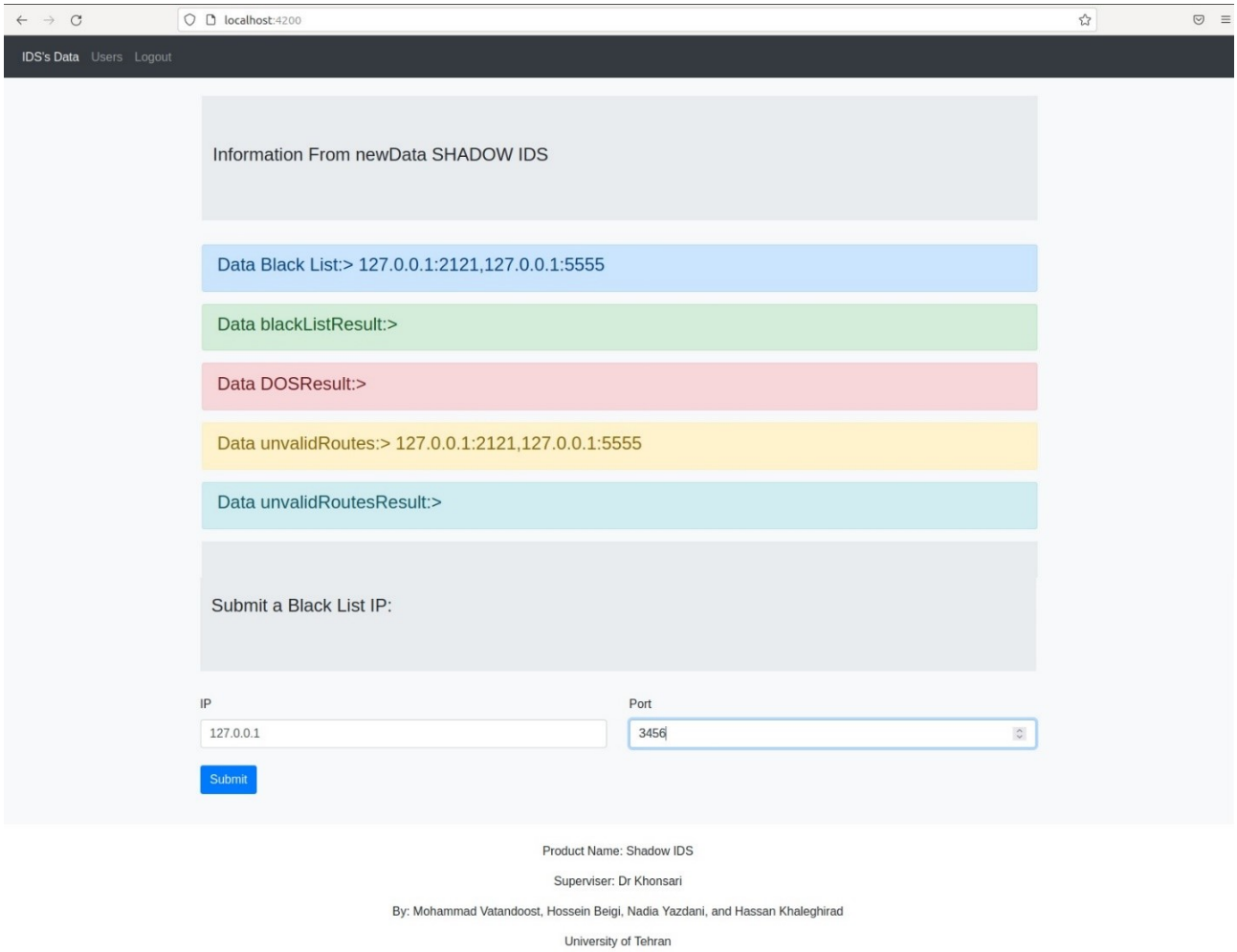
شکل ۱: ارسال ۱۱۵۶۳ packet توسط client.

```
('Client IP Address: ', ('127.0.0.1', 49244))
Received message from client.
('Client IP Address: ', ('127.0.0.1', 49244))
Received message from client.
('Client IP Address: ', ('127.0.0.1', 49244))
Received message from client.
('Client IP Address: ', ('127.0.0.1', 49244))
Received message from client.
('Client IP Address: ', ('127.0.0.1', 49244))
Received message from client.
('Client IP Address: ', ('127.0.0.1', 49244))
Received message from client.
('Client IP Address: ', ('127.0.0.1', 49244))
Received message from client.
('Client IP Address: ', ('127.0.0.1', 49244))
Received message from client.
('Client IP Address: ', ('127.0.0.1', 49244))
Received message from client.
('Client IP Address: ', ('127.0.0.1', 49244))
Received message from client.
('Client IP Address: ', ('127.0.0.1', 49244))
Received message from client.
('Client IP Address: ', ('127.0.0.1', 49244))
Received message from client.
('Client IP Address: ', ('127.0.0.1', 49244))
Received message from client.
('Client IP Address: ', ('127.0.0.1', 49244))
```

شکل ۲: دریافت packet در سرور و نمایش مشخصات client.

برنامه رابط کاربری IDS

برای ارتباط با برنامه IDS که یک برنامه سمت سرور هست نیاز به یک برنامه Client وجود دارد. برای نوشتن برنامهClient از فریم ورک Angular استفاده شده است. این فریم ورک که یک فریم ورک برای برنامه های Front-end است قابلیت های بالایی در نوشتن سرویس ها و همچنین نمایش اطلاعات دریافتی از سرور با استفاده از قالب HTML است. شکل زیر صفحه IDS را نشان می دهد.



مهمترین بخش برنامه رابط کاربری استفاده از دو سرویس برای دریافت اطلاعات از سرور IDS و همچنین ثبت اطلاعات در سرور IDS است.

سرور IDS اطلاعات زیر را برای client فراهم می کند:

- لیست IP هایی که بلاک شده اند. List Black Data
- لیست نتیجه اطلاعات IP های بلاک شده blackListResult Data
- لیست نتایج حملات انکار سرویس: DOSResult Data
- لیست مسیرهایی غیر مجاز IP) های بلاک شده در این لیست هستند) unvalidRoutes Data
- نتایج مسیرهایی غیر مجاز: unvalidRoutesResult Data

اطلاعات فوق با استفاده از یک سرور get از سرور دریافت شده و بر روی صفحه اصلی واسط کاربری نمایش داده می شود.

با استفاده از یک سرویس post زوج اطلاعات <IP, Port> که در یک فرم موجود در واسط کاربری نوشته می شوند با فشردن دکمه ثبت به سرور فرستاده می شود. به محض ثبت اطلاعات در سرور IDS سرویس دریافت اطلاعات دوباره اجرا می شود و اطلاعات نمایش داده شده بر روی صفحه اصلی واسط کاربری بروزرسانی می شود.

ایجاد گراف شبکه

در این قسمت فایل های اکسل حاصل از بخش مانیتورینگ، یک به یک خوانده می شود و برای هرکدام یکبار تابع make_TDG فراخوانی می شود.

```
if __name__ == '__main__':
    path = "../Data/"
    csvData = [f for f in listdir(path) if isfile(join(path, f))]
    for filePath in csvData:
        print("file Path :", filePath)
        tdg = make_TDG(path+filePath)
        nx.draw(tdg,  edge_color='r')
```

تابع make_TDG، گراف شبکه را برای یک بازه زمانی می سازد.

```
def make_TDG(csvFile):
    tdg = nx.DiGraph()
    global node_counter
    with open('packetsData.csv', newline='') as csvfile:
        reader = csv.reader(csvfile, delimiter=' ', quotechar='|')
        i = 0
        for row in reader:
            # skip header row
            if i == 0:
                i = i +1
                continue
            element = row[0].split(",")
            if not (element[12] in nodes):
                nodes[element[12]] = node_counter
                tdg.add_node(node_counter)
                node_counter = node_counter + 1

            if not (element[13] in nodes):
                nodes[element[13]] = node_counter
                tdg.add_node(node_counter)
                node_counter = node_counter + 1

            tdg.add_edge(nodes[element[12]], nodes[element[13]])

    return tdg
```

تشخیص حملات با کمک یکریختی(Isomorphim) در گراف TDG

این بخش با کمک مقاله [۱] انجام‌شده‌است.

همانطور که در بخش قبل بیان‌شد، از روی ترافیک ثبت شده شبکه، یک گراف به نام گراف TDG(Traffic Dispertion Graph) ساخته می‌شود. در این گراف، هر راس نشان‌دهنده یک آدرس IP در شبکه و هر یال نشان‌دهنده یک ارتباط بین دو گره از شبکه می‌باشد.

برای تشخیص حملات موجود در بسته‌های ثبت‌شده از شبکه، در کنار گراف TDG از مفهوم یکریختی(Isomorphism) در گراف‌ها استفاده می‌کنیم. یک یکریختی بین دو گراف G و G^* در واقع یک نگاشت از گره‌های گراف G به گره‌های گراف G^* است به طوریکه برچسب و ساختار یال‌ها حفظ شود. همچنین یکریختی زیرگرافی، به معنای یک نگاشت از گره‌های G به گره‌های یک زیرگراف از G^* است که برچسب و ساختار یال‌ها حفظ شود.

روال کار در این بخش اینگونه است که با شبیه‌سازی حملات شناخته‌شده و ثبت بسته‌های مرتبط با آن‌ها، گراف TDG را برای هر یک از این حملات می‌سازیم. سپس یکریختی گراف حمله با گرافیک ترافیک شبکه بررسی می‌شود. وجود یکریختی زیرگرافی به معنای رخداد حمله موردنظر در شبکه می‌باشد. برای بررسی یکریختی گراف‌های TDG، از تابع isomorphism موجود در کتابخانه networkx استفاده شده‌است. این تابع از الگوریتم VF2 استفاده شده‌است.

```
if __name__ == '__main__':
    print_hi('PyCharm')
    path = "./Attack-Data/"
    traffic_tdg=make_TDG("./traffic.csv");
    attacks=[]
    #nx.draw(traffic_tdg,  edge_color='r')
    #plt.show()
    csvData = [f for f in listdir(path) if isfile(join(path, f))]
    for filePath in csvData:
        print("Cheking Attack file:", filePath)
        tdg = make_TDG(path+filePath)
        nx.draw(tdg,  edge_color='r')
        check_attack = isomorphism.DiGraphMatcher(tdg, traffic_tdg)
        if(check_attack.is_isomorphic()):
            print("     Attack Detected");
            attacks.append(filePath);
        else:
            print("     No Attack");
    print("#"+str(len(attacks))+" attack(S) Detected in the traffic file!")
    print(attacks)
        #plt.show()
```

تشخیص ناهنجاری

ما از دو روش برای تشخیص ناهنجاری در گراف‌های شبکه استفاده کرده‌ایم. روش Kmax، روش Graph Edit Distance.

در روش Kmax ما به دنبال گره‌ای با ماکسیمم درجه هستیم، درجه این گره معیار خوبی برای تعیین وجود حملات Dos در شبکه می‌باشد.

```
#####Checking kmax(max degree)
print("Cheking Kmax:")
node_degree_list=traffic_tdg.degree()
kmax=0
for item in node_degree_list:
    kmax=max(item[1],kmax)
if(kmax>kmax_threshold):
    print("----->Anomaly Detected based on kmax")
    attacks.append("Kmax")
```

در روش Graph Edit Distance، با کمک فرمول زیر، میزان تغییرات گراف شبکه را در دو بازه زمانی متوالی اندازه‌گیری می‌کنیم. در صورتی که این تغییرات بیشتر از حالت معمول باشد، نشان‌دهنده وجود ناهنجاری در شبکه می باشد.

$$d(G_i, G_j) = |V_i| + |V_j| - 2\,|V_i \cap V_j| + |E_i| + |E_j| - 2\,|E_i \cap E_j|$$

```
#####Checking Edit Distance
print("Checking Edit Distance:")
if(prev_traffic_tdg):
    for v in nx.optimize_graph_edit_distance(prev_traffic_tdg,traffic_tdg) :
        minv=v
        break;
    if(minv>edit_distance_threshold):
        print("----->Anomaly Detected based on edit_distance")
        attacks.append("Edit Distance")
```

منابع

[۱] Do Quoc Le et al. Traffic dispersion graph based anomaly detection. In: Proceedings of the Second Symposium on Information and Communication Technology. ،۲۰۱۱ pp. ۳۶.۴۱