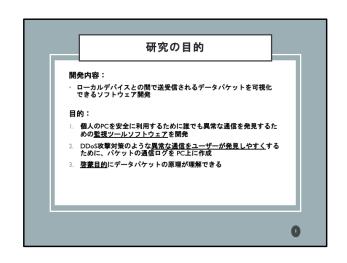


C#を用いたWindows向けのパケット可視化システムの開発と題しまして計算機システム研究室とソフトウェア研究室の合同研究のムハマドが発表します。

どうぞよろしくお願いいたします。



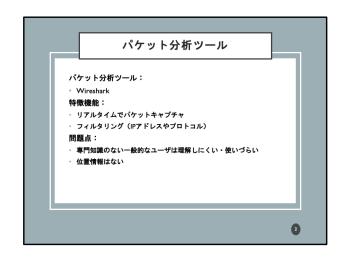
研究内容としては、ローカルデバイスとの間で送受信されるデータパケットを可視化できるソフトウェア を開発します。

この研究の目的としては3つあります。

まず、個人のPCを安全に利用するために誰でも異常な通信を発見するための監視ツールソフトウェアを 開発します。

そして、DDoS攻撃対策のような異常な通信をユーサ゛ーか゛発見しやすくするために、パケットの通信ログを PC上に作成するソフトウェアを開発する目的があります。

最後には、啓蒙(けいもう)目的にデータパケットの原理が理解できる、学べるソフトウェアを開発します。



パケットの送受信状況を見る手段としてはWiresharkのようなパケット分析ツールがあります。

Wiresharkはリアルタイムでパケットキャプチャ機能があり、それに対するフィルタリングも対応します。

ただし、Wiresharkの場合は、専門知識のない一般的なユーザは理解しにくいし、使いづらいシステムであります。

そして、位置情報も対応しないという問題点があります。

		パク	ーツ	ト分析ツール
•••				Capturing from Wi-Fit end
		৭ 🗢 🗢 🧰 র	<u> •</u>	■
No. Time	Source	Destination	Profescel L	
1473 50.967678 1474 50.967888	192,168,11,6	192,168,11,7	TCP	78 64348 - 49687 [PSH, ACK] Seq=9 Ack=1 Win=4896 Len=4 TSval=1439231888 TSecr=1234281763 66 49687 - 64348 [ACK] Seq=1 Ack=13 Win=2848 Len=8 TSval=1234216676 TSecr=1439231888
1475 51.674831	192, 568, 11, 7	172,217,26,13#	TOP	54 [TCP Dup ACK 348#1] 58936 - 443 [ACK] Seg=1 Ack=1 Win=2848 Len=8
1475 51,788863 1477 51,889125	172,217,25,110 fe88;;1ccd;71a1;e4	192,168,11,7	TLSv1	66 [TCP Dup ACK 340#1] [TCP ACKed unseen segment] 443 = 50036 [ACK] Seg=1 Ack=2 Min=270 Len=0 TS 265 Acel:(atlon Sota
1478 51.089239	fe88::481:4008:796	fe80::1ccd:71a1:e4	TCP	86 51861 - 49911 [AOX] Seq=1750 Ack=1921 Win=130752 Len=0 TSusl=1234217595 TSecr=1439232000
1479 51.889578 1488 51.889682	fe88::481:4568:796 fe88::481:4568:796	fe80::1ccd:7la1:e4_	TLS+1	356 Application Data 86 51861 - 49911 [FZN, ACK] Seg-2020 Ack-1921 Win-131872 Lenn® TSval-1238217595 TSecr-1439232000
1481 51.892425	fe80::1ccd:71a1:e4.	fe80:1401:4558:796	TCP	86 49911 - 51861 [AOK] Seq-1921 Ack-2020 Win-138784 Len-8 TS-s1-1439222889 TSecr-1234217595
1482 51.892433 1483 52.634628	fe88::1ccd:71a1:e4_ Micre-St_82:78:3d	fe80::481:466:796 Broadcast	TCP	86 49911 - 51861 (ADX) Seq=1921 Ack=2021 Win=130784 Len=0 T5wal=1439232809 T5ecr=1234217595 60 Who has 192.168.11.17 Tell 192.168.11.2
1484 54.583752	49.12.156.198	192.168.11.7	TLSv1	98 Application Data
1485 54.583925 1486 54.583953	192,168,11,7 192,168,11,7	49.12.156.198	TCP TLSv1-	66 62641 - 443 [ACK] Seq=30 Ack=49 Win=2048 Len=0 TSval=1234220287 TSecr=3924582000 94 Application Data
1487 54.682312	Micro-St_82:78:34	Broadcast	ARP	68 Who has 192.168.11.17 Tell 192.168.11.2
1488 54.887498 1489 54.976253	Buffale_64:c4:60 192:160:11:7	Broadcast 216,239,32,136	ARP	42 Who has 192,168,11.97 Tell 192,168,11.1 54 [TCP Kero-Alive] 53943 = 643 [6CK] Sep-647 Arks793 Winn333972 Lenn®
1490 54.983773	49.12.156.198	192.168.11.7	TCP	66 443 - 62641 [ACK] Seq=49 Ack=58 WIn=787 Len=8 TSval=3924583224 TSecr=3234228287
1492 55.000426 1492 55.251488	216.239.32.116 192.168.11.7	192.168.11.7 35.190.00.1	TOP	66 [TCP Keep-Alive ACK] 443 = 51843 [ACK] Seq=793 Ack=648 Win=66816 Len=8 TSval=2725328415 TSec= 54 [TCP Keep-Alive] 51842 = 443 [ACK] Seq=2818 Ack=1183 Win=131872 Len=8
1493 55.272222	33-198-88-1	192-168-11-7	TOP	66 TTCF Earp-Alive ACK1 443 - 51842 [ACK] Sep-1103 Acks/2015 Nine/2000 TSval-2420016105 TSva
> Ethernet II, Src: > Internet Protocol > Transmission Conte > Transport Layer Se	Apple_97:72:2e (f0:38: Version 6, Src: fe80:: ol Pretecel, Src Pert:	98:97:72:2e), Dst: 16 401:4068:796d:cd9d, E 51049, Dst Port: 499	:b8:69:d6: st: fe80::	
att ab 97 81 2e 9 2d b 68 79 6d c 71 a1 e4 ac d 61 86 80 18 80 62 67 55 68 af 80 80 82 18 2 77 9 91 52 80 7 77 77 78 80 59 80 80 80 82 82 62 80 80 82 83 62 80 80 82 83 85 80 80 82 83 85 80 80 82 83 85 80 80 80 82 85 80 80 80 80 80 80 80 80	5 49 fe 59 60 60 60 60 6 6 5d fe 50 60 60 60 6 6 73 c7 69 c2 f6 61 8 60 cc 77 60 60 61 6 8 60 cc 77 60 60 61 6 6 67 17 63 63 61 69 6 6 22 21 6a a6 3a 59 5 2 56 2f d2 2a 65 7d 8 61 56 c3 57 69 7c 2	## 00 00 04 01 0 0 0 0 0 0 0 0 0 0 0 0 0 0		
	apture in progresso			Packets 1994 - Displayed 1994 (199.0%) Profile Default

これは起動中のWiresharkの例です。

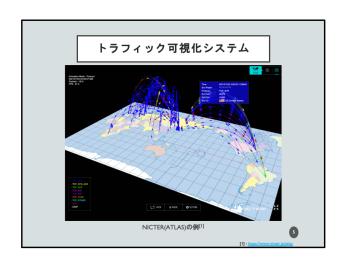
ご覧になっている通り、本システムはわかりにくいシステムであります。

トラフィック可視化システム	
トラフィック可視化システムの研究: - NICTER -> 情報通信研究機構(NICT)の研究プロジェクト -> 具体的なシステム: -> ATLAS -> DAEDALUS	
	0

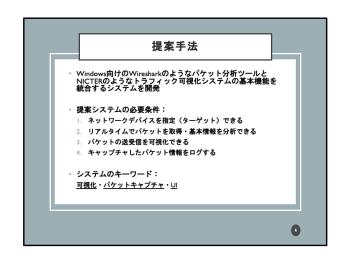
トラフィック可視化システムを紹介しますと、日本内だとNICTERという研究プロジェクトが一般的です。

NICTERに関係する具体的なシステムはATLASとDAEDALUSがあります。

これらのシステムは、プロトコルの自動解析しながら、日本国内のパケット通信のトラフィック可視化できる特徴があります。



例としてNICTERのATLASを表示します。



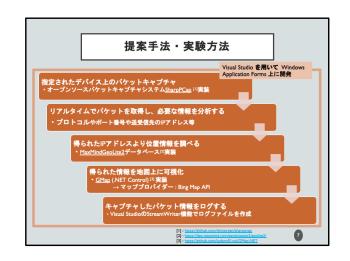
本研究の提案手法に移りますと、Windows向けのWiresharkのようなパケット分析ツールとNICTERのようなトラフィック可視化システムの基本機能を統合するシステムを開発します。

このシステムを開発するために、必要条件は4つあります。

まずは、

- 1. ネットワークデバイスを指定(ターゲット)できる
- 2. リアルタイムでパケットを取得・基本情報を分析できる
- 3. パケットの送受信を可視化できる
- 4. キャップチャしたパケット情報をログする

本システムのキーワードとしては可視化・パケットキャプチャとUIとなります。



実験方法はこのようになります。

まず、指定されたデバイス上のパケットをキャプチャし、リアルタイムでパケットを取得したら必要な情報を分析します。

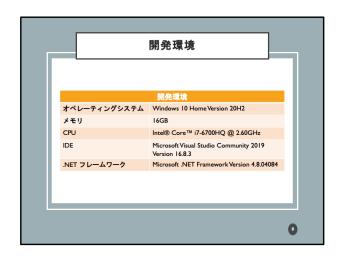
この部分はオープンソースパケットキャプチャシステムのSharpPCapを利用しています。

そして、得られたIPアドレスより、MaxMindGeoLite2データベースを使って位置情報を調べます。 得られた位置情報を地図上に可視化をします。可視化を担当するのはGmapの.NET Controlを実装しています。

なお、今回のマッププロバイダーはBing Map APIを使用しています。 最後に、Visual StudioのStreamWriter機能でキャプチャしたパケット情報をログします。

この流れではシステムの基本アルゴリズムと考えられます。

これらはVisual Studioを用いてWindows Application Forms 上に開発します。



開発環境はこのようになっています。



結果としては、このようなシステムが開発しました。

なお、本研究ではこのシステムはPacVisという名前をつけます。

PacVisを説明すると、3つの部分に分けることができます。



まず、デバイスを管理する部分です。

この部分では3つのポイントがあります。

なお、一番大事なポイントはポイント1となります。

ポイント1は「Check Device(s)」ボタンを押すとネットワークデバイスをリストで表示され、リストからパケットキャプチャするデバイスを選択できる部分は提案システムの必要条件1が達成します。



次、パケットキャプチャの分析と情報表示の部分です。

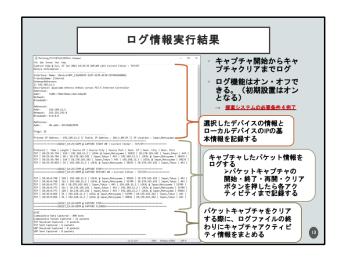
ここでは、リアルタイムでパケットの情報をカラーコード付きリスト型で表示され、提案システムの必要 条件2を完了します。



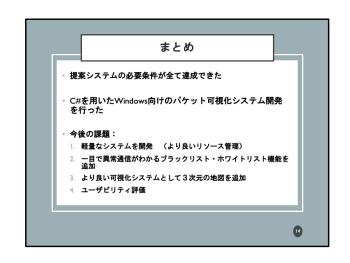
最後に、PacVisの下半分のところで可視化の部分があります。

ご覧になる通り、ポイント4はパケット通信に従って「最新件」順にTCPパケットの送受信をリアルタイムで可視化できます。

これで提案システム必要条件3が達成しまた。



キャプチャ開始からキャプチャクリアまで全てのアクティビティがログします。 このログ機能で提案システムの必要条件4が完了できました。 なお、ログ機能は設定でオン・オフできます。



まとめて、提案システムの必要条件が全て達成でき、C#を用いたWindows向けのパケット可視化システム開発が行いました。

今後の課題としては、より良いリソース管理ができる軽量なシステムに改善できるかという課題があります。

そして、ブラックリスト・ホワイトリスト機能で一目で異常通信がわかるよう、より良い可視化システムとして3次元の地図も追加できたらさらにいいUIができればと思います。

最後に、今回は新型コロナウイルスでユーザビリティ評価ができず、ユーザビリティ評価を実施することも今後の課題になります。

以上で発表終わります。ご静聴ありがとうございました。

補足:UDPの可視化問題

- UDPはTCPと違って、プライベート・リザーブIPアドレスが 非常に多い。
- 可視化にすると、ゲオロケーション情報を必要だが、ブライベート・リザーブIPアドレスはデータベースを調べてもゲオロケーション情報を取得できない。
- ゲオロケーションが取得できなければ、システムはエラーと 表示されてしまう。
- この問題より、UDPを可視化にすると、例外条件を多数追加 必要で、安定なシステムを開発できたと判断するため、スタ ビリティテストをしなければならない。(時間の問題)
- 加えて、角例外を調べると、アルゴリズムの複雑度が高めて しまって、処理時間が増える可能性がある。

15

