TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN – ĐHQG-HCM
Khoa Mạng máy tính & Truyền thông

Đánh lừa IDS và các biện pháp đối phó

NT204 – Hệ thống tìm kiếm, phát hiện và ngăn ngừa xâm nhập

GV: Đỗ Hoàng Hiển

hiendh@uit.edu.vn

Các kỹ thuật đánh lừa IDS

- **Keyword:** Evading IDS
- **Đánh lừa IDS - IDS evasion** là quá trình **thay đổi các tấn công** để **đánh lừa IDS** nhận diện thành các traffic hợp lệ và ngăn IDS tạo ra các cảnh báo

1 Insertion Attack	7 Unicode Evasion	13 Polymorphic Shellcode
2 Evasion	8 Fragmentation Attack	14 ASCII Shellcode
3 Denial-of-Service Attack	9 Overlapping Fragments	15 Application-Layer Attacks
4 Obfuscating	10 Time-To-Live Attacks	16 Desynchronization
5 False Positive Generation	11 Invalid RST Packets	17 Encryption
6 Session Splicing	12 Urgency Flag	18 Flooding

Đánh lừa IDS

01. Insertion Attack

- Là quá trình kẻ tấn công **lừa IDS đọc các gói tin không hợp lệ**
- Xảy ra khi: NIDS **có chính sách bảo vệ không chặt chẽ** bằng hệ thống nội bộ
  - IDS có thể chấp nhận những gói tin mà các hệ thống đầu cuối sẽ loại bỏ
- Tấn công nhằm thêm dữ liệu vào thông tin được đọc bởi IDS
- IDS nhận được **nhiều gói tin** hơn so với hệ thống đầu cuối

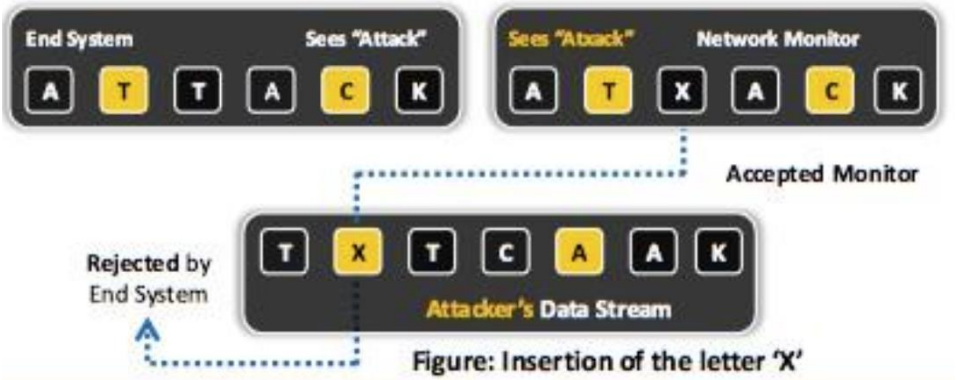


Figure: Insertion of the letter 'X'

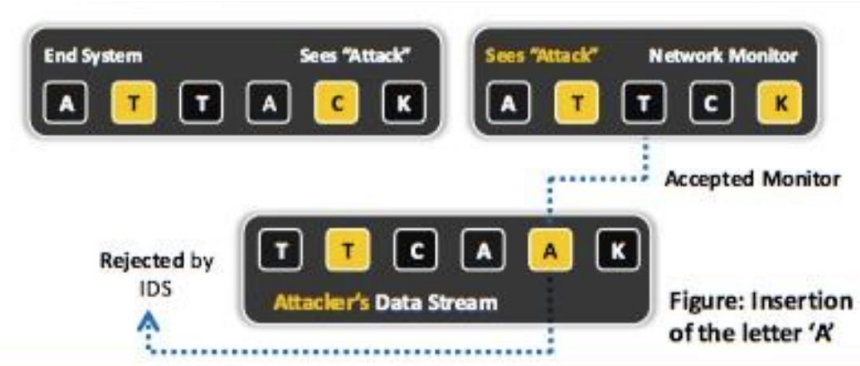
Ví dụ 1: Ký tự X chỉ được IDS chấp nhận

*Ví dụ 2: IDS nhận diện chuỗi **phf** trong HTTP*

- Thông thường:
  GET /cgi-bin/**phf**? → **OK**
- Kẻ tấn công gửi:
  GET /cgi-bin/**p**これは**h**攻撃**f**?
  → **??**

02. Evasion

- Ở kỹ thuật này, **hệ thống đầu cuối chấp nhận gói tin IDS chặn**
- Kẻ tấn công có thể tấn công máy tính mà IDS không nhận ra
- Kẻ tấn công gửi **các phần của yêu cầu tấn công** trong các gói tin mà IDS sẽ chặn sai, cho phép bỏ đi 1 số phần của stream dưới góc nhìn của IDS
- IDS nhận **ít gói tin hơn** so với hệ thống đầu cuối



Ví dụ:

- *Một tấn công gửi **byte-by-byte**, và có một byte bị loại bỏ bởi IDS -> IDS không thể phát hiện tấn công này*
- *Ở đây, **IDS nhận ít gói tin hơn** hệ thống đầu cuối*

Ví dụ 1: IDS sẽ lọc 1 ký tự A

Insertion attack và evasion attack

Các trường hợp thực tế

- Khai thác các vấn đề trong NIDS về phân tích mạng và các giao thức cơ bản
  - Phân tích các trường header
  - Xử lý các option trong header
  - Việc lắp ráp lại các fragment/segment
- NIDS và hệ thống đầu cuối nhìn thấy các stream dữ liệu khác nhau
  - NIDS cho phép/chặn traffic khác so với hệ thống đầu cuối
- Khác biệt trong hiện thực các giao thức
  - Không theo chuẩn giao thức
  - Các OS có thể khác nhau trong protocol stack
- Cấu hình end system và router
- Các option ở mức ứng dụng/socket

Insertion attack và evasion attack

Ví dụ - Sự nhập nhằng trong NIDS

Trường liên quan	Sự nhập nhằng (vấn đề) cho NIDS
TTL	Gói tin đến hệ thống đầu cuối trước khi TTL bằng 0?
Length, DF	Tất cả các downstream link có thể truyền gói tin lớn mà không phân mảnh nó (không bật cờ DF)?
IP Option(s)	Hệ thống đầu cuối/router có chấp nhận các gói tin có thêm trường IP Option(s)?
TCP Option(s)	Hệ thống đầu cuối có chấp nhận gói tin có thêm trường TCP Option(s)?
Data	Hệ thống đầu cuối có chấp nhận dữ liệu trong gói SYN?
ToS	Gói tin có tuân thủ tất cả các router nội bộ (DiffServ)?
IP Frag Offset	Hệ thống đầu cuối có lắp ráp các fragment chồng chéo nhau?
TCP Seq No	Hệ thống đầu cuối có lắp ráp các segment chồng chéo nhau?

03. Denial-of-Service Attack (DoS)

- Nhiều IDS sử dụng **server lưu trữ alert tập trung**
- Kẻ tấn công có thể **thực hiện DoS** hoặc các tấn công khác vào server này để **làm chậm hoặc crash server**
- Khi đó, việc tấn công sau đó **sẽ không bị ghi log**

- **Với kỹ thuật qua mặt này, kẻ tấn công có thể:**
  1. Khiến 1 thiết bị bị khoá
  2. Khiến cá nhân không thể điều tra tất cả các cảnh báo
  3. Tạo ra nhiều cảnh báo cần được các hệ thống quản lý xử lý
  4. Làm đầy không gian ổ cứng để không thể ghi log các tấn công
  5. Tiêu tốn tài nguyên xử lý của thiết bị và từ đó có thể bỏ sót tấn công

04. False Positive Generation

1. Kẻ tấn công có hiểu biết về IDS có thể **tạo các gói tin độc hại** chỉ để tạo ra các cảnh báo

2. Những gói tin này được gửi đến IDS để tạo ra **một lượng lớn các cảnh báo false positive**

3. Sau đó, kẻ tấn công lợi dụng những cảnh báo false positive này để **che giấu traffic tấn công thực sự**

4. Kẻ tấn công có thể bypass qua IDS vì **rất khó để phân biệt traffic tấn công** với số lượng lớn các false positive

Đánh lừa IDS

DoS và False Positive Generation

- Vấn đề
  - NIDS cần mô phỏng được hoạt động của các thiết bị đầu cuối và mạng nội bộ
  - Giới hạn về tài nguyên (CPU, bộ nhớ, bộ lưu trữ, băng thông)
  - NIDS thường hoạt động ở trạng thái “fail-open” – khi bị crash nó vẫn cho phép traffic đi qua để đảm bảo mạng vẫn hoạt động
- Một số dạng tấn công:
  - **DoS vào CPU** *nhắm tới các hoạt động tiêu tốn tài nguyên tính toán (*lắp ráp các fragment/segment, mã hoá/giải mã…)
  - **DoS vào bộ nhớ** *nhắm tới các hoạt động quản lý trạng thái (*TCP 3-way handshake, lắp ráp các fragment/segment…)
  - **DoS băng thông mạng** *nhắm tới khiến NIDS không xử lý được gói tin đang truyền*
  - **DoS hệ thống phản ứng**: tạo nhiều cảnh báo (false positive), chặn truy cập hợp lệ bằng cách giả mạo địa chỉ, che giấu tấn công thực

05. Obfuscating – Làm rối mã

- Trong kỹ thuật làm rối mã, kẻ tấn công **encode payload tấn công** ở dạng khác để chỉ có host đích decode được còn IDS thì không
- Một số ví dụ về kỹ thuật rối mã:

1. Kẻ tấn công có thể thay đổi **đường dẫn được tham chiếu trong signature** để đánh lừa HIDS

2. Kẻ tấn công có thể **encode pattern tấn công trong Unicode** để bypass bộ lọc của IDS, nhưng các Web server vẫn hiểu được

3. **Code đa hình** có thể qua mặt các signature-based IDS bằng cách tạo ra các pattern tấn công khác biệt, khi đó sẽ không có signature nào có thể phát hiện được tấn công

4. Các tấn công trong **các giao thức được mã hoá** như HTTPS cũng là 1 dạng làm rối mã

Ví dụ: Obfuscation – Rối mã

Thử dự đoán đoạn code JS thực hiện công việc gì?

```
var
_0x5823=['log','bind','warn','625336fEAUme','table','exception','1133956sREoHn','prototype','1bLfSRk','in
fo','2MuEkXR','__proto__','length','trace','toString','146844WbdIMk','354245ZvPPNt','1bcnHtc','return\x20
(function()\x20','986831plMbcm','Hello\x20IDPS\x20Class!','3dbYhsT','1Hqobww','1708103qdTKCs','514751TNpg
PW','constructor','console','{}.constructor(\x22return\x20this\x22)(\x20)'];(function(_0x40a260,_0x1026f2
){var _0x2ebee1=_0x2a5b;while(!![]){try{var
_0x285eb0=parseInt(_0x2ebee1(0x190))*parseInt(_0x2ebee1(0x1a7))+parseInt(_0x2ebee1(0x1a6))*parseInt(_0x2e
bee1(0x1a1))+parseInt(_0x2ebee1(0x19d))+-parseInt(_0x2ebee1(0x191))*parseInt(_0x2ebee1(0x193))+-
parseInt(_0x2ebee1(0x19a))+parseInt(_0x2ebee1(0x1aa))*parseInt(_0x2ebee1(0x19f))+parseInt(_0x2ebee1(0x192
))*-parseInt(_0x2ebee1(0x1a8));if(_0x285eb0===_0x1026f2)break;else
_0x40a260['push'](_0x40a260['shift']());}catch(_0x1810c1){_0x40a260['push'](_0x40a260['shift']());}}}(_0x
5823,0x9991c));function hi(){var _0x113adf=_0x2a5b,_0xf3cbe6=function(){var _0x433b1d=!![];return
function(_0x5c1b55,_0x579058){var _0x2e7687=_0x433b1d?function(){if(_0x579058){var
_0x5f02e3=_0x579058['apply'](_0x5c1b55,arguments);return _0x579058=null,_0x5f02e3;}}:function(){};return
_0x433b1d=![],_0x2e7687;};}(),_0x4c6d16=_0xf3cbe6(this,function(){var _0x6ef7b6=_0x2a5b,_0x18fdfd;try{var
_0x47e64b=Function(_0x6ef7b6(0x1a9)+_0x6ef7b6(0x196)+');');_0x18fdfd=_0x47e64b();}catch(_0x1c15a5){_0x18f
dfd=window;}var
_0x3d659e=_0x18fdfd[_0x6ef7b6(0x195)]=_0x18fdfd[_0x6ef7b6(0x195)]||{},_0x58330b=[_0x6ef7b6(0x197),_0x6ef7
b6(0x199),_0x6ef7b6(0x1a0),'error',_0x6ef7b6(0x19c),_0x6ef7b6(0x19b),_0x6ef7b6(0x1a4)];for(var
_0x463254=0x0;_0x463254<_0x58330b[_0x6ef7b6(0x1a3)];_0x463254++){var
_0x7f7e41=_0xf3cbe6[_0x6ef7b6(0x194)][_0x6ef7b6(0x19e)][_0x6ef7b6(0x198)](_0xf3cbe6),_0x4fbffa=_0x58330b[
_0x463254],_0x3af806=_0x3d659e[_0x4fbffa]||_0x7f7e41;_0x7f7e41[_0x6ef7b6(0x1a2)]=_0xf3cbe6['bind'](_0xf3c
be6),_0x7f7e41[_0x6ef7b6(0x1a5)]=_0x3af806['toString'][_0x6ef7b6(0x198)](_0x3af806),_0x3d659e[_0x4fbffa]=
_0x7f7e41;}});_0x4c6d16(),alert(_0x113adf(0x1ab));}function
_0x2a5b(_0x116c69,_0x2c3ca5){_0x116c69=_0x116c69-0x190;var _0x380e01=_0x5823[_0x116c69];return
_0x380e01;}hi();
```

Ví dụ: Obfuscation – Rối mã

Thử dự đoán đoạn code JS thực hiện công việc gì?

https://obfuscator.io/

```
var
_0x5823=['log','bind','warn','625336fEAUme','table','exception','1133956sREoHn','prototype','1bLfSRk','in
fo','2MuEkXR','__proto__','length','trace','toString','146844WbdIMk','354245ZvPPNt','1bcnHtc','return\x20
(function()\x20','986831plMbcm','Hello\x20IDPS\x20Class!','3dbYhsT','1Hqobww','1708103qdTKCs','514751TNpg
PW','constructor','console','{}.constructor(\x22return\x20this\x22)(\x20)'];(function(_0x40a260,_0x1026f2
){var _0x2ebee1=_0x2a5b;while(!![]){try{var
_0x285eb0=parseInt(_0x2ebee1(0x190))*parseInt(_0x2ebee1(0x1a7))+parseInt(_0x2ebee1(0x1a6))*parseInt(_0x2e
bee1(0x1a1))+parseInt(
parseInt(_0x2ebee1(0x1
))*-parseInt(_0x2ebee1
_0x40a260['push'](_0x4
5823,0x9991c));functic
function(_0x5c1b55,_0x
_0x5f02e3=_0x579058['a
_0x433b1d=![],_0x2e768
_0x47e64b=Function(_0x
dfd=window;}var
_0x3d659e=_0x18fdfd[_0x6ef7b6(0x195)]=_0x18fdfd[_0x6ef7b6(0x195)]||{},_0x58330b=[_0x6ef7b6(0x197),_0x6ef7
b6(0x199),_0x6ef7b6(0x1a0),'error',_0x6ef7b6(0x19c),_0x6ef7b6(0x19b),_0x6ef7b6(0x1a4)];for(var
_0x463254=0x0;_0x463254<_0x58330b[_0x6ef7b6(0x1a3)];_0x463254++){var
_0x7f7e41=_0xf3cbe6[_0x6ef7b6(0x194)][_0x6ef7b6(0x19e)][_0x6ef7b6(0x198)](_0xf3cbe6),_0x4fbffa=_0x58330b[
_0x463254],_0x3af806=_0x3d659e[_0x4fbffa]||_0x7f7e41;_0x7f7e41[_0x6ef7b6(0x1a2)]=_0xf3cbe6['bind'](_0xf3c
be6),_0x7f7e41[_0x6ef7b6(0x1a5)]=_0x3af806['toString'][_0x6ef7b6(0x198)](_0x3af806),_0x3d659e[_0x4fbffa]=
_0x7f7e41;}});_0x4c6d16(),alert(_0x113adf(0x1ab));}function
_0x2a5b(_0x116c69,_0x2c3ca5){_0x116c69=_0x116c69-0x190;var _0x380e01=_0x5823[_0x116c69];return
_0x380e01;}hi();
```

```
1 function hi() {
2   alert("Hello IDPS Class!");
3 }
4 hi();
```

06. Session Splicing

- Kẻ tấn công **chia nhỏ traffic tấn công** thành nhiều gói tin để không 1 gói nào bật cảnh báo trên IDS
- Hiệu quả:
  - Có thể qua mặt các IDS **không lắp ráp lại các gói tin** trước khi kiểm tra chúng với các signature tấn công
  - Thêm delay giữa các gói tin được gửi để qua mặt quá trình lắp ráp gói tin trên IDS
    - Một số IDS **ngừng lắp ráp** nếu không nhận các gói sau 1 khoảng thời gian
    - Một số IDS ngừng hoạt động nếu có các session active lâu hơn thời gian lắp ráp
  - Các tấn công session splicing thành công sẽ **không bị ghi log** trên IDS
- Một số công cụ có thể thực hiện tấn công này: **Nessus**, **Whisker**

06. Session Splicing (tt)

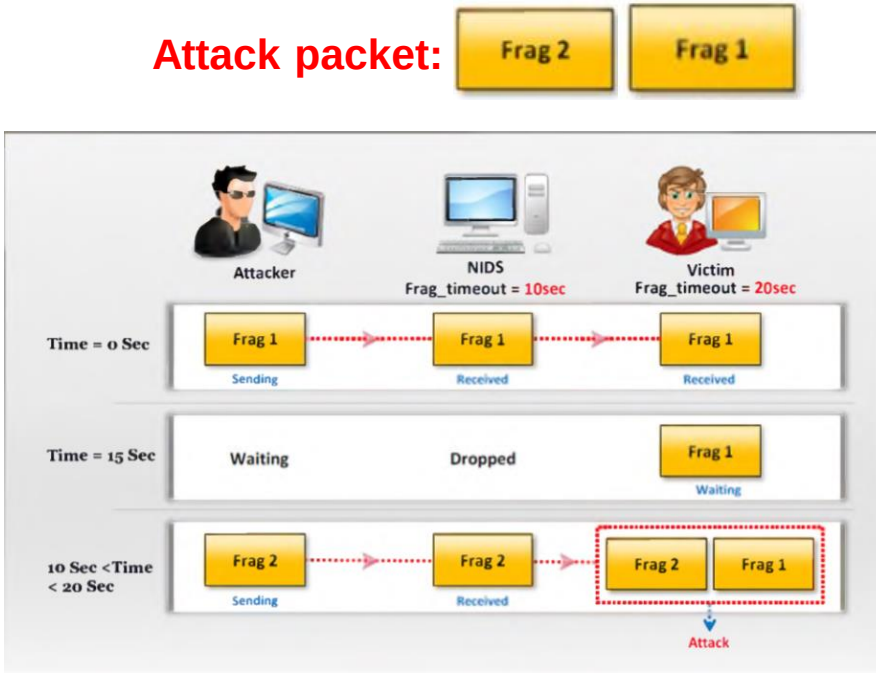
- Session Splicing với Whisker
  - Tấn công ở mức Network
  - Khác với IP fragmentation
  - Gửi các phần của request trong các gói tin khác nhau
    - "GET / HTTP/1.1" sẽ được chia thành nhiều gói tin thành dạng

    "GE", "T ", "/", " H", "T", "TP", "/1", ".1"

- **IP Fragmentations:** Gói tin quá lớn đối với tầng data link, router có thể chia nhỏ thành nhiều fragment
- **Session Splicing:** Truyền payload qua nhiều gói tin **một cách có mục đích** để đánh lừa việc phát hiện tấn công - kích thước nhỏ hơn cần thiết
- Biện pháp đối phó
  - Fragment reassembly
  - Session reassembly
  - Send a reset [RST]

07. Fragmentation Attack

- Fragmentation – Hoạt động phân mảnh có thể dùng để tấn công khi IDS và host có **thời gian timeout cho việc phân mảnh khác nhau**
- **Ví dụ 1:** Timeout ở host là **20s** > timeout ở IDS là **10s.** Giả sử kẻ tấn công gửi các fragment cách nhau **15s?**
  - Sau 15s, IDS sẽ không lắp ráp các fragment tiếp theo do đã vượt quá thời gian timeout, còn host khi nhận vẫn lắp ráp tiếp các fragment này
  - Kẻ tấn công tiếp tục gửi các fragment sau 15s delay đến khi toàn bộ payload đã được nhận và lắp ráp ở host

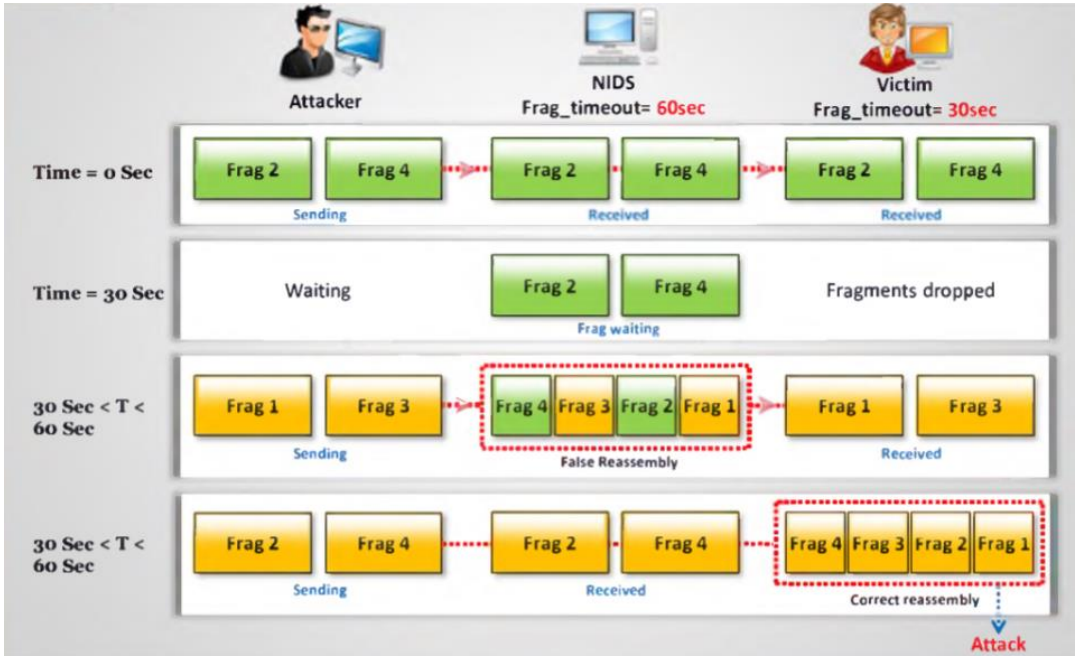


07. Fragmentation Attack (tt)

Attack packet: 

- **Ví dụ 2:** Timeout ở host là **30s** < timeout ở IDS là **60s**

- Cả host và IDS nhận frag 2 và 4 trên tổng 4 frag; frag 2, 4 có payload sai
- Sau 30s, host bỏ 2 frag này và không gửi ICMP vì không nhận được frag 1
- Host và IDS nhận tiếp frag 1 và 3
- IDS cố lắp ráp 4 frag, nhưng checksum sẽ không hợp lệ nên gói tin bị bỏ
- Host và IDS nhận frag 2 và 4 thật trước khi thời gian fragment ở host time out



- Victim lắp ráp 4 frag thành payload tấn công, còn IDS timeout và drop 2 frag

08. Overlapping Fragments

- Overlapping Fragment – Phân mảnh chồng chéo: **tạo ra loạt các fragment nhỏ** với các **sequence number chồng chéo** nhau
  - Ví dụ: fragment đầu gồm 100 bytes payload với seq = 1, fragment thứ 2 có seq bị chồng có giá trị 96, v.v…
- Khi **lắp ráp lại gói tin**, host đích cần biết cách lắp ráp các TCP fragment bị chồng chéo
  - Một số OS như Windows W2K/XP/2003: giữ fragment gốc khi trùng seq
  - Một số OS như Cisco IOS sẽ chọn các fragment đến sau có cùng seq

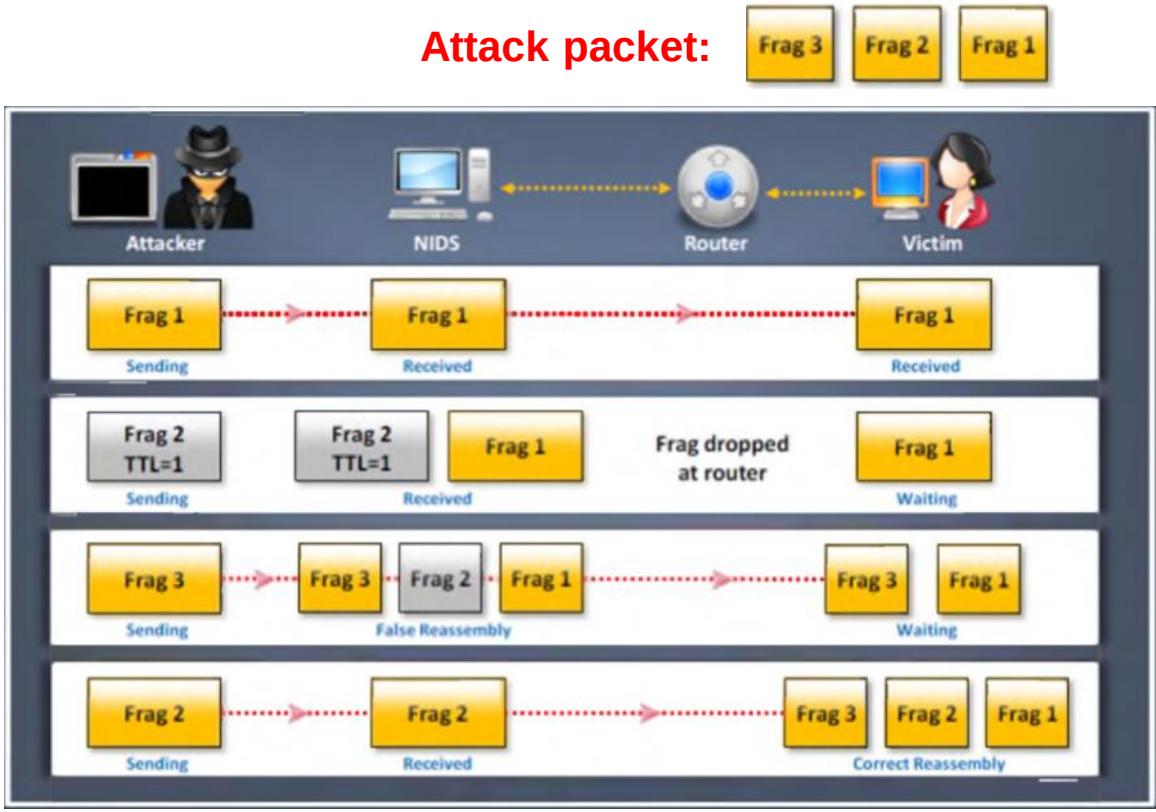
Đánh lừa IDS

09. Time-To-Live Attacks

- Attacker cần có **hiểu biết trước về topology** của mạng nạn nhân.
  - Sử dụng công cụ như **traceroute** để biết **số router nằm giữa kẻ tấn công và nạn nhân**.

- Các bước tấn công:

1. Attacker chia traffic thành **3 fragments**
2. Kẻ tấn công gửi **frag 1 với TTL cao**, frag 2 **giả** với TTL thấp
3. IDS nhận cả 2 fragment, nhưng victim chỉ nhận **fragment đầu**
4. Kẻ tấn công gửi tiếp **frag 3 với TTL cao**
5. IDS lắp ráp 3 fragment thành 1 gói tin vô nghĩa và drop
6. Victim sau đó nhận **frag 2** thật, và lắp ráp lại thành traffic tấn công, khi đó tấn công diễn ra nhưng không bị ghi log



10. Invalid RST Packets

- Tấn công này dựa trên **gói RST** và giá trị **checksum** của nó:
  - Cờ **reset (RST)** trong trường TCP được dùng để đóng 1 kết nối TCP
  - TCP sử dụng trường checksum 16 bit để **kiểm tra lỗi** trong header và dữ liệu
- Cách tấn công:
  - Trong tấn công này, attacker **gửi gói RST** đến victim với giá trị **checksum không hợp lệ**
  - IDS khi nhận gói RST nghĩ là **TCP session đã kết thúc** → **IDS** ngưng xử lý gói tin liên quan mặc dù victim vẫn sẽ nhận gói tin
  - Victim **kiểm tra checksum trong gói RST** và drop nó
  - Tấn công này cho phép **attacker giao tiếp** với victim trong khi IDS nghĩ rằng kết nối giữa 2 bên đã kết thúc

11. Urgency Flag

- Cờ Urgent (URG) trong TCP header được dùng để đánh dấu dữ liệu cần **xử lý khẩn cấp** tại host đích
- Nếu cờ URG được bật, giao thức TCP gán trường Urgent Pointer thành **giá trị offset 16 bit** trỏ đến byte cuối của dữ liệu khẩn cấp trong segment
- Nhiều IDS **không xem xét urgent pointer** và xử lý tất cả gói tin trong traffic trong khi victim chỉ xử lý dữ liệu khẩn cấp
- Điều này có thể khiến IDS và victim **có tập gói tin khác nhau**, có thể bị khai thác bởi attacker để gửi traffic tấn công
- Theo RFC 1122, urgency pointer sẽ khiến 1 byte dữ liệu nằm ngay sau dữ liệu khẩn cấp bị mất trong trường hợp kết hợp dữ liệu khẩn cấp và dữ liệu thường

Ví dụ:

```
Packet 1: ABC
Packet 2: DEF Urgency Pointer: 3
Packet 3: GHI
Kết quả cuối cùng: ABCDEFHI
```

Đánh lừa IDS

12. Unicode Evasion

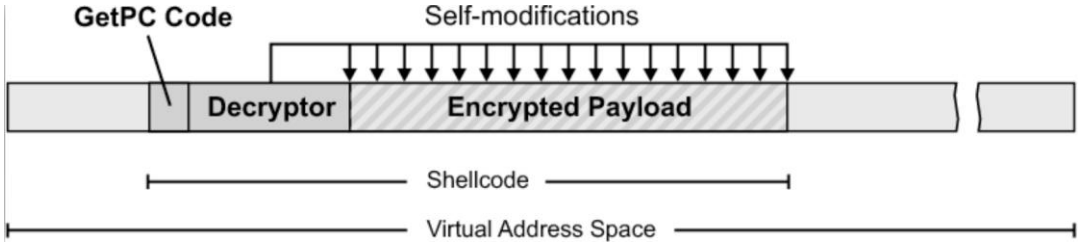
- Unicode là một **hệ thống mã hoá ký tự** nhằm hỗ trợ trao đổi, xử lý và hiển thị văn bản
  - Ví dụ: / → %u2215, e → %u00e9
- Trong mã Unicode, có thể có **nhiều dạng biểu diễn khác nhau của cùng 1 ký tự**
  - Ví dụ: \ có thể biểu diễn 5C, C19C, E0819C
  - Một số IDS **có thể xử lý Unicode không chính xác**
- Attacker có thể **chuyển chuỗi tấn công sang dạng Unicode** để qua mặt các phép so khớp pattern hay signature trên IDS
- Attacker có thể **encode URL** trong request HTTP với các ký tự Unicode để **bypass** phát hiện các tấn công web của IDS

https://unicode-table.com/en/

https://www.branah.com/unicode-converter

13. Polymorphic Shellcode – Đa hình

- Nhiều IDS dựa trên signature sử dụng signature là những **string thường được dùng** trong các shellcode để phát hiện shellcode
- Tấn công shellcode đa hình bao gồm nhiều **string shell code không cố định** để gây khó khăn cho việc định nghĩa signature để có thể phát hiện tấn công
- Attacker **encode payload** với các kỹ thuật và đặt 1 decoder ở phía trước payload → shellcode **được viết lại hoàn toàn** mỗi lần nó được gửi đi để đánh lừa việc phát hiện
- Kỹ thuật này cũng **qua mặt các string shellcode hay dùng**, do đó, nó khiến các signature shellcode trở nên vô dụng.



http://shell-storm.org/shellcode/

14. ASCII Shellcode

Assembly	Mã máy	ASCII
`pop eax`	\x58	X
`xor al, 58`	\x34\x58	4X

- ASCII shellcode chứa các ký tự được biểu diễn trong **chuẩn ASCII**
- Attacker sử dụng ASCII shellcode để qua mặt các signature của IDS vì việc **so khớp pattern** không hoạt động hiệu quả với các giá trị ASCII
- Phạm vi của ASCII code **có giới hạn** vì không phải tất cả các lệnh assembly đều có thể chuyển trực tiếp sang các giá trị ASCII
- Giới hạn này có thể khắc phục bằng cách sử dụng **tập các lệnh** khác để có thể chuyển sang dạng ASCII

https://nets.ec/Ascii_shellcode

The following is an ASCII shellcode example:

```
char shellcode[] =
"LLLLYhb0pLX5b0pLHSSPPWQPPaPWSUTBRDJfh5t
DS"
"RajYX0Dka0TkafhN9fYf1Lkb0TkdjfY0Lkf0Tkg
fh"
"6rfYf1Lki0tkkh95h8Y1LkmjpY0Lkq0tkrh2wnu
X1"
"Dks0tkwjfX0Dkx0tkx0tkyCjnY0LkzC0TkzCCjt
X0"
"DkzC0tkzCj3X0Dkz0TkzC0tkzChjG3IY1LkzCCC
C0"
"tkzChpfcMX1DkzCCCC0tkzCh4pCnY1Lkz1TkzCC
CC"
"fhJGfXf1Dkzf1tkzCCjHX0DkzCCCCjvY0LkzCCC
jd"
"X0DkzC0TkzCjWX0Dkz0TkzCjdX0DkzCjXY0Lkz0
tk"
"zMdgvvn9F1r8F55h8pG9wnuvjrNfrVx2LGkG3ID
pf"
"cM2KgmnJGgbinYshdvD9d";
```

Đoạn ASCII shellcode trên sẽ chạy 1 shell **/bin/sh**. Các chuỗi **bin** và **sh** nằm ở những byte cuối của shell code

15. Application-Layer Attacks

- Các ứng dụng truy cập các file đa phương tiện (audio, video và hình ảnh) thường **nén** chúng thành các kích thước nhỏ hơn để tối đa hoá tốc độ truyền
- IDS không thể phát hiện **signature của các dạng file đã nén**
- Điều này có thể cho phép attacker **khai thác lỗ hổng** trong dữ liệu bị nén

16. Desynchronization

- Có thể xảy ra trước và sau kết nối TCP
- **Pre-Connection SYN**
  - Attacker gửi 1 gói **SYN trước khi tạo kết nối thực sự**, nhưng với checksum không hợp lệ
  - Nếu một gói SYN được nhận **sau khi TCP Control Block được mở**, IDS sẽ thiết lập lại sequence number cho phù hợp để khớp với số thứ tự của gói SYN mới nhận.
  - Attacker gửi các **gói SYN giả mạo** với các sequence number không hợp lệ để làm IDS không còn đồng bộ (desynchronize the IDS)
  - Điều này **ngăn IDS** theo dõi tất cả các traffic, cả traffic hợp lệ và traffic tấn công

16. Desynchronization (tt)

- **Post-Connection SYN**
  - Kỹ thuật này cố gắng **làm IDS không đồng bộ** từ sequence number thực tế mà kernel đang sử dụng
  - Gửi một **gói SYN sau kết nối** trong luồng dữ liệu, gói tin này sẽ có các **số thứ tự khác nhau**
  - Tuy nhiên, máy đích sẽ **bỏ qua gói SYN** này, vì nó đang tham chiếu đến một kết nối đã được thiết lập
  - Mục đích của cuộc tấn công này là để **IDS đồng bộ lại sequence number** của nó với gói SYN mới
  - Sau đó, IDS sẽ bỏ qua mọi dữ liệu, là **một phần hợp pháp của luồng dữ liệu gốc**, vì nó sẽ đang chờ một sequence number khác
  - Sau khi thành công trong việc đồng bộ lại IDS với gói SYN, **gói RST với sequence number mới** được gửi và đóng kết nối của nó

Các dạng đánh lừa khác

Mã hoá

Khi attacker đã thiết lập được **một session được mã hoá với victim**, đó là **tấn công đánh lừa IDS hiệu quả nhất**

Flooding

Attacker gửi một loạt **các traffic dư thừa để tạo ra traffic nhiễu**, và nếu IDS không thể xử lý tốt traffic nhiễu thì traffic tấn công thực sự có thể sẽ không phát hiện được

Các công cụ đánh lừa IDS/Firewall

Công cụ **Traffic IQ Professional** cho phép **theo dõi và kiểm tra** hoạt động của các **thiết bị bảo mật** bằng cách tạo ra các **traffic ứng dụng chuẩn và traffic tấn công** giữa 2 VM

http://www.idappcom.com

Một số công cụ khác

Hotspot Shield
https://www.hospotshield.com

FTester
https://inversepath.com

Snare Agent for Windows
https://www.intersectalliance.com

Tomahawk
http://tomahawk.sourceforge.net

Atelier Web Firewall Tester
http://www.atelierweb.com

Freenet
https://freenetproject.org

Proxifier
https://www.proxifier.com

Biện pháp đối phó với Tấn công Đánh lừa IDS

Làm sao để ngăn tấn công đánh lừa IDS?

1. Đóng các **switch port** có liên quan đến các host tấn công đã biết
2. **Phân tích theo chiều sâu** các traffic mạng khả nghi để phát hiện các nguy cơ tấn công
3. Sử dụng gói **TCP FIN** hoặc **RST** để đóng các session TCP tấn công
4. Tìm kiếm các **nop opcode** khác ngoài 0x90 để ngăn các tấn công shellcode đa hình
5. Đào tạo user cách xác định **pattern tấn công** và **cập nhật/vá** các hệ thống và thiết bị mạng
6. **Triển khai IDS** sau khi phân tích mạng, đặc điểm của traffic mạng và số lượng host cần giám sát
7. Sử dụng **traffic normalizer** để loại bỏ các phần đáng ngờ trong các gói tin trước khi đến IDS
8. Đảm bảo IDS chuẩn hoá được **các gói tin đã phân mảnh** và cho phép lắp ráp chúng đúng thứ tự
9. **Định nghĩa DNS server** cho client trên router hoặc các thiết bị mạng tương tự
10. **Củng cố an ninh** của tất cả các thiết bị giao tiếp như modem, router, switch,...
11. Nên chặn các gói **ICMP TTL expired** ở external interface và thay đổi trường TTL thành 1 giá trị lớn
12. Thường xuyên cập nhật cơ sở dữ liệu **signature của antivirus**
13. Sử dụng **giải pháp chuẩn hoá traffic** tại IDS để tránh bị đánh lừa
14. Lưu trữ **các thông tin tấn công** (IP attacker, IP victim, thời gian) để phân tích trong tương lai

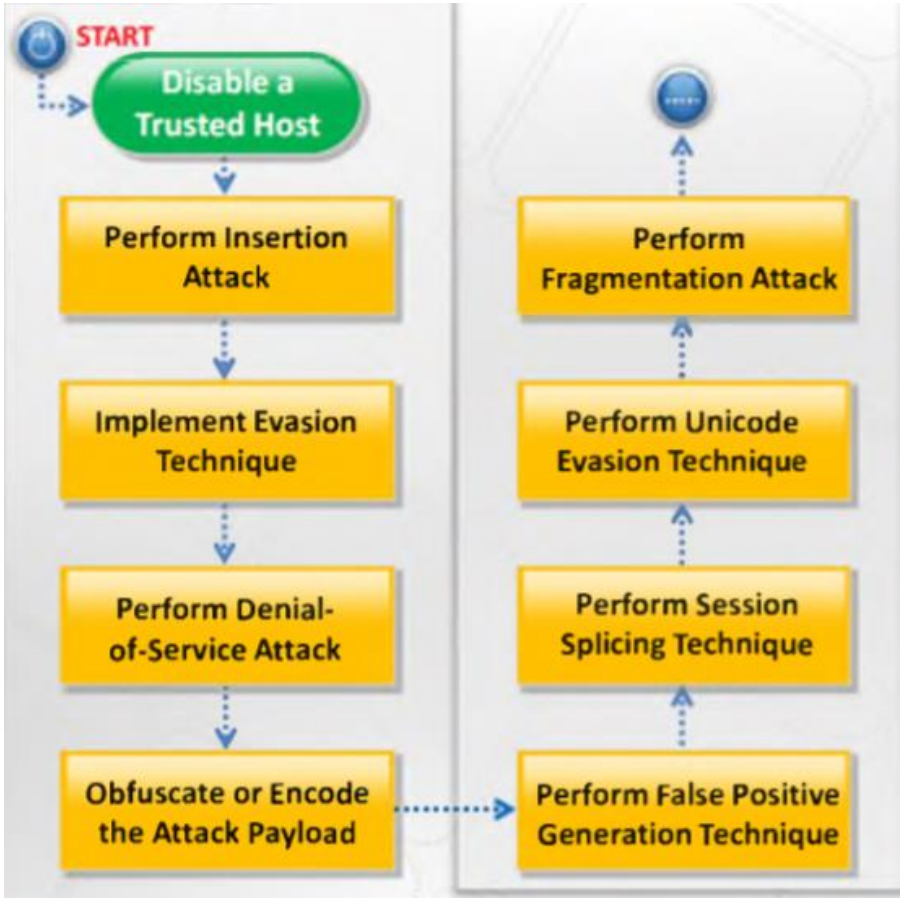
Đánh giá IDS

IDS Penetration Testing

- Pentest IDS giúp đánh giá **khả năng lọc traffic ra và vào mạng**
- **Vì sao cần Pentest IDS?**
  - Kiểm ra IDS có đảm bảo thi hành **các policy IDS của tổ chức** hay không
  - Kiểm tra IDS có đảm bảo thi hành **các policy bảo mật mạng** hay không
  - Kiểm tra IDS có đủ tốt để **ngăn chặn các tấn công từ bên ngoài**
  - Kiểm tra hiệu quả của **vành đai bảo vệ mạng**
  - Kiểm tra **lượng thông tin mạng có thể truy cập được** từ phía kẻ xâm nhập
  - Kiểm tra **các nguy cơ bảo mật có** trên IDS có thể bị khai thác
  - Đánh giá **tương quan giữa các rule IDS** và các hành động được thực hiện bởi chúng
  - Kiểm tra policy bảo mật có **được thi hành chính xác** thông qua chuỗi các rule IDS hay không

Đánh giá IDS

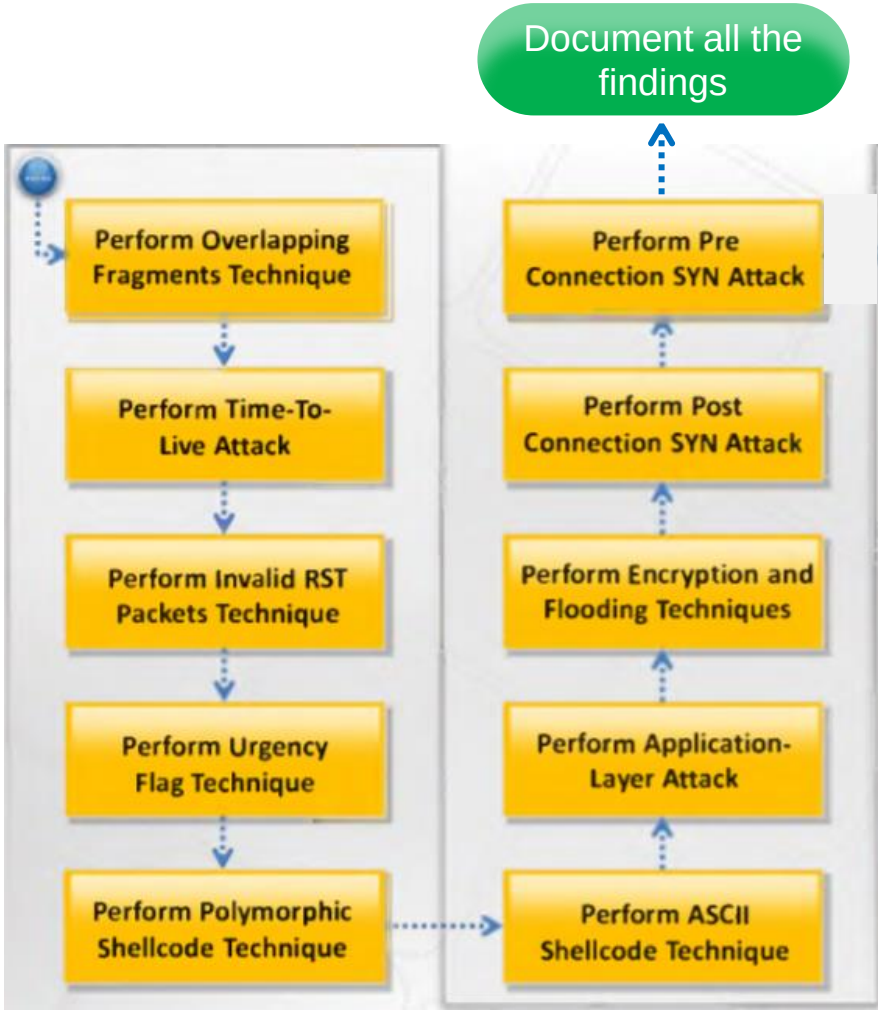
IDS Penetration Testing



- **Bước 1:** Vô hiệu hoá Trusted Host
- **Bước 2:** Thực hiện Insertion Attack
- **Bước 3:** Thực hiện kỹ thuật Evasion
- **Bước 4:** Thực hiện tấn công DoS
- **Bước 5:** Làm rối mã hoặc encode payload tấn công: Ví dụ thực hiện encode gói tin tấn công để IDS không nhận diện được nhưng IIS web server vẫn có thể decode và bị tấn công
- **Bước 6**: Dùng kỹ thuật Tạo False Positive: thử bypass IDS bằng cách giấu traffic tấn công trong 1 số lượng lớn **các cảnh báo false positive**
- **Bước 7**: Sử dụng **kỹ thuật session splicing**, ví dụ để bypass IDS bằng cách giữ session active lâu hơn thời gian lắp ráp gói tin của IDS
- **Bước 8**: Sử dụng kỹ thuật đánh lừa với Unicode, với **định dạng Unicode** của các ký tự để qua mặt các signature của IDS
- **Bước 9**: Thực hiện **tấn công fragmentation** với thời gian timeout cho việc lắp ráp gói của IDS **nhỏ hơn** và **lớn hơn** thời gian của victim

Đánh giá IDS

IDS Penetration Testing (tt)



- **Bước 10:** Thực hiện kỹ thuật chồng chéo fragment: tạo nhiều fragement có sequence number chồng chéo
- **Bước 11:** Thực hiện tấn công Time-To-Live
- **Bước 12:** Thực hiện kỹ thuật gói tin RST không hợp lệ để ngăn IDS theo dõi các gói tin
- **Bước 13:** Thực hiện kỹ thuật cờ URG để đánh giá IDS vì 1 số IDS không xem xét urgent pointer
- **Bước 14:** Thực hiện kỹ thuật shellcode đa hình, ví dụ thử mã hoá shellcode để qua mặt IDS
- **Bước 15**: Thực hiện kỹ thuật ASCII shellcode: giấu shellcode trong các ký tự ASCII để qua mặt các signature của IDS
- **Bước 16**: Thực hiện tấn công ở tầng Application để xem khả năng xử lý dữ liệu bị nén
- **Bước 17**: Thực hiện mã hoá và flooding
- **Bước 18**: Thực hiện tấn công Post-connection SYN
- **Bước 19**: Thực hiện tấn công Pre-connection SYN
- **Bước 20**: Tài liệu hoá tất cả các thông tin thu thập được

Nhắc lại…

Tổng kết

- Kẻ tấn công có thể đánh lừa IDS với nhiều kỹ thuật đánh lừa khác nhau:

1 Insertion Attack	7 Unicode Evasion	13 Polymorphic Shellcode
2 Evasion	8 Fragmentation Attack	14 ASCII Shellcode
3 Denial-of-Service Attack	9 Overlapping Fragments	15 Application-Layer Attacks
4 Obfuscating	10 Time-To-Live Attacks	16 Desynchronization
5 False Positive Generation	11 Invalid RST Packets	17 Encryption
6 Session Splicing	12 Urgency Flag	18 Flooding

- IDS penetration testing giúp đánh giá và tuỳ chỉnh khả năng lọc traffic ra và vào của IDS

Chuẩn bị cho tuần tới…

- Hôm nay: **Đánh lừa IDS và Các biện pháp đối phó**
- Chuẩn bị cho tuần sau: **Lecture 09 – 10: IDS dựa trên AI và IDS cho AI**
  - Hiểu cơ bản về ML/DL
  - Xem trước tài liệu:

Câu hỏi/thắc mắc (nếu có)?

Today end, **See you next week!**