Chào các bạn, ở box này bác conmale đang có 1 topic rất hay: Ký sự các vụ DOS HVA. Trong quá trình phân tích, bác có đưa ra khá nhiều khái niệm mới mẻ về TCP/IP, protocol stack… Việc nghiên cứu kỹ TCP/IP là yêu cầu bắt buộc đối với những người muốn đi sâu vào lĩnh vực networking. Các vị tiền bối đi trước đã bảo với tớ rằng: “Tất cả những khó khăn em gặp đều có thể giải quyết được nếu như em nắm vững TCP/IP và hiểu rõ bản chất của nó”. Các bạn có thể kiểm chứng câu nói này qua các bài viết của bác conmale.   
Topic này không phải là để dạy dỗ, hay là tutorial gì. Các vấn đề về TCP/IP có lẽ các bạn nên tự tìm hiểu, nghiên cứu. Chủ đề này tớ sẽ post các câu hỏi, các vấn đề còn vướng mắc và khó hiểu trong quá trình tớ tìm hiểu TCP/IP. Và hi vọng là các bạn sẽ tham gia nhiệt tình.   
  
Hôm nay tớ xin khai trương bằng mục Checksum, bao gồm: IP Checksum và Checksum của các protocol khác(TCP, UDP, ICMP…). Dưới đây tớ sẽ trình bày cách hiểu của tớ về 2 loại checksum này, nếu như tớ hiểu sai thì ai đó correct giùm cái nhá.   
  
Như bạn đã biết, ở mỗi layer trong TCP/IP model người ta đều xây dựng một cơ chế kiểm sóat lỗi, mục đích là để kiểm tra xem packet có bị thay đổi trong quá trình truyền tin hay không?(tớ xin bỏ qua Ethernet checksum nhá :p).   
Bạn xài tcpdump hay Ethereal gì đó, capture 1 UDP packet chẳng hạn. Rùi, nhảy zô IP Header, sẽ dòm thấy 1 field là: Checksum. Field này có kích thước là: 16bits.   
  
  
Cách tính IP checksum:   
  
  
Đầu tiên là ở phía sender. Bạn xem IP Header như là một chuỗi các bit 0, 1. Rùi, bây giờ chia cái chuỗi này ra theo từng “đọan”, mỗi đọan dài 16 bits.   
Ví dụ: tớ có 1 chuỗi 0,1 thế này: 0110011001100110 0101010101010101 0000111100001111   
Thực hiện phép cộng(nhị phân) ba từ này lại với nhau, được kết quả là: A = 100101011001010   
Okie, bây giờ lấy phần bù của chuỗi A(cách lấy phần bù, chuyển bit 0 thành bit 1, bit 1 thành bit 0) ta được chuỗi B như sau: 0011010100110101. gửi nóàOkie, đem cái chuỗi này bỏ vào field Checksum của IP datagram đi.   
  
Ở phía receiver:   
  
  
Thằng receiver sau khi nhận được datagram này sẽ tính lại checksum theo cách y như trên(khi tính tổng sẽ có 1 số hạng là Checksum do sender tính). Nếu như được kết quả là 16 bits 1 thì Header của IP datagram này không bị thay đổi gì so với khi vứtà gói tin bị lỗi àgửi đi. Còn nếu như có 1 bit nào đấy bằng 0 luôn, không báo gì về cho thằng sender biết cả .   
Như vậy, bạn có thể thấy là mục đích của IP Checksum là kiểm tra xem Header của IP datagram có bị lỗi trong quá trình truyền hay không ?(khi tính toán ta chỉ quan tâm đến Header, không ngó ngàng gì tới phần data cả )   
  
Chú ý:   
  
  
có thể sẽ có IP Header nếu có thêm các thông tin trong phần Option trường hợp độ dài của IP Header sẽ không chia hết cho 16(ta cần chia thành các chuỗi 16 bits mà). Trong trường hợp này, thằng IP layer sẽ tự thêm vào các pad byte(tất cả các bit đều set bằng 0) để làm sao kích thước của IP Header đủ chia hết cho 16.   
Đối với Checksum của các protocol khác như TCP, UDP, ICMP thì có khác. Sự khác biệt ở đây là: Khi Reiceiveràtính checksum sẽ bao gồm luôn cả phần data trong gói tin đó có thể phát hiện được lỗi trong phần data(cái này là hơn thằng IP Checksum đây)   
  
Lấy ví dụ UDP Checksum:   
  
  
UDP Checksum bao gồm: pseudo-header, UDP Header , data và pad-byte(nếu cần thiết)   
  
Pseudo-header là gì?   
  
  
Nó gồm một số field lấy từ IP header lên: bao gồm 32-bit source IP address, 32-bit destination address, 8 unused-bit(8 bit 0), 8-bit protocol, 16-bit UDP length.   
  
Mục đích của pseudo-header để làm gì?   
Phần này em mong các bác chỉ giáo, em xin trình bày cách hiểu của em. 2 cái trường IP là để check lại xem gói tin này có đến đúng đích hay chưa? Trường Protocol là để kiểm tra lại xem thằng IP layer nó phân phát gói tin lên đúng protocol hay chưa? (Cái này ông Richard bảo là double-check)   
Cách tính UDP Checksum thì cũng tương tự như trên. Nếu như Receiver tính ra checksum mà có 1 bit 0 thì gói tin đã bị lỗi trên thằngàđường truyền. Nếu như Checksum nhận được tòan là bit 0 cả sender đã không tính Checksum trước khi gửi đi.   
  
Chú ý:   
- UDP Checksum là không bắt buộc nhưng đừng dại gì mà disable nó   
- TCP Checksum là bắt buộc phải có   
  
Hix, dài quá, nhưng đây là những gì em hiểu về Checksum. Em àxin các bác bổ sung ý kiến và liên hệ các trường hợp thực tế(nếu có) cái này thì em còn kém lắm. À còn nữa, em muốn hỏi trong trường hợp UDP Checksum. Packet sang đến IP layer của thằng Receiver rùi, thế khi thằng Receiver tính lại checksum vẫn phải thêm thằng Pseudo-header vào nữa à ?   
  
  
  
Khi gói tin được chuyển lên layer phía trên IP layer và được demultiplexed thì ở tầng này phải lo chuyện gởi lại gói tin bị lỗi checksum (restransmision), tùy theo ứng dụng của các giao thức.   
  
Cả TCP và UDP đều dùng pseudo-header (đầu giả ) và dùng chung một thuật toán để tính checksum.   
  
  
  
  
QUOTE   
  
  
Hix, dài quá, nhưng đây là những gì em hiểu về Checksum. Em xin các bác bổ sung ý kiến và liên hệ các trường hợp thực tế(nếu có) à cái này thì em còn kém lắm. À còn nữa, em muốn hỏi trong trường hợp UDP Checksum. Packet sang đến IP layer của thằng Receiver rùi, thế khi thằng Receiver tính lại checksum vẫn phải thêm thằng Pseudo-header vào nữa à ?   
  
  
  
  
  
  
Không, UDP checksum là "end-to-end" checksum. UDP check được thằng sender tính, nhưng lại được thằng receiver xác nhận. Cho nên không có bước thằng "receiver tính lại checksum". Có 3 chuyện xảy ra:   
- nếu checksum được gởi đi là 0 ==> thằng gởi đã không tính checksum, không có chuyện gì đáng nói.   
- nếu checksum được gởi đi là 1 hết ==> thằng gởi đã tính checksum, checksum ok, không có chuyện gì làm thêm   
- nếu checksum được gởi đi không phải là 0 (thằng gởi đã tính checksum) nhưng có giá trị 0 trong 16 bits này ==> checksum error, thằng nhận chỉ cần biết thế này là đủ và nó lặng lẽ ném gói tin ấy vào thùng rác.   
  
Checksum là phương tiện để đo lường tính trung thực của gói tin và cái gì cũng có cái giá của nó cả, càng xác định tính trung thực thì càng tạo ra "overhead" --> càng chậm. Sở dĩ TCP bắt buộc phải có checksum vì nó muốn bảo đảm gói tin gởi nhận hoàn toàn trung thực. Nếu có sự cố, nó gởi lại lần nữa, và lần nữa nếu cần, cho đến khi đầu nhận nhận được gói tin trung thực. Bởi vậy, TCP "chắc ăn" hơn nhưng chậm hơn UDP rất nhiều ở điểm này. Trong môi trường lý tưởng như một nội mạng có thiết bị hoàn chỉnh, có topology vững vàng, kết cấu hợp lý thì vấn đề checksum có lẽ ít cần thiết. Tuy nhiên networking ngày nay không còn biên giới hạn hẹp nữa nên cái "môi trường lý tưởng" này càng ngày càng trở nên không thể được. Đây là lý do tại sao IP header của IPv6 không còn checksum field nữa.   
  
  
  
  
QUOTE   
  
  
Hum nì đọc wa cái Evasion attack, hè hè, bít thêm được 1 cái nữa. Thằng NIDS không cóa cơ chế tính checksum -> gặp evasion attack gần như là bó tay, chặn không hết nổi . Kiểu này lại phải xài đến HIDS xem ra mới ngăn chặn được phần nào bọn Evasion attack nì nhỉ?   
  
  
  
  
  
snort lo được chuyện evasion. Tuy nhiên phải thiết kế signature cho "chiến", không thì rất dễ bị false positive.   
  
Ngoài ra trên application layers thì có nhiều tools khác có thể thể giúp hạn chế evasion. Ví dụ, mod\_security cho apache thực hiện chuyện này khá ngon và độc hơn nữa là nó có thể dùng snort signatures (đã được chuyển hóa) để trị "evasive".   
  
  
  
QUOTE   
  
  
Hum nì cóa câu hỏi tiếp ạ. Vẫn là các field trong IP Header   
greencool.gif (có chỗ nào hỏi hết ) greenbiggrin.gif .   
Câu hỏi của em xoay quanh trường Type-of-service(TOS). Trong IP Header TOS có độ dài là 8 bits   
- 3 bit đầu tiên là 3 bit ưu tiên(precedence). Em còn nhiều chỗ thắc mắc ở 3 bits này đây.   
Các câu hỏi:   
+ Giá trị của từng bit thế nào ? Ý nghĩa là gì ?   
  
- 1 unused bit: cái này không xài, để dành tương lai cần thì xài( greenbiggrin.gif )   
- 4 bits TOS   
+ Minimize delay(ưu tiên độ trễ)   
+ Maximize througput (ưu tiên thông lượng)   
+ Maximize reliability(ưu tiên độ tin cậy)   
+ Minimize monetary cost --> bit này thì em chịu, không hiểu nó để làm gì, trong cái table của Richard có đưa ra thằng NNTP nó set bit này bằng 1, nhưng em vẫn chưa hiểu được ý nghĩa của nó ???   
  
  
  
  
  
Tự hỏi rồi tự giải thích hết rồi còn trả lời gì nữa?   
  
"Minimize delay" --> giảm thiểu độ chậm trễ.   
  
"Minimize monetary cost" --> "giảm thiểu phí tổn". Có những dịch vụ không quan trọng về tính khẩn cấp, có chậm một tí cũng chẳng sao, có chiếm ít băng thông của không ảnh hưởng gì ví dụ như NNTP. Protocol NNTP này là Usenet News và tin tức có vào news client chậm một tí cũng không sao. Nói chung những dịch vụ được xếp loại "thứ yếu" thì có thể dùng TOS là "mmc".   
  
Lý do tại sao protocol như telnet lại được khuyến khích dùng "md"? tại vì cần bảo đảm những diễn tiến xảy ra trên remote machine được cấp báo ngay trên local machine để tạo tính "real time interaction".   
  
Lý do tại sao protocol như ftp cho data chẳng hạn lại được khuyến khích dùng "mt"? tại vì cần phải đẩy dữ liệu đi càng nhanh càng tốt để kết thúc giai đoạn tải dữ liệu trong thời gian ngắn nhất.   
  
Nói chung TOS dùng theo nhu cầu nhất định nào đó.   
  
  
  
  
QUOTE   
  
  
Ở bên trên là ý nghĩa(+ thắc mắc greenbiggrin.gif ) của từng bit . Bây giờ em gom lại về TOS field luôn.   
  
Ý nghĩa của TOS field là gì?   
Theo em được biết, hùi xưa mạng Internet nó hơi yếu greenbiggrin.gif. Router không thể xử lý 1 lúc quá nhiều packet --> phải dòm vào cái TOS field để xác định độ ưu tiên, xử lý thằng nào trước thằng nào sau ??? (Không biết có gì sai không ạ ?)   
  
  
  
  
  
Đúng nhưng thiếu một tí. TOS được thiết kế nhưng thực tế ứng dụng không hoàn toàn đi theo đường lối chung. Có quá nhiều OS và quá nhiều ứng dụng không tuân thủ theo RFC (M$ là một trong những điển hình của việc không tuân thủ RFC). Hơn nữa, packets cần phải đi xuyên qua routers chớ không chỉ quanh quẩn trong một subnet và nhiều router không ứng dụng TOS thì cũng bó tay. Lý do tại sao router không muốn ứng dụng TOS? bởi vì nó tạo nên sự phức tạp đáng kể cho việc tạo nhiều routing tables thích ứng. Thêm một lý do khác là tùy routing protocol nào được dùng nữa; trong một mạng mình có thể hoàn toàn quản lý, mình có thể chọn lựa loại routing protocol một cách đồng bộ thì chuyện tạo những routing tables đáp ứng tuy mất thời gian nhưng vẫn có thể làm được. Nếu packets đi xuyên qua nhiều network khác nhau và mỗi router có một policy khác nhau, dùng loại routing protocol khác nhau thì... càng bó tay.   
  
Router, hay nói chính xác hơn là routing protocol có nhiều loại, nhiều đặc tính cũng như hạn chế khác nhau. Không phải loại routing protocol nào cũng "thèm" để ý đến TOS và cũng không phải routing protocol nào cũng "không thể xử lý một lúc quá nhiều packets". Trên thực tế, các ứng dụng cho routing rất phức tạp (và càng ngày càng phức tạp hơn) trong việc xử lý (chọn những factors để quyết định cho routing). Thật ra các routing protocol "hiện đại" ngày nay càng ngày càng "thông minh" trong việc xử lý routing và đưa vào nhiều yếu tố cho việc quyết định routing chớ không chỉ đơn giản "dòm" cái TOS và... "đẩy" packet đi. Nếu bồ thích, nên "ngâm" thêm các routing protocols thì sẽ thấy.   
  
  
  
  
QUOTE   
  
  
Tại sao ngày nay người ta lại không dùng TOS field này nữa ? Hình như là chỉ có một số ứng dụng đặc biệt vẫn xài cái này(em chỉ biết qua loa đến thế chứ chưa biết là thằng nào xài greenbiggrin.gif) Em vác Ethereal capture thử cũng không "chộp" được IP packet nào có TOS field này cả ??? Nó thay bằng cái mới rồi(cái mới này thì em chưa tìm hiểu thêm)   
  
  
  
  
  
Tại sao ngày nay người ta lại không dùng TOS field này nữa? Có chớ, TOS vẫn được dùng trong các thủ thuật traffic shaping. Tuy nhiên, chúng chỉ có giá trị trong giới hạn mình có thể quản lý được. Trong phạm vi cho phép, bồ vẫn có thể dùng TOS để tạo các chế độ routing theo nhu cầu vào theo ưu tiên mình chọn. Ví dụ, bồ có 2 cổng đi ra Internet, một cổng băng thông lớn, một cổng băng thông nhỏ. Bồ có thể dùng TOS để xây dựng routing tables để ép các loại trafìc không cần nhanh đi xuyên qua đường băng thông nhỏ như SMTP, NTP chẳng hạn và các protocol khác như HTTP, FTP đi qua đường băng thông lớn. Hơn thế nữa, bồ có thể "ép" traffic đi từ mỗi host có giới hạn nhất định nào đó để tránh tình trạng nghẽn tắt.   
  
Bồ dùng Ethereal để capture thử traffic trong LAN hay traffic từ Internet? Nếu traffic đi từ Internet thì hầu như bồ chỉ thấy có giá trị TOS là 0x0 mà thôi vì hầu hết các routers không ứng dụng TOS vì quá mất thời gian. Nếu bồ sniff trong LAN và các traffic này không được ấn định cụ thể giá trị TOS thì lúc nào bồ cũng chỉ có thể thấy TOS là 0.   
  
Thay bằng cái mới?. Chừng nào IPv4 còn được dùng thì tính chất của nền tảng giao thức của nó vẫn là như vậy. Ngày nay băng thông càng ngày càng cải thiện nên chuyện TOS không còn được đặt nặng nữa. Trong giới nghiên cứu mạng có hai nhánh chính:   
- hỗ trợ cho quan điểm áp đặt TOS ở protocol level   
- hỗ trợ cho quan điểm xem nhẹ TOS mà chỉ chú trọng khối lượng băng thông để phục vụ   
  
Đám thứ nhất cho rằng protocol không nên bị phụ thuộc vào chất lượng băng thông và các nhà sản xuất phải tuân thủ theo quy định, đám này là đám idealist. Đám thứ nhì thì cho rằng việc phát triển băng thông sẽ giải quyết vấn đề chất lượng tiêu dùng nhanh chóng và hiệu quả hơn là đi ngược lại nền tảng TOS của protocol, đám này là đám realist. Cả hai đám đều có cái lý của họ vì họ xét vấn đề từ hai hướng khác nhau.   
  
  
  
  
QUOTE   
  
  
Không biết là có hỏi nhiều quá không ? greenbiggrin.gif   
  
  
  
  
  
Hì hì, lấy cái "cân" cân thử hỏi thế là nhiều hay ít

- Theo em được biết thì tính checksum của UDP bao gồm cả phần header và phần data. Người ta sẽ coi cả segment như là một chuỗi các từ 16 bit. Đầu tiên người ta tính tổng các từ này lại ( phép cộng nhị phân) sau đó lấy phần bù của nó và ghi giá trị này vào trường UDP checksum ở bên gửi. Ví dụ như là:  
Một UDP datagram có 16bits Source port, 16 bits Destionation port, 16 bits UDP length, 16 bits Checksum và 32 bits data. Như vậy theo quy tắc trên thì người ta sẽ tính tổng của:  
Source port + Destionation port+ UDP length+ data---> lấy phần bù---> ghi vào UDP checksum.  
Tính như thế này có đúng chưa ạ ?  
  
- Theo như cách tính này thì UDP pseudo header không có được tính đến, nhưng mà khi người ta lại nói là "UDP bao gồm cả 12- byte pseudo- header chỉ để phục vụ cho việc tính toán checksum". Điều này nghĩa là sao ạ ?  
  
- Tại sao UDP length lại có cả trong phần header và phần pseudo-header, trong khi ta đã chỉ ra rằng trường này là có thể có hoặc không( vì UDP length= IP datagram length- IP header)