Mittuniversitetet
MID SWEDEN UNIVERSITY

DT172G Nätverksanalys

Laboration I:
Insamling av data

Lennart Franked*

30 oktober 2019

Innehåll

Syfte

I denna laboration kommer ni att installera och arbeta med PCAP, DUMP, Wireshark och SNMP för datainsamling. Ni ska efter utförandet skriva en reflektion om era erfarenheter kring datainsamlingen i kursens diskussionsforum.

Mål

Efter att du har avklarat *L1 - Insamling av data* har du uppfyllt följande delmål:

*E-post: lennart.franked@miun.se.

- kunna på ett strukturerat sätt utföra datainsamling på ett nätverk,
- ha kännedom om vart i ett nätverk man bör placera sin övervakning för att uppnå önskat resultat.
- behärskar olika verktyg för att samla in data, samt
- utbyta erfarenheter kring utförande av datainsamling och föra en konstruktiv dialog med en studiekollega.

Läsanvisningar

Innan du påbörjar *laboration ett* skall du ha deltagit eller sett inspelningarna utav de första fem föreläsningarna i kursen. Samt tagit del av de angivna läsanvisningarna för respektive föreläsning.

För första delmomentet i denna laboration skall du dessutom översiktligt läst [1, Kap 1 - 7] samt sett [2], för att få en bredd över hur man kan använda programmet.

För det andra delmomentet i denna laboration, kommer ni att använda er utav DPKG [3], och ni skall därför ha studerat de exempel som finns att hitta på [4] samt översiktligt läst manpage för tcpdump(8) vilket är det program ni kommer använda för att samla in rådata.

Slutligen läser ni översiktligt följande RFC:er: [5], [6], [7], [8], [9], [10], [11], [12].

1 Introduktion

I denna inledande laboration kommer vi att kolla på hur vi på olika sätt kan samla in data om nätverket och dess noder. Det är viktigt att behärska de olika verktyg som finns för detta och vi kommer därför i denna laboration att utforska olika metoder för insamling av data. Vi börjar med att använda oss av verktyget Wireshark som är ett grafiskt program som använder sig i grunden av biblioteket pcap för att samla in data. Beroende på vad vi vill göra, så kan vi bli begränsade av det grafiska gränssnittet i Wireshark. Vi kommer därför att på nästa delmoment av denna laboration plocka bort det grafiska gränssnittet ur ekvationen och arbeta med tcpdump(8) som likt wireshark använder sig utav biblioteket pcap. Tcpdump(8) i sig är ett användbart verktyg och det går på liknande sätt som i Wireshark att filtrera det insamlat data. Om vi däremot vill plocka ut specifik data, för att utföra beräkningar med det, kan det bli aningen svårare. Vi kommer därför utforska möjligheterna att med hjälp av Python kan plocka ut och utföra beräkningar av det data vi får ut från tcpdump(8). Det sista vi kommer göra i denna laboration är att kolla på hur vi kan använda oss av SNMP för att samla in data. Vi kommer därför installera en SNMP agent och laborera med vad för typ av data vi kan få fram.

Tabell 1: Uppgiftsindelning

Gruppmedlem:	Uppgifter som skall utföras
1	1 - 2
2	3 - 4
3	5 - 6

2 Genomförande

Nedan följer de tre deluppgifter där vardera uppgift innehåller övningsuppgifter ni skall utföra, för att lära er behärska de olika insamlingsmetoderna.

Varje uppgift innehåller ett antal frågor som skall besvaras, de frågor ni ska besvara bestäms utifrån vart i studiegruppen ni hamnar. Se kurssidan i Moodle för mer information.

Se avsnitt 3 för information om hur ni skall redovisa era resultat.

2.1 Datainsamling med hjälp av Wireshark

Wireshark har två stycken typer av filter, capture och display. Skillnaden mellan dessa är att Capture filtrerar vilken trafik som den skall samla in, medan Display samlar in all trafik och därefter filtrerar ut önskad data. Det går i Wireshark att spara datat i ett antal olika format, dock är majoriteten av dem varianter av pcap-formatet. Detta gör det möjligt för oss att använda Wireshark för att samla in rådata och därefter skriva ett program i Python för vidare analys.

Starta igång en datainsamling i Wireshark och med en webbläsare gå in på startsidan för miun.se (utför detta i private browsing så att ditt resultat inte blir förvanskat av tidigare cachad information)

I detta moment ska du besvara följande frågor:

1. Skapa ett filter som gör att du hittar samtliga fragment av ett UDP-datagram.
2. Hur hittar man det paket som har lägst TTL.
3. Skapa ett filter som visar omsända paket.
4. Skapa ett filter som visar de TCP-bekräftelser som mottagits fler än en gång (DUP ACK).
5. Hitta ethernetpaket som har en felaktig checksumma.
6. Med hjälp av funktionen 'Follow TCP streams' hur många paket krävdes det för att få upp startsidan på miun.se?

2.2 Datainsamling med hjälp av Tcpdump och Python

Vi har nu kollat på hur man samlar in data med hjälp av Wireshark. Det händer dock att vi vill hämta in data för att presentera detta, eller utföra beräkningar på datat. Det är dock inte alltid möjligt att utföra detta i Wireshark och vi kommer därför nu kolla på hur vi kan plocka ut precis den data vi vill med hjälp av tcpdump och python.

Till er hjälp använder ni exempelvis 'Dumb Packet Module'(dpkt) till Python, hittar ni andra bibliotek som ni tycker fungerar bättre för att arbeta med pcap får ni självklart använda denna istället. Likt tidigare uppgift, så starta igång tcpdump(8) och med en webbläsare går du in på startsidan för miun.se(likt tidigare utför du detta i private browsing så att ditt resultat inte blir förvanskat av tidigare cachad information)

Ni ska ni skriva ett enklare program för att lösa följande problem:

1. Skriv ut de eller det paket som har lägst TTL?
2. Skriv ut alla omsända paket.
3. Hur många paket krävdes det för att få upp startsidan på miun.se?
4. Tiden för det första paketet i dataströmmen för att komma åt startsidan på miun.se
5. Tiden för det sista paketet i dataströmmen för att komma åt startsidan på miun.se
6. Skriv ut alla dubbla TCP-bekräftelses (DUP ack)

2.3 Datainsamling med hjälp av SNMP

Vi har nu kollat på olika sätt att analysera nätverket genom att lyssna på nätverket. Ibland behöver vi dock mer information som vi inte kan få genom att titta på nätverkstrafiken. Exempelvis CPU, minnesanvändning eller andra interna processer på våra noder. För att kunna få fram denna information kan vi ta till hjälp protokollet SNMP. Genom att installera en SNMP-agent på en nod kan vi komma åt information som annars inte är åtkomlig genom vanlig paketinsamling.

Observera att anvisningarna nedan är skrivna för Ubuntu/Linux, väljer ni att använda något annat system får ni anpassa instruktionerna därefter.

2.3.1 Installation av SNMP

Börja med att installera en SNMP-server, exempelvis Net-SNMP [13]. Ni kan med fördel använd den officiella dokumentationen till hjälp vid installationen. För denna laboration går det utmärkt att använda version 2c av SNMP. Kom ihåg att kolla vilka beroenden (eng. *dependencies*) som krävs (Python, Perl) vid installationen.

Nästa steg blir nu att konfigurera er SNMP-server, vilket enklast kan göras med hjälp av `snmpconf(1)` om ni använder er av Net-SNMP. De konfigurationsfiler ni måste skapa är `snmpd.conf` vilket är agentens konfiguration, samt `snmp.conf` vilket är SNMP-klientens konfiguration.

Ni kan senare under projektet experimentera med exempelvis traps och andra funktioner som SNMP har.

I `snmpd.conf` måste följande konfigureras:

- Community sträng för SNMPv2c, en för read-only och en för read-write.
- System information.

I `snmp.conf` kan ni sätta *default community* till det ni satte för er SNMP-agent, på så sätt behöver ni inte fylla i community-strängen varje gång ni ska göra en SNMP-förfrågan mot er nyligt installerade agent. Ange även att den som standard ska använda version 2c av SNMP.

Avsluta `snmpconf(1)` och kopiera `snmp.conf` samt `snmpd.conf` enligt anvisningarna och starta därefter `snmpd(8)` med root-rättigheter.

2.3.2 Användning av SNMP

Du har nu installerat både en SNMP-agent samt en SNMP-klient och kan nu börja samla in information från agenten med hjälp av

- `snmpget(1)`,
- `snmpgetnext(1)` samt
- `snmpwalk(1)`.

Testa att din agent är igång genom att exempelvis göra en `snmpwalk(1)` på grenen

```
iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).system(1)
```

i MIB-trädet.

Se Net-SNMP:s officiella dokumentation [13] alternativt manualsidorna för `snmpget(1)` och `snmpwalk(1)` för information om hur ni använder SNMP-klienten.

Om ni vet vilken information ni vill hämta ut från MIB-trädet kan ni använda [14] för att söka efter objekt. Sök exempelvis efter vilket objekt som visar vilken IP-adress er dator har och bekräfta sedan detta genom att göra en `snmpget(1)` på detta OID.

Med hjälp av SNMP, ta reda på följande information:

1. Hur många paket krävs det för att få upp startsidan på miun.se?

2. Hur många korrupta paket har tagits emot samt vilka typer av korrupta paket kan man se?
3. Din dators lager 3-adresser samt lager 2-adresser.
4. Datorns routing- och ARP-tabeller.
5. primärminne och swapminne,
6. CPU-användning,

I de fall du hämtat information från MIB trädet, ange både objektnamn samt OID till detta objekt.

3 Examination

Dina svar på de uppgifter du utfört ska skickas in som PDF i inlämningslådan för L1. När ni har lämnat in er laborationsrapport och blivit godkänd på denna, ska ni nu inom gruppen skriva ihop en summering av era erfarenheter från laborationen ni utfört. Utgå från följande frågor i er summering.

1. Med cirka 300 ord, resonera kring hur och vart man bör samla in data för att effektivisera insamling av data.
2. Hur många paket krävdes det för er att få upp startsidan
3. Var det någon skillnad i antalet paket som krävdes för att komma åt Miuns startsida inom gruppen? Ange hur många paket ni fick och ge en teori om varför de skiljer sig åt. miun.se?
4. I [1, Kap 7] finns det, utöver 'Follow TCP Streams' en del funktioner omnämnda, testa dessa och välj därefter inom gruppen ut den funktion som ni tycker är mest användbar för dig själv och dina studentkollegor. Med cirka 200 ord motivera varför ni valde just denna funktion.
5. Leta upp ett objekt i SNMP som ni tycker kan ge användbar information och ge en kort förklaring om denna. Motivera ert val.
6. Lägg ut era lösningar för uppgifterna från delmoment 1 -3. Förklara hur ni löste uppgifterna, ditt tillvägagångssätt samt hur din lösning fungerar.

Efter att ni publicerat ert inlägg ska ni nu kommentera en annan grupps inlägg. Följande gäller vid val av inlägg att kommentera på.

- Reservera inlägget genom att först skriva gruppnamnet innan ni börjar kommentera (på så vis undviker vi att flera grupper kommenterar på samma inlägg).
- Om möjligt, kommentera inte på den grupp som kommenterar på ert inlägg.

- I den mån det går, inläggen ska inte varit kommenterade på tidigare. Om det inte finns några okommenterade inlägg, avvakta någon dag innan ni skriver en kommentar, eller 'efterlys' ett inlägg.

Ha följande i åtanke när ni kommenterar inlägg:

1. Ni ska vara en kritisk vän. Håll en god ton, var konkret och objektiv. Ge utförliga och hjälpsamma kommentarer. Motivera väl.
2. Finns det någon aspekt kring effektiv datainsamling som gruppen har missat? Eller någon aspekt som ni själv inte tog med?
3. Var det någon skillnad i antalet paket som krävdes för att komma åt Miuns startsida? Hade ni samma teori om varför antalet skilde sig åt? Reflektera över detta.
4. Vilka funktioner och objekt rekommenderar författaren att man använder i Wireshark och SNMP? Ge konstruktiv kritik men lyft även upp de bra poänger som nämns. Varför valde du själv bort just denna funktion? Om du valde samma, förklara varför.
5. Hur har gruppen löst uppgifterna? Förstår ni lösningarna? Om inte, be om att få en bättre förklaring.
6. Fungerar lösningarna?
7. Ge synpunkter, belys de bra delarna. Vad kan förbättras? Motivera väl.
8. Ett bra sätt att komma igång med kommentarerna är att börja med att ge en kort summering över hur din studentkollega har löst uppgiften. Du visar på så sätt hur du själv uppfattar lösningen och kan i din summering ge kommentarer. Exempelvis *'Som jag förstår din lösning har du har valt att skapa en loop som går igenom samtliga paket. För varje paket plockar du ut IP-numret och lagrar dessa i en lista. Denna lista använder du sedan för att ta reda på hur många paket som sänts och mottagits. Detta var en väldigt intressant lösning, har du funderat på att istället basera antalet sända paket på MAC-adressen istället? Varför har du sorterat din lista?'*
9. Kom ihåg att era inlägg ska visa att ni har förstått uppgifterna och dess lösningar.

Referenser

[1] Ulf Lamping, Richard Sharpe och Ed Warnicke. *Wireshark User's Guide.* URL: `https://www.wireshark.org/docs/wsug_html_chunked/` (hämtad 2015-06-30).

[2] Gerald Combs. *Introduction To Wireshark.* 2009. URL: `https://www.wireshark.org/video/wireshark/introduction-to-wireshark/` (hämtad 2015-06-30).

[3] *DPKT*. URL: https://dpkt.readthedocs.org (hämtad 2015-09-15).

[4] *Dpkt Examples*. URL: https://dpkt.readthedocs.org/en/latest/examples.html (hämtad 2015-09-15).

[5] M.T. Rose och K. McCloghrie. *Structure and identification of management information for TCP/IP-based internets*. RFC 1155 (INTERNET STANDARD). Internet Engineering Task Force, maj 1990. URL: http://www.ietf.org/rfc/rfc1155.txt.

[6] J.D. Case, M. Fedor, M.L. Schoffstall m. fl. *Simple Network Management Protocol (SNMP)*. RFC 1157 (Historic). Internet Engineering Task Force, maj 1990. URL: http://www.ietf.org/rfc/rfc1157.txt.

[7] K. McCloghrie och M. Rose. *Management Information Base for Network Management of TCP/IP-based internets:MIB-II*. RFC 1213 (INTERNET STANDARD). Updated by RFCs 2011, 2012, 2013. Internet Engineering Task Force, mars 1991. URL: http://www.ietf.org/rfc/rfc1213.txt.

[8] K. McCloghrie, D. Perkins och J. Schoenwaelder. *Structure of Management Information Version 2 (SMIv2)*. RFC 2578 (INTERNET STANDARD). Internet Engineering Task Force, april 1999. URL: http://www.ietf.org/rfc/rfc2578.txt.

[9] S. Waldbusser, R. Cole, C. Kalbfleisch m. fl. *Introduction to the Remote Monitoring (RMON) Family of MIB Modules*. RFC 3577 (Informational). Internet Engineering Task Force, aug. 2003. URL: http://www.ietf.org/rfc/rfc3577.txt.

[10] B. Claise. *Cisco Systems NetFlow Services Export Version 9*. RFC 3954 (Informational). Internet Engineering Task Force, okt. 2004. URL: http://www.ietf.org/rfc/rfc3954.txt.

[11] B. Claise. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information*. RFC 5101 (Proposed Standard). Obsoleted by RFC 7011. Internet Engineering Task Force, jan. 2008. URL: http://www.ietf.org/rfc/rfc5101.txt.

[12] J. Quittek, S. Bryant, B. Claise m. fl. *Information Model for IP Flow Information Export*. RFC 5102 (Proposed Standard). Obsoleted by RFC 7012, updated by RFC 6313. Internet Engineering Task Force, jan. 2008. URL: http://www.ietf.org/rfc/rfc5102.txt.

[13] "Net-SNMP". 2013. URL: http://www.net-snmp.org/.

[14] Cisco. "SNMP Object Navigator". 2013. URL: http://tools.cisco.com/Support/SNMP/do/BrowseOID.do?local=en.