Syddansk Universitet

DM830 - Netværkssikkerhed

Orme, fokus på Morris ormen

Skrevet af
Kenneth A. Knudsen (210190)
Lars Andersen (120290)

14. maj 2012

Indhold

1 Indledning **1**

2 Hvad er en orm? **1**

3 Formeringsmetoder **1**

4 Formeringsmodel **2**

5 Orme teknologier **3**

6 Orme på mobile enheder **3**

7 Modforanstaltninger **4**

7.1 Krav til effektivt forsvar 4

7.2 Forsvarsmetoder 4

7.3 Proaktiv orme-inddæmning 5

7.4 Netværksbaseret orme-forsvar 6

8 Morris ormen - Verdens første computer orm **7**

8.1 Sendmail angreb 7

8.2 Fingerd 7

8.3 Remote shell angreb 8

8.4 Konsekvenser 8

1 Indledning

Denne rapport omhandler emnet computerorme. I første del af rapporten gennemgås teorien bag computerorme. Man vil kunne finde svar på spørgsmål som: Hvordan angriber en orm? og hvordan forsvare man sig mod et ormeangreb? Sidste del af rapporten handler om et konkret eksempel på en orm. Dette er historiens første orm, nemlig Morris Ormen.

2 Hvad er en orm?

Kendetegnet for en orm er, at den spreder sig selv, ofte igennem sikkerhedshuller. Ormeinfektionen kan ske uden brugerindblanding, men det er intet krav. En orm kan for eksempel også sprede sig som en vedhæftet fil via e-mail, hvor brugeren skal åbne filen før systemet bliver inficeret. Kriteriet for orme er altså, at de kan sprede sig fra inficerede systemer til andre systemer.

Når en orm har inficeret et system kan den foretage skadelige ting, som fx at slette filer på det inficerede system, sende spam-mails og andre dokumenter via e-mail. En orm bruges ofte i komplekse angreb, hvor den fungerer den som transporteringsenhed, hvor det medsender andre former for malware, fx en trojansk hest, som installere en bagdør på det inficerede system.

Generelt vil en orm altid belaste netværket, som den operere på, da den udnytter netværket til at sprede sig på.

3 Formeringsmetoder

En af ormens store fokuspunkter er, hvordan den får kopieret sig selv videre til en ny maskine. Til dette formål, har den nogle redskaber med netværksadgang, at gøre brug af.

En hyppig anvendt metode, er at en orm mailer sig selv til andre maskiner, den bliver så kørt, når mailen enten er modtaget eller læst. Dette redskab var ganske anvendelig i computerens unge dage, da mail var det eneste kommunikationsmiddel ud af virksomhedens interne netværk.

En orm kan befinde sig på en maskine, der igennem visse programmer har rettigheder til at køre programmer på eksterne maskiner. Disse muligheder er oplagte for en orm at benytte. Det kunne også blot være huller i programmer med netværksadgang som ormen udnytter.

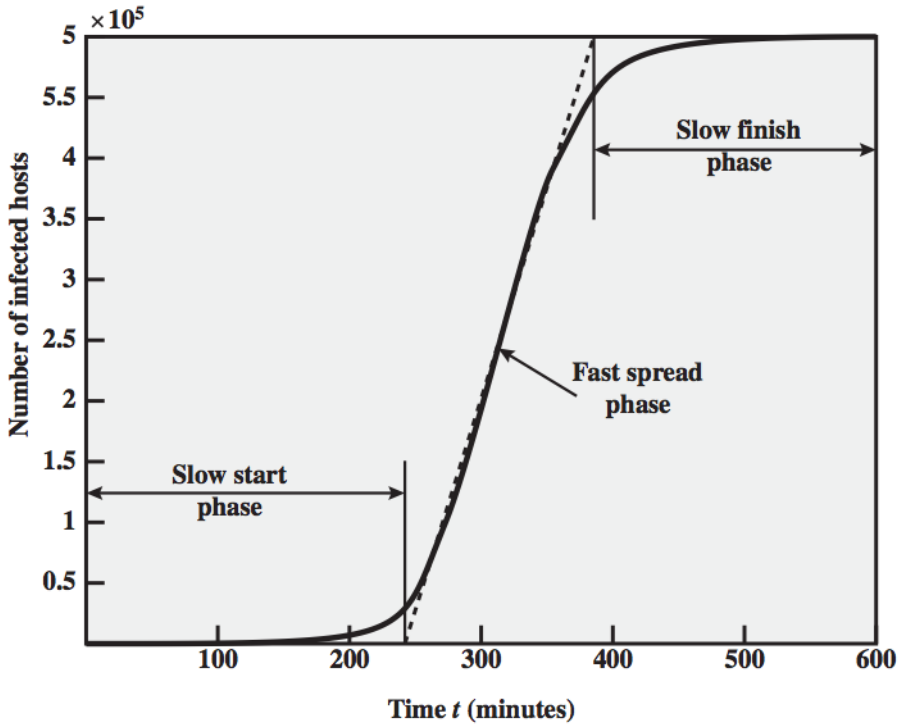
Orme kan også udnytte en computers fjernadgang til andre maskiner. Dette kræver dog ofte, at et kodeord skal knækkes.

4 Formeringsmodel

Hastigheden for en orms spredning afhænger af en række faktorer. Herunder hvilken måde den bruger til spredning, fx gennem e-mail eller direkte via internettet, hvilke sikkerhedshuller der udnyttes, er de store eller små? Eller måske et zero-day exploit? En faktor er også om ormens angreb ligner en tidligere orms måde at angribe på. Tidligere fremgangsmåder vil nemmere blive opdaget af sikkerhedssoftware.

Generelt kan man opdele ormens spredning i tre faser. *Initial phase* (eller startfasen), ormen sættes i gang og spredningen foregår i et langsomt tempo, dog sker spredningen eksponentielt. *Middle phase* Spredningen sker i et højt tempo, i noget som minder om lineær tid. Når de mest udsatte maskiner er blevet inficeret, så når vi til *finish phase*, hvor ormen bruger lang tid på at inficere allerede inficerede maskiner, og har svært ved at inficere systemer med bedre beskyttelse.

De beskrevne faser kan illustreres med nedenstående formeringsmodel



Figur 1: Oversigt over ormens spredningsfaser

5 Orme teknologier

Orme bliver mere og mere avancerede, for at kunne gennemtrænge sikkerhedssoftwaren, som også hele tiden udvikler sig. Her er en liste af teknikker nyere orme benytter sig af.

- **Multiplatform:** Orme er ikke længere begrænset til kun en platform.
- **Multi-exploit:** Orme bruger gerne flere forskellige måder, at formere sig på. Så den har flere strenge at spille på.
- **Ultrahurtig spredning:** Inden man frisætter sin orm, kan man lave en skanning efter sårbare maskiner. På den måde får ormen et boost i startsfasen.
- **Polymorphic:** Orme kan tage forskellige udseender på forskellige maskiner, men med samme funktionelle operationer. Dette gør det vanskeligt for sikkerhedshedssoftwaren at finde ormen.
- **Metamorphic:** Denne egenskab ligner den polymorfiske, men her ændres både ormens udseende samt dens adfærd.
- **Transporteringsenhed:** Da orme køre som en selvstændig process, bliver de ofte brugt til at transportere andet malware.
- **Zero-day exploit:** Noget af det mest effektive, er at udnytte sikkerhedsbrister som endnu ikke er fundet, dette skaber en yderst værdifuld overraskelsesfaktor.

6 Orme på mobile enheder

Et fænomen vi vil komme til at se meget til i en nær fremtid er orme på mobile enheder. Målet er smartphones og tablets, som har mulighed for at installere apps. Dermed er det muligt at installere en orm på enheden.

De første orme man har kendskab til stammer fra 2004. Disse orme spredte sig via Bluetooth og MMS. Målet var primært de populære Nokia mobiler.

Nyere mobile OS'er som Android og iOS (iPhone) har den egenskab at de køre apps i en sand-box. Dette gør det svært for ormen at få adgang til det underliggende system, og dermed svært for ormen at sprede sig videre. Det er dog lykkedes enkelte orme at få adgang til mobilens sms-funktion, og de har dermed været i stand til at sende beskeder til dyre mobilnumre. Andre orme har haft succes med at stjæle personlige oplysninger fra enhederne.

Et andet angreb som mobile orme bruges til, er at sprede computerorme, som så vil inficere en computer, så snart en mobil tilsluttes til computeren via et kabel.

7 Modforanstaltninger

7.1 Krav til effektivt forsvar

For at modstå ormeangreb skal man have sig noget godt sikkerhedsoftware. For at blive karakteriseret som god, er der visse krav det skal overholde.

Generalitet, software skal håndtere flere typer orme, så det ikke er begrænset til en specifik orm. Dette gælder selvfølgelig både internt på netværket og mod eksterne angreb. Softwaren skal heller ikke modificere eller kræve noget specifikt af den maskine det er installeret på.

Hvis en orm opdages, så skal der reageres prompte, så det formindsker spredningen.

Softwaren skal kunne håndtere den udfordring at kreatøren af en orm kender til alverdens sikkerhedssoftware og derfor ved hvordan det operere. Alligevel skal ormen kunne stoppes, om ikke andet, så med en hurtig opdatering.

Det sidste krav er, at alle kravene tilsammen ikke bør drive rovdrift på maskinens resurser. Sikkerhedssoftware bør altså tage så lidt af systemets ressourcer som muligt.

Det skal nævnes, at endnu intet stykke software dækker alle kravene alene.

7.2 Forsvarsmetoder

Når man skal forsvare sig mod orme kan man gøre brug af forskellige forsvarsmetoder. Disse metoder inddeles i seks klasser:

- **A: Signature-based worm scan filtering:** Denne klasse identificere mistænkelig trafik og generere en orme signatur, som forebygger orme i at tilgå og forlade et netværk eller en inficeret maskine. Denne klasse har problemer med polymorphic orme, da de ændre deres udseende og dermed også deres signatur.
- **B: Filter-based worm containment:** Samme tilgang som klasse A, men fokusere på indhold frem for signatur. Filteret tjekker en besked for orme. Dette kan være meget effektiv, men kræver effektive opdagelsesalgoritmer.

- **C: Payload-classification-based worm containment:** Gennemsøger netværks-pakker for at se om de indeholder en orm. Man skal tage sig i agt for, at der kan opstå mange falske positiver (begrænser systemets ydeevne) og negativer (ormen bliver ikke opdaget).
- **D: Threshold random walk (TRW) scan detection:** Hacker laver portscans af IP-addresse for at få adgang til system. TRW kigger efter tilfældige forsøg på at finde åbne porte, som er tegn på et ormeangreb. Effektiv mod den gængse struktur set hos orme.
- **E: Rate limiting:** Begrænser antallet af forbindelser en host kan lave, samt antallet af unikke IP-adresser der kan forbindes til. Denne metode kan lave forsinkelser i den almindelige trafik. Metoden virker ikke på stealth orme, der spreder sig langsomt.
- **F: Rate halting:** Blokere udgående trafik når en given threshold for antallet af udgående forbindelser er overskredet. Metoden skal hurtigt kunne fjerne fejlblokeringer på en transparent måde. Det er muligt at kombinerer metoden med signatur- og filter-baserede metoder. Ligesom Rate limiting klassen, virker metoden ikke på langsomme stealth orme.

7.3 Proaktiv orme-inddæmning

Proaktiv orme-inddæmning (på engelsk *Proactive Worm Containment*, forkortet PWC) er host baseret. PWC er designet til at behandle orme, der spreder sig hurtigt. Softwaren på hosten kigger efter stigninger i raten af udgående forbindelsesforsøg samt kigger på hvordan disse forsøg fordeler sig til andre enheder på nettet. Når en stigning opdages, så blokeres fremtidige udgående forbindelser fra hosten.

PWC systemet består af en PWC manager og PWC agenter på host-maskinerne. Et eksempel på en arkitektur, som inkluderer PWC, ses på figur 2.

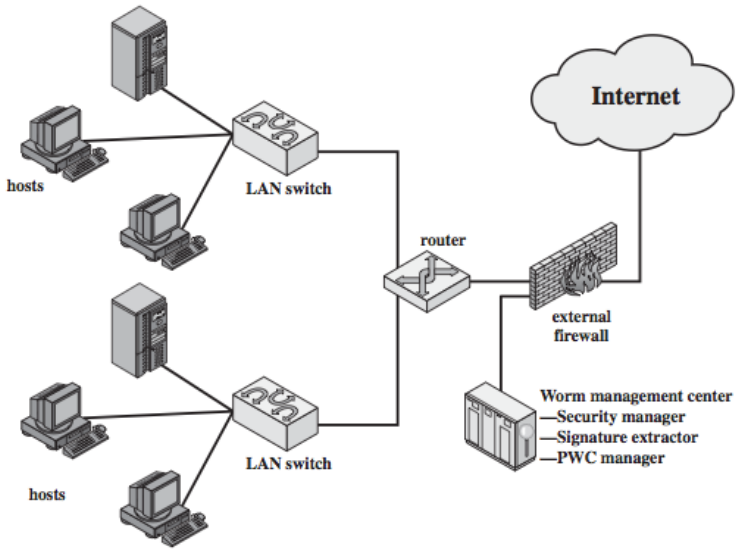
Funktionaliteten af PWC arkitekturen kan beskrives således:

A. PWC agent overvåger udgående trafik, hvis stigning opdages blokeres al udgående trafik og der sendes en alarm til PWC manageren.

B. PWC manager modtager en alarm og sender den videre til alle agenter

C. Agenter (hosts) modtager alarm og må beslutte om den skal ignorere alarmen, hvilket den gør, hvis der gået en del tid siden den sidste indkomne pakke. Ellers blokkeres al udgående trafik og start relaxation analyse.

D. Relaxation analyse: Agent overvåger udgående trafik i tidsperiode, hvis trafik ikke overstiger forudsat threshold, så fjernes blokering. Ellers fortsæt blokering, og overvågning fortsætter i endnu en tidsperiode. Hvis



Figur 2: Eksempel på PWC implementering.

thresholden fortsætter med at være oversteget, så vil agenten isolerer hosten og rapportere til PWC manager

7.4 Netværksbaseret orme-forsvar

Det vigtige ved et godt netværksbaseret forsvar er, at programmerne på de enheder der overvåger netværket er gode. Enhederne kan være passive sensorer, firewalls eller honeypots (En enhed der opdager forsøg på uautoriseret brug). Disse enheder skal indeholde programmer, der både håndtere indgående- og udgående trafik.

Enheder der overvåger udgående trafik, betyder mindre hvor de er placeret i netværket, derimod skal indgående overvågningsenheder placeres mellem virksomhedens netværk og internettet.

En brugt teknik til at overvåge indgående trafik er, at se om der kommer trafik med adresse til ikke brugte IP'er. Denne mistænkelige adfærd medfører at en notifikation sendes til en correlation server, som samler og vurdere den information overvågningsenhederne sender.

Generelt bruges overvågningen til, at holde øje med mystisk adfærd. Problemet er bare at blokere ved uautoriseret brug og tillade ved lovligt brug.

Der findes også opstillinger, der kan kaldes selv-helbredende. Disse er noget af det mest effektive mod zero-day exploit angreb. Når et angreb registreres sendes både det program der bliver angrebet og angriberen til en sandbox.

Her analyseres og testes, hvis der findes en patch, så testes den igennem. Hvis denne er acceptabel udsendes en opdatering til det program der blev angrebet.

8 Morris ormen - Verdens første computer orm

I 1988 var der en studerende ved Cornell university, der den 2. November slap en orm løs på MIT (Massachusetts Institute of Technology). Hans navn var Robert Tappan Morris. Den orm han kreerede bliver gerne omtalt som historiens første orm, om ikke andet så den første orm med så stor medieopmærksomhed.

Ormen indeholdt ikke som sådan noget skadeligt, dog viste den sig, at den indeholdte fejl, der lavede alvorlig ravage på inficere maskiner.

Robert har sidenhen udtalt, at han ønskede at måle internettets størrelse, samt sætte større fokus på netværkssikkerhed, hvilket ikke var specielt respekteret på daværende tidspunkt.

I de følgende afsnit beskrives de tre angrebsmetoder som Morris Ormen brugte.

8.1 Sendmail angreb

Morris ormen udnyttede en bagdør i Sendmail programmets debug mode. Denne debug mode tillod at man kunne sende programmer og køre dem på modtagerens maskine, uden brugeren behøvede at foretage sig noget.

Det var en meget usædvanlig egenskab for en mail klient, men skulle eftersigende være brugt til test af Sendmail programmet. En stor fejl var dog at debug mode var slået til som standard på de maskiner, det var installeret på.

Morris ormen medsendte sig selv i form af VAX og Sun kodefiler, som den efterfølgende kompilerede og kørte på modtagermaskinen.

8.2 Fingerd

Morris ormens anden angrebsmetode var, at udnytte et hul i Finger programmet. Slutbogstavet 'd' står for daemon.

Finger programmer indeholder brugerinformation, det kan være fulde navne, email, telefonnummer o.l. Disse data kan så hentes ud via forespørgsler til den pågældende server.

Sårbarheden ved Fingerd var, at man kunne lave et overflow på bufferen i programmets input-funktion. Derved kunne ormen placeres i bufferen så den blev kørt på maskinen.

Der findes adskillige kald fra C-biblioteket, hvilket det meste var skrevet i dengang, hvorved man kunne overskride en buffer. En patch til dette problem blev at sætte en "bound"på disse kald.

8.3 Remote shell angreb

Morris ormens trejde angrebsmetode gik ud på at logge sin ind via remote shell (RSH/REXEC) og prøve at opnå root adgang på maskinen. Dette gjorde den ved at gætte kodeorderet. Først prøvede den at gætte kodeordet ud fra en kombination af brugernavn, for- og efternavn, hvis oplysninger var tilgængelige. Derefter forsøgte den at gætte kodeordet ud fra en medsendt fil, som indeholdt 432 ord, som Robert Morris mente ville være populære kodeord. Hvis det mislykkedes at gætte kordeordet med denne fremgangsmåde, så brugte den systemets ordbog til at fortsætte kodeordsgætteriet.

Hvis det lykkedes at komme ind som root, kiggede ormen i maskinens host filer, efter betroede maskiner, som den kunne sprede sig videre til.

8.4 Konsekvenser

Morris ormen var, som nævnt tidligere, umiddelbart ganske harmløs. At den alligevel nåede at give en del computeradministratorer hovedpine skyldtes, at ormen kunne inficere den samme computer mere end en gang. Morris ønske var at måle nettes størrelse, og han frygtede at hvis ormen stillede spørgsmålet "Er denne computer inficeret?". Så ville det blive brugt imod ham selv og det vil være for nemt at stoppe spredningen. Derfor implementerede han, at hver syvende der svarede ja til spørgsmålet, blev inficeret alligevel. Denne sandsynlighed var alt for højt sat og rigtig mange maskiner blev inficeret så mange gange, at de blev ikke-funktionelle.

Ormen nåede angiveligt at sprede sig til omkring 6.000 ud af 60.000 maskiner i USA. Det havde dog den positive effekt, at fokus på sikkerhed blev øget.

For Morris selv, så betød det at han var den første der blev dømt under en lov ved navn Computer Fraud and Abuse Act fra 1986, han blev idømt 3 års betinget fængsel, 400 timers samfundstjeneste og en bøde på 10.000 dollars.