Oppdatering av programvare og informasjonssikkerhet

Topic 169 - INF3510 Informasjonssikkerhet

Våren 2017

Problemstilling

Hvilke utfordringer finnes knyttet til oppdatering av programvare og informasjonssikkerhet? Hvordan kan disse eventuelt håndteres?

Daniel Lange (Kand. 15637)
Ingeborg Eilertsen (Kand. 15588)
Marius Christensen (Kand. 15703)

Antall Ord: 6728

Universitetet i Oslo

Innholdsfortegnelse

1. Innledning: Utvikling av informasjonssystemer, før og nå

Det er ikke lenge siden vi kjøpte alle dataprogrammer på en CD-plate i en fysisk butikk. Dette var ofte den eneste versjonen av programmet, med mindre det ble gitt ut en ny utgave ett år senere, eller mer. Når produktet først var kommet i butikkhyllene så var det grundig testet og klar for sluttbrukeren. Dataprogrammene ble ofte utviklet etter fossefallsmodellen, og vedlikeholdet ble holdt til et absolutt minimum da det ikke fantes noen effektiv måte å distribuere oppdateringer på. Med internett derimot fikk utviklerselskapene helt nye og kostnadseffektive måter å distribuere nyere versjoner på, noe som også åpnet opp for nye tilnærminger til systemutviklingsprosessen. Den gjeldende praksisen med en lite fleksibel og trinnvis prosess, ble gradvis mer smidig. Smidige systemutviklingsmodeller som for eksempel Scrum og Kanban tillater å legge til eller endre funksjonalitet i programvaren kontinuerlig, og at feil kan rettes underveis. Derfor kan man diskutere om programmer som lanseres i dag er like godt testet som før, når det viser seg at disse nye mulighetene gjør at det ofte kan komme kritiske oppdateringer for programvaren kort tid etter utgivelsen. Noe av grunnen til dette kan være at det er ennå viktigere enn før for utviklerne at deres produkt blir det første og største i markedet, fordi man ønsker å tiltrekke seg en stor brukermasse for å oppnå *the bandwagon effect.* (Dette begrepet refererer til troen som mennesker kan ha på at dersom alle andre gjør det, så må det være riktig.) Muligheten som internett gir, gjør at de kommersielle utviklerne kan prioritere *time to market* fremfor omfattende testing. For å oppnå denne fordelen så inngås det kompromisser, som reduserer den generelle kvaliteten og sikkerheten til produktet i første omgang, for deretter å forbedre den ved hjelp av av oppdateringer senere. Dette kan de gjøre fordi de legger patching inn som en del av utviklingskostnadene til programvaren (Goldschmidt, Dark & Chaudhry, 2010, s. 8).

Det at de kommersielle utviklerne har insentiv til å gi ut programmer med mer feil enn tidligere, gir nye muligheter for dem som ønsker å utnytte sikkerhetshull til egen vinning. I tillegg så har vi fått enda flere enheter å forholde oss til, noe som igjen gir nye muligheter, både for kommersielle og kriminelle krefter. Trusselagenter kan blant annet bruke internett for å hamstre inn informasjon til ondsinnede formål, stjele passord for å få tilgang til tjenester, kryptere datamaskiner for deretter å kreve penger for å gjenåpne dem. I gjennom tilgang til internett har datamaskinen og andre enheter blitt mer eksponert og truslene har blitt

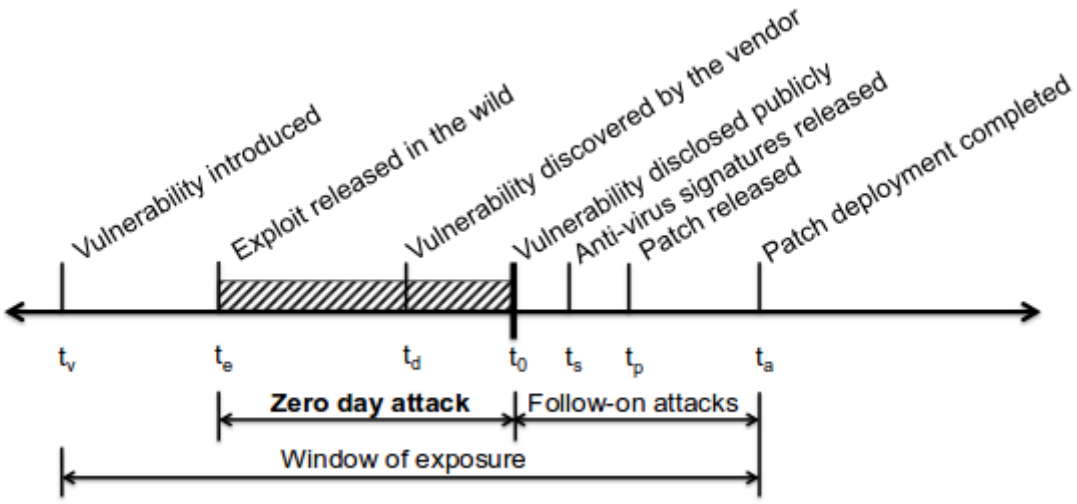
flere. Det har derfor blitt nødvendig for operativsystemene og annen programvare å fokusere mer på å rette feil i programvaren for å minske risikoen for at noen klarer å utnytte sårbarhetene. Sårbarhetene har også blitt lettere å utnytte ved hjelp av internett, fordi informasjon om sårbarhetene kan spres kjapt gjennom diskusjoner og lukkede fora.

I vår oppgave har vi valgt å skrive om informasjonssikkerhet og oppdatering av programvare. Spesielt ønsket vi å undersøke hvilke utfordringer som finnes, og hva man eventuelt kan gjøre for å møte disse. Vi har sett både på privatpersoner og bedrifter. Datagrunnlaget i oppgaven består av litteratur, i tillegg har vi laget og gjennomført en nettbasert spørreundersøkelse. Denne ble benyttet for å undersøke privatpersoners oppdateringsvaner.

2. Distribuering av sikkerhetsoppdateringer

2.1. Hvorfor er det viktig å oppdatere i tide?

En sårbarhet er en svakhet i et datasystem som kan utnyttes for å stjele eller gjøre skade på verdier (*information assets*). Sårbarheter kan eksistere i alle typer av programvare, og kan utnyttes av trussel-agenter frem til de blir funnet og tatt hånd om av ansvarlige parter. Om sårbarheten blir utnyttet eller ikke, avhenger ikke bare av at sårbarheten finnes i systemet. Det er i tillegg nødvendig at en trussel-agent får kjennskap til den aktuelle sårbarheten, at vedkommende har evne til å utvikle eller finne en exploit som kan utnytte den, samt at vedkommende har tilstrekkelig motivasjon til å gjennomføre angrepet. (Jøsang A., 2017b, s. 4) Som vi ser på tidslinjen i figur 1 så varer *window of exposure* helt fra sårbarheten blir introdusert i systemet, frem til patchen er utviklet og installert av sluttbruker. *Zero day attack* er angrep som utføres av trusselagenten før utviklerselskapet blir klar over sårbarheten i systemet. Ukjente sårbarheter er vanskelige å unngå, fordi det alltid vil være en risiko for at et program inneholder defekter som kan utnyttes. Det eneste man kan gjøre er å forebygge angrep ved å kontinuerlig lete etter og deretter fikse sårbarheter i programvaren.



Figur 1. Hentet fra Theel (2010)

Dersom det eksisterer en bestemt kombinasjon av trusler, sårbarheter og potensiell skade så innebærer det at det er knyttet en viss risiko til å ha systemet i bruk. Det er derfor viktig for alle institusjoner og virksomheter som forvalter, bruker og tilbyr tjenester ved hjelp informasjonssystemer, å arbeide bevisst med informasjonssikkerhet, slik at de kan ha tillit til at de tilbyr en tjeneste med tilstrekkelig sikkerhet for sine brukere. En måte som dette kan gjøres på er å innføre et *Information Security Management System* som gjør at man kan arbeide med informasjonssikkerheten på en systematisk måte. (Jøsang, 2017a, s.17) En viktig del av denne prosessen er risikovurdering, der man identifiserer alle risikoene og følger opp med konkrete tiltak eller *security controls* i den neste fasen. I sin artikkel; Reducing Internet-Based Intrusions: Effective Security Patch Management, argumenterer Bill Brykczynski og Robert A. Small for å i tillegg inkludere en systematisk *patch management process* som en del av organisasjonens information security management. De mener at en effektiv patch management process er et viktig tiltak for å forhindre angrep på informasjonssystemene fordi den sikrer at oppdateringene blir funnet, vurdert, testet og installert i tide og på denne måten fjerner mange av sårbarhetene fra systemet slik at de ikke lenger kan utnyttes. Dette er viktig fordi etter at sikkerhetsfiksen blir introdusert og distribuert, så kan risikoen for å bli utsatt for et angrep øke dersom man ikke oppdaterer fort nok: “The jump in number of attacks after patching is indicative of the fact that attackers think that users do not patch their systems quickly enough.“ (Arora, Nandkumar & Telang, 2006, s. 12)

Viktigheten av å oppgradere programvare reflekteres også i Mørketallsundersøkelsen (NSR, 2016, s.22) sine anbefalte sikringstiltak. Der dette blir nevnt under punktet for grunnleggende tiltak for mindre virksomheter:

- Oppgradere program- og maskinvare.
- Vær rask til å installere sikkerhetsoppdateringer.

Det at dette blir nevnt her viser at det å oppdatere i tide fortsatt kan være utfordrende, og ikke er helt etablert som allmenn praksis blant norske bedrifter. Spesielt for de små virksomhetene, som antageligvis mangler mye ressurser og kompetanse som skal til for å håndtere dette. De større bedriftene har nok mer bevissthet og rutiner på plass, men som vi skal se i punkt 5.2 så finnes det fortsatt mange utfordringer knyttet til dette.

2.2. Patch og Vulnerability Management

Ifølge NIST Special Publication 800-40, revisjon 3, ”Guide to Enterprise Patch Management Technologies,” er patch management prosessen av å identifisere, anskaffe, installere og verifisere patcher for produkter og systemer (NIST, 2013). Patcher er software oppdateringer beregnet på å fjerne svakheter eller defekter i programvaren, eller å tilføre nye funksjoner eller funksjonalitet. Programvareoppdateringer er ofte nødvendige for å fikse eksisterende problemer med programvare som blir lagt merke til etter den første utgivelsen. Mange av disse patchene har å gjøre med sikkerhet, mens andre kan adressere spesifikk funksjonalitet for programmer.

En tilnærming for patch management er å bruke en desentralisert modell der hver software pakke på hver enhet selv sjekker om oppdateringer er tilgjengelig og installerer den. En slik måte kan se ut som en enkel løsning på problemet, men det har også ulemper. Blant annet kan den være risikabel for større organisasjoner spesielt med tanke på krav om administratorrettigheter, uoversiktlig status for hvert program og høy bruk av båndbredde ved at hver enkel enhet skal laste ned samme patch, gjerne samtidig. En annen tilnærming, og den som blir ansett som den beste praksisen sett fra et sikkerhetsperspektiv, er *Centralized Patch Management* (Maymi & Harris, 2016, s.988). Det finnes flere måter dette kan gjøres på. Slik

som å ha en agent på hver enhet som kommuniserer med en oppdateringsserver, eller ved at serveren er autorisert til å sjekke enheten for oppdateringer.

Uansett hvilke av tilnærmingene som benyttes ønsker man å være proaktiv fremfor reaktiv og å være kjapp med å installere patchen, men først etter at den er testet grundig. Virtualiseringsteknologi gjør det mulig å sette opp en omgivelse der patcher kan testes, slik at man kan få replikert infrastrukturen og få testet patchen på et system som er representativt for tjenester og programmer. Det vil være en stor fordel å automatisere patching og regresjonstesting i så stor grad som mulig. Vi skal senere ta for oss to eksempler på kompleksiteten ved å ha en sårbarhet som finnes i mange kodebaser og ulempen ved dårlige praksiser og mangel på automatisering.

2.3. Typer oppdateringer

All programvare har versjonsnummer knyttet til seg. Dette for å holde styr på hvilken iterasjon av programmet som brukes og hvilke funksjoner og feilrettinger programvaren inneholder. Det vanligste er at programmet er representert med en serie med numre, separert med punktum. Der tallene til venstre representerer store oppgraderinger. Slike oppdateringer har gjerne også mer forbrukervennlige versjonsnavn, slik som at MacOS 10.12 blir kalt Sierra. Disse store oppdateringene kan gjerne koste penger, mens numrene til høyre indikerer ofte en mindre endring og er gratis, også kalt *point release*.

Ulike programvarehus og utvikler firmaer har forskjellige måter å distribuere software endringer på og hvordan disse blir omtalt. For å kunne beskrive disse på et generelt nivå uten at det blir spesifikt til et selskap skal vi dele inn software patchene i noen vanlige kategorier.

2.4. Hotfix

En hotfix er en mindre endring som retter et problem i softwaren som er kritisk nok til at den ikke kan vente til en senere patch. Den inneholder typisk ikke tidligere feilrettinger, men kun de filene som trenger å endres for å rette den spesifikke feilen. Andre navn på hotfix eller lignende begreper er *Quick Fix Engineering* (QFE) eller *Limited Distribution Release* (LDR) (Ballintijn, 2005, s. 5).

2.5. Point Release

Vi kan skille mellom en programvareoppdatering, ofte kalt en patch, og en programvareoppgradering som er en større endring eller forbedring enn den nåværende versjonen. En point release er typisk en gratis oppdatering som er ment for å løse problemer knyttet til funksjoner som ikke fungerer slik som de er ment, eller legge til mindre forbedringer i ytelse eller kompatibilitet. En slik oppdatering kan typisk inkludere driveroppdateringer som kan forbedre ytelsen av hardware eller forbedre kompatibilitet med nye modeller. Hensikten med disse oppdateringene er å adressere feil, eller en oppsamling av feilrettelser, heller enn å introdusere nye funksjoner. Ofte trengs det å ryddes opp etter en stor oppdatering med nye funksjoner. Andre navn kan være *minor release* eller *maintenance release*.

I større programvare applikasjoner som operativsystemer og databasesystemer er det vanlig å lansere en service pack 10-20 måneder etter produktlansering. Det gjøres hovedsakelig fordi det er lettere og minsker sannsynligheten for feil sammenlignet ved å installere individuelle patcher. Samtidig gjør det arbeidet med oppdatering av datamaskiner over store nettverk mer håndterbart.

2.6. Security Patch

En sikkerhetspatch er en endring som gjøres for å korrigere en svakhet som er beskrevet som et sikkerhetsproblem. Målet er at det korrigerende tiltaket vil hindre vellykket utnyttelse og fjerne eller redusere trusselens evne til å utnytte den bestemte sårbarheten. Avhengig av programvareutvikleren kan sikkerhetsproblemer kategoriseres i forskjellige nivåer av alvorlighetsgrad. Basert på alvorlighetsgraden vil utvikleren utstede sikkerhetsoppdateringer. Microsoft deler for eksempel sikkerhetsoppdateringene inn i kritisk, viktig, moderat eller lav. Der nivåene prøver å reflektere hvor vanskelig det er utnytte sårbarheten sett i sammenheng med påvirkningen av utnyttelsen. Andre kategoriseringssystemer som SANS legger større vekt på om sårbarheten er funnet i standardkonfigurasjonen, mens CERT/CC lager en numerisk verdi og vektlegger faktorer slik som internett infrastruktur (Mell, Scarfone & Romanosky, 2007).

2.7. Uoffisiell Patch

En uoffisiell patch er ofte en ikke-kommersiell patch laget til programvaren av en tredjepart istedenfor den originale utvikleren. I likhet med en hvilken som helst annen patch er målet å fikse feil eller mangler i programvaren. Eksempler på dette er sikkerhetsfikser gjort av spesialister for å rette feil som er så kritiske at den originale utvikleren bruker for lang tid. Andre eksempler er patcher som sørger for kompatibilitet for et produkt som blir ignorert eller ikke lengre er støttet av den originale utvikleren. I kapittel 3.1 skal vi se nærmere på et prosjekt som patchet en kritisk sårbarhet i flere tusen open source prosjekter.

3. Eksempler på trusler

Ved manglende oppdatering er det en sjanse at noen bruker sårbarhetene til å infisere datamaskinene med virus, malware eller annet skadevare. For eksempel kan man få tilsendt en en infisert PDF eller Word fil, som utnytter en sårbarhet i systemet når dokumentet åpnes. Avsenderen kan være ukjent eller de kan gi seg ut for å være noen som man kan stole på, som for eksempel: Bank, Skatteetaten eller lignende. Ifølge en undersøkelse gjort av Symantec (Selvaraj & Gutierrez, 2010, s. 3) ser det ut som at økningen i infiserte PDF dokumenter har vokst i takt med utbredelsen av filformatet, og hvor mange som bruker det. Noe som kan tolkes som at jo flere bruker en teknologi, jo mer sannsynlig er det at noen vil prøve å finne sårbarheter i den:

> The amount of malicious PDFs seen in the wild has increased dramatically over the last 3 years. This is due to the success that malware authors are attaining via PDF distribution. The threat landscape is not homogenous in that there are many different types of PDFs and different ways in which malicious PDFs are used to compromise computers. To really understand the PDF threat landscape we need to discuss different methods of distribution for these malicious PDFs as well as the different types of PDFs that are being seen in the wild. (Selvaraj & Gutierrez, 2010, s. 2)

Et eksempel på når ting går galt er ormen "Code Red" i 2001 som infiserte Microsoft sin software Internet Information Services (IIS) som ofte brukes til å verte nettsider. Denne ormen brukte den infiserte maskinen til å kjøre en packet-flooding denial of service angrep mot forhåndsinnstilte IP-adresser. Da ormen ble oppdaget ble det allerede anslått rundt

25.000 infiserte maskiner (CERT - Carnegie Mellon University, 2002) En annen relatert orm var Nimda som ble oppdaget i September 2001, denne ormen sendte e-poster som virket tom men egentlig kjørte en binær fil som ble åpnet uten at brukeren la merke til dette. (CERT - Carnegie Mellon University, 2001)

3.1. Operation Rosehub

Tidlig i 2015 oppdaget Foxglove Security en feil som ble døpt ”Mad Gadget”. Den utnytter en sårbarhet i mye brukte nytte-klasser i *Apache Commons Collections* som håndterer *deserialization* – en måte å konvertere data fra et format til et annet og transportere det på tvers av nettet. Feilen åpner opp for *Remote Code Execution*, som betyr at angriperen kan være hvor som helst i verden og få tilgang til maskinen. Etter at sårbarheten ble kjent med et proof-of-concept tok alle de store firmaene som Apache, Oracle, Cisco, Intel, HP og IBM affære og tettet sårbarheten raskt. Men ulikt de store selskapene så har ikke open source prosjekter dedikerte ansatte som følger med kontinuerlig, men baserer seg isteden på hjelp fra frivillige for å informere dem. Det skulle gå enda fem måneder før en Google ansatt tilfeldigvis oppdaget at flere fremstående open source prosjekter enda ikke hadde fikset feilen. På den tiden hadde sårbarheten blant annet vært årsak til at mer enn 2000 datasystemer i San Francisco metroen ble infisert med Ransomware. Blant annet rammet de betalingsløsninger og informasjonssystemet til rutetabellene. Rent praktisk førte dette til at store deler av kollektivsystemet ble satt ut av drift og påvirket hundretusener av mennesker. Noe som førte til at rutetabeller ble skrevet for hånd og klistret opp på oppslagstavler (Khandelwal, 2016).

Omfanget av sårbarheten var blitt så stort at det tok en arbeidsgruppe på 50 ansatte i Google 20% av tiden deres i flere måneder å lage patche de over 2600 sårbare prosjektene som de klarte å spore opp på GitHub. Dette skiller seg fra uoffisiell patch ved at kildekoden ligger åpent tilgjengelig og at utviklerne bidrar til å rette feil i prosjektet, fremfor å selv distribuere en egen fiks. Selv om arbeidsgruppen klarte å minimere skadene fra denne spesifikke Java sårbarheten er det nærliggende å tro at den fremdeles finnes i mange uadresserte open source og closed source prosjekter. En studie fra Veracode i 2016 estimerer at 97% av Java applikasjoner inneholder minst én alvorlig kjent sårbarhet (Veracode, 2016, s. 2). Dette er

med på å understreke viktigheten av å holde programvare oppdatert og å ha gode rutiner for og rette sårbarheter etter hvert som de blir kjent.

3.2. Knight Capital

Et eksempel på en gang det virkelig gikk galt er historien om Knight Capital. Et selskap som mistet 172 222 dollar hvert eneste sekund i 45 minutter, som følge av dårlig Patch Management rutiner og prosedyrer. Det totale tapet ble estimert til totalt 465 000 000 dollar og slo effektivt et selskap med flere hundre millioner dollar i formue konkurs på under én time (SEC, 2013).

I 2012 var Knight den største aktøren innen amerikanske aksjer med en markedsandel på rundt 17%. Knight's Electronic Trading Group (ETG) hadde et gjennomsnittlig daglig handelsvolum på mer enn 3,3 milliarder, og håndterte transaksjoner for over 21 milliarder dollar daglig. Selve årsaken til feilen er teknisk og ikke så viktig her, men i mellom 27. juli og 31. juli 2012 distribuerte Knight manuelt en ny programvare til et begrenset antall servere per dag – totalt åtte servere i det hele. Etter det amerikanske børskrakket i 1929 ble Securities And Exchange Commission (SEC) opprettet, blant annet for å beskytte investorer og sikre god drift av verdipapirmarkedene. Dette er hva SEC-rapporten sier om den manuelle distribusjonsprosessen:

> During the deployment of the new code, however, one of Knight's technicians did not copy the new code to one of the eight SMARS computer servers. Knight did not have a second technician review this deployment and no one at Knight realized that the Power Peg code had not been removed from the eighth server, nor the new RLP code added. Knight had no written procedures that required such a review.
>
> [...]
>
> Knight did not have an adequate written description of its risk management controls as part of its books and records in a manner consistent with Rule 17a-4(e)(7) of the Exchange Act, as required by Rule 15c3-5(b).
>
> (SEC Filing | Release No. 70694 | 2013, 16. oktober)

Hendelsen med Knight Capital kan være en leksjon for alle utviklings- og driftsgrupper. Det er ikke nok å bare lage god programvare og teste den. Det er også nødvendig å sørge for at

den leveres riktig og at det er gode rutiner for å fange opp feil som måtte oppstå. Slik at kundene får verdien og sikkerheten som er nødvendig, og for å unngå uheldige konsekvenser som konkurs. Ingeniørene som distribuerte systemet hadde ikke skylden alene, i dette tilfellet var ikke prosessen selskapet hadde satt opp hensiktsmessig for å møte risikoen selskapet ble utsatt for. I tillegg var prosessen deres iboende utsatt for feil, fordi når distribusjonsprosessen er avhengig av at mennesker leser og følger instruksjoner, oppstår en risiko for menneskelig feil. Feilene kan forekomme i tolkningen, eller i utførelsen av instruksjonene.

Det vil være en fordel om patchingen kan automatiseres og repeteres, for å begrense så mange potensielle menneskelig feil som mulig. Hadde Knight Capital implementert et automatisert distribusjonssystem - komplett med konfigurasjon, distribusjon og testautomatisering, kunne feilen som forårsaket den katastrofale feilen muligens vært unngått (SEC, 2013). Et par prinsipper for kontinuerlig levering gjelder her: (1) Utgivelse av programvare skal være en repeterbar, pålitelig prosess. (2) Automatiser så mye som er rimelig.

4. Formidling av sårbarheter

Hvordan en sårbarhet formidles etter at den er oppdaget er et spørsmål som kan få konsekvenser for sikkerheten til det gjeldende systemet. Den som oppdager sårbarheten må ta stilling til hvordan denne sårbarheten skal formidles videre. Viktige spørsmål da blir ifølge Goldschmidt et al. (2010, s. 15); når skal informasjonen formidles, på hvilken måte, hva skal formidles og til hvem? Dersom vedkommende velger å publisere all informasjonen om sårbarheten i et offentlig forum, både hvordan man finner og utnytter denne, så kalles det *full disclosure*. Et annet alternativ er *responsible disclosure*, hvor den som finner sårbarheten varsler de som er ansvarlige for den gjeldende softwaren og arbeider sammen med dem for å løse problemet. Sårbarheten blir først offentliggjort når det foreligger en patch. (Goldschmidt et al., 2010, s. 15)

Tilhengerne av full disclosure mener at å legge ut all informasjonen med en gang vil føre til bedre sikkerhet for alle, fordi det blir lagt et større tidspress på alle aktører, og dermed redusere the window of exposure. Dersom det foreligger en exploit, vil utviklerne ifølge

denne argumentasjonen i større grad bli presset til å utvikle en patch raskt, for å unngå dårlig publisitet. Dersom de ansvarlige ikke blir tilstrekkelig presset frykter tilhengerne av full disclosure at utviklerfirmaene skal prioritere å legge til inntektsgivende features og fikse bugs i programvaren, fremfor å utvikle en sikkerhetspatch. Brukerne av systemet vil også bli tvunget til å oppdatere tidligere for å ikke bli utnyttet. I tillegg vil en full disclosure gi nødvendig informasjon til andre aktører som ønsker å utvikle andre typer forsvar mot exploiten, som for eksempel brannmurer, antivirus, software og intrusion detection systems (IDS). Dette kan være en fordel fordi det i noen tilfeller kan dette ta mindre tid enn å fikse feilen i selve programvaren. (Goldschmidt et al., 2010, s. 15)

Motstanderne av full disclosure, det vil si de som argumenterer for responsible disclosure mener at det er uansvarlig legge ut informasjon om sårbarheten før det finnes tilgjengelige forsvarsmekanismer for brukerne. Et annet argument mot full disclosure er at informasjonen som gjøres tilgjengelig i denne potensielt kan føre til at angrepene blir sterkere. Dette fordi informasjonen som finnes i patchen kan gjøre trusselagentene i stand til å utvikle bedre exploits. (Goldschmidt et al., 2010, s. 15). En studie gjennomført av Arora, Nandkumar og Telang (2006) fant resultater som kan tyde på at dette er tilfelle, og understreker dermed viktigheten av å tenke nøye igjennom hvordan patchene skal utgis. Det kan være at de som ønsker å legge ut informasjon om sårbarheter ikke alltid har brukernes beste i tankene, men derimot er ute etter å promotere seg selv eller tjene penger på salg av sikkerhetsevalueringer. Et annet argument for responsible disclosure er at dette kan gi utviklerselskapene bedre tid til å utvikle og teste patchen og dermed øke kvaliteten på denne. I tillegg så foretrekker ofte store bedrifter og organisasjoner at det ikke blir utgitt patcher for ofte, da dette medfører økte kostnader knyttet til å distribuere og installere disse (Goldschmidt et al., 2010, s. 16).

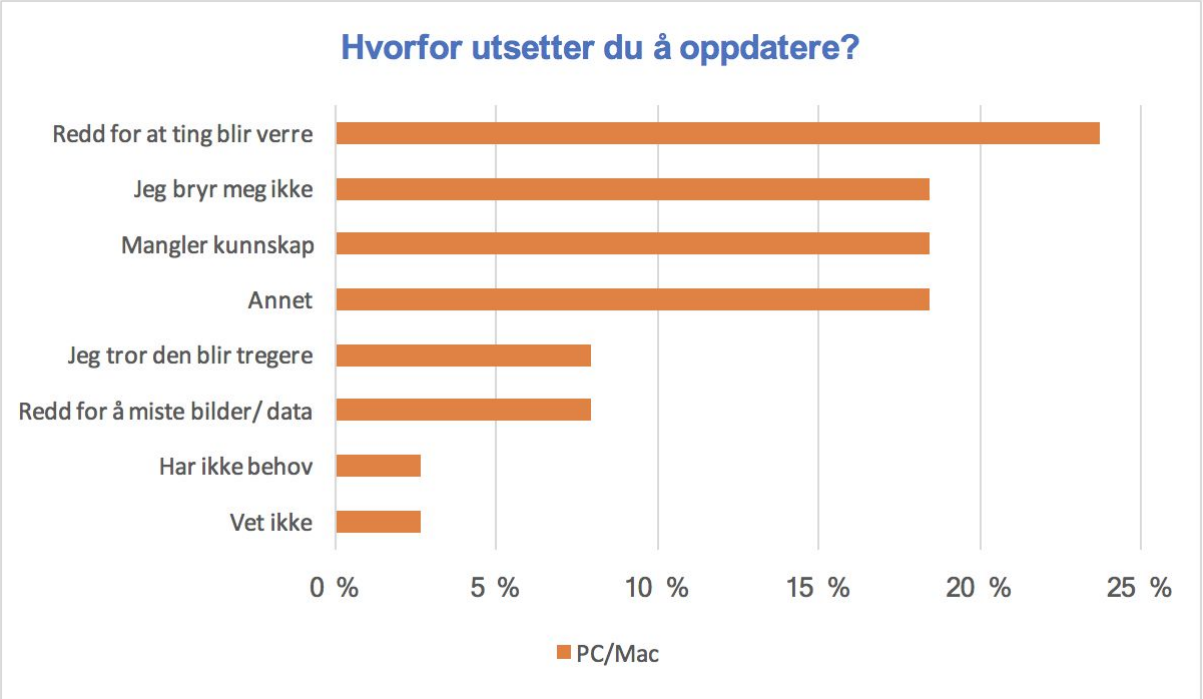
I tillegg til full disclosure og responsible disclosure finnes det noe som heter silent disclosure. Det er når utviklerbedriften inkluderer patchen i programvareoppdateringen uten å gi brukerne informasjon om dette. Fordelen med denne formen for disclosure er at et stort antall enheter og systemer kan oppdateres automatisk, og at ingen informasjon gis til de som ønsker å bruke patchen til å utvikle exploits. Motstanderne kritiserer denne formen for at den fratar brukerne kontroll over sine miljøer, da disse oppdateringene ikke alltid er av like god kvalitet og kan føre til at systemer ikke fungerer som det skal.

Hvordan den som finner sårbarheten best skal gå frem for å formidle denne viser seg å være et sammensatt spørsmål som det er vanskelig å gi noe tydelig svar på. Hva som er den beste løsningen; full, responsible eller silent disclosure avhenger av situasjonen, hva slags sårbarhet det er snakk om og hvor kritisk det er å få den fikset.

5. Oppdatering av programvare; vaner, rutiner og utfordringer

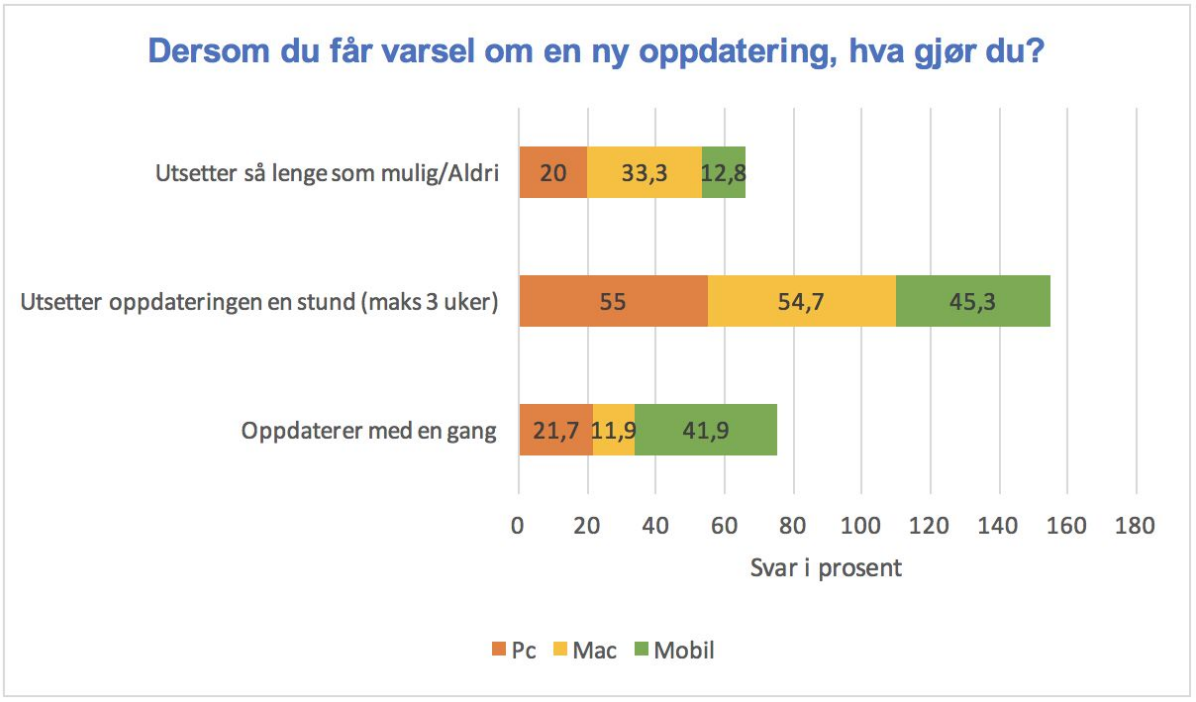
5.1. Privatpersoner

I forbindelse med vår oppgave har vi gjennomført en spørreundersøkelse om vaner knyttet til oppdatering av programvare på PC/Mac og mobil. Vi var interessert i å finne ut av hvilke vaner privatpersoner har knyttet til dette, og deres bevissthet om koblingen mellom oppdateringer og informasjonssikkerhet. I tillegg ønsket vi også å finne ut om de har hørt om de vanligste sikkerhetstrusselene. Vi fikk inn 86 svar. Av de som svarte på undersøkelsen er 55% under 30 år, mens 45% er over. 58% av respondentene er i arbeid, 37% studerer, mens de resterende 5% faller inn under kategorien annet. Når det kommer til interesse i datateknologi svarer 45% av respondentene at de er svært interessert, mens 35% er litt eller mindre interessert. I vår undersøkelse har vi valgt å bruke kvantitative metoder for å samle inn statistikk for å få et inntrykk av massen, det var ikke nødvendig for oss å gå i dypere detalj i form av kvalitative studier for å få til dette. For å finne respondenter til spørreskjemaet benyttet vi oss av convenience sampling ved at vi sendte spørreskjemaet ut til alle aldersgrupper via e-post, sosiale medier, sms og lignende. Undersøkelsen var anonym, og tok ca. 4-6 minutter å fylle ut. Undersøkelsen har kun tatt høyde for datamaskiner og mobiler, ikke tablets. Dette fordi vi så at å inkludere dette gjorde spørreskjemaet unødvendig komplisert og repetitivt, noe som kunne føre til at respondentene ikke fullførte.



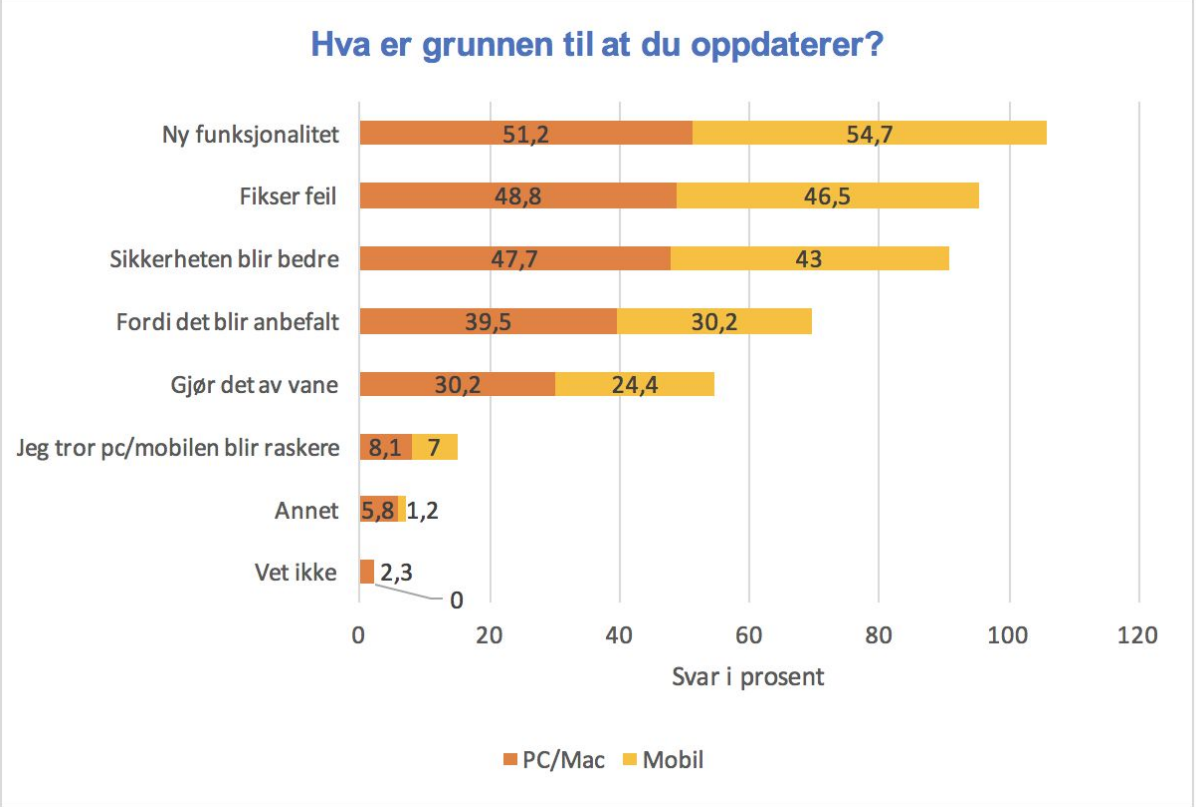
Figur 2.

I følge vår undersøkelse, har vi blant annet funnet at den største grunnen til at folk utsetter å oppdatere datamaskinen sin er ”Det tar tid / Passer ikke” med 87,2%. Figuren over viser svaralternativer utenom “Det tar tid / Passer ikke”, for å gi et inntrykk av hvilke andre grunner til at personer utsetter oppdatering. De vanligste grunnene at respondentene utsetter oppdateringer, med unntak at det ikke passer, er at de er redd for at ting blir verre, at de ikke bryr seg for lite til å orke å oppdatere og at de føler at de mangler kunnskap.



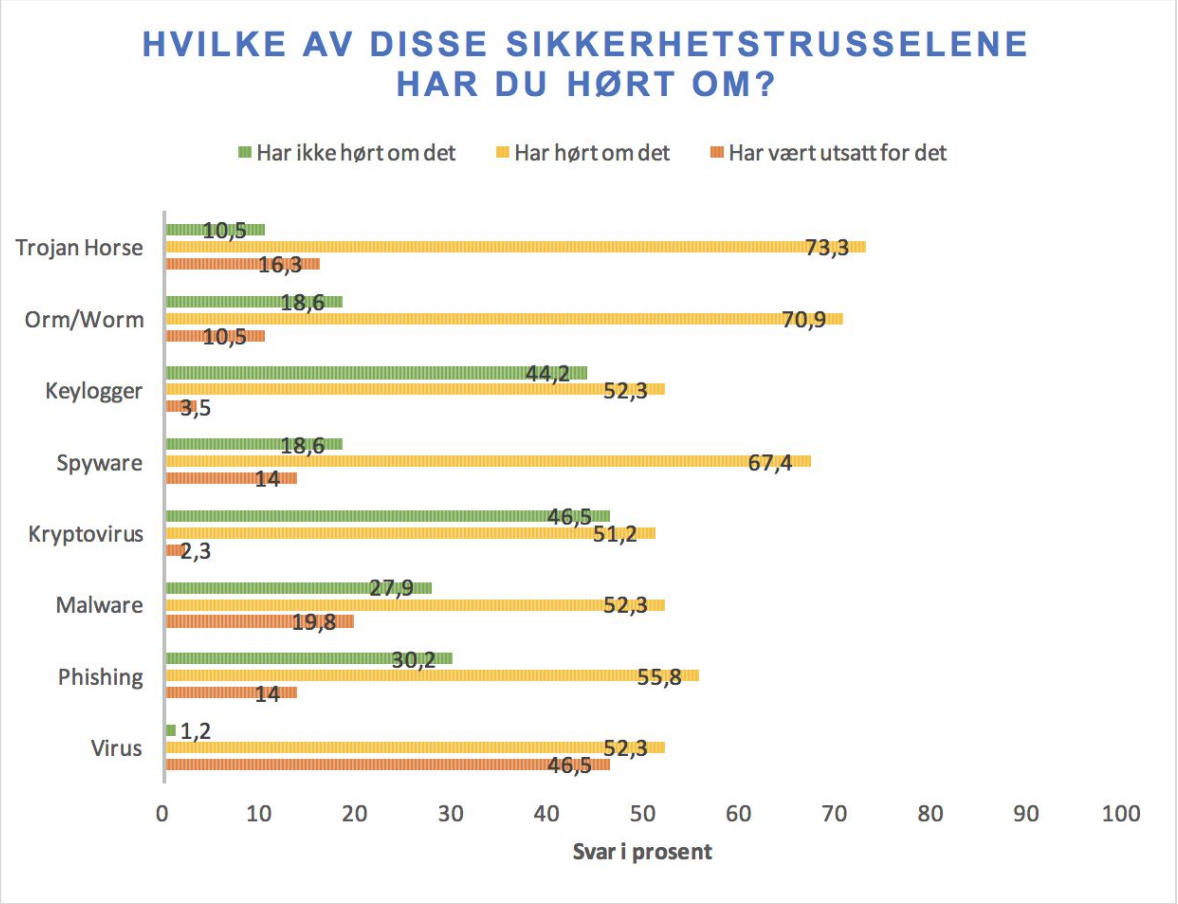
Figur 3.

Det var ikke datagrunnlag for å si noe om hvorfor man utsatte å oppdatere mobile enheter, fordi de aller fleste oppdaterte med en gang eller innen rimelig tid, slik som grafen over viser.



Figur 4.

I hovedsak viser undersøkelsen at privatpersoner er bevisst på at de skal oppdatere enhetene sine, men ikke nødvendigvis med en gang, da majoriteten utsetter en stund til det passer. Sett i en sikkerhetssammenheng kan tre uker være nok tid til at en enhet blir angrepet av en sårbarhet, spesielt hvis det er en sårbarhet som er offentliggjort, slik som forklart om window of exposure i kapittel 2.1.

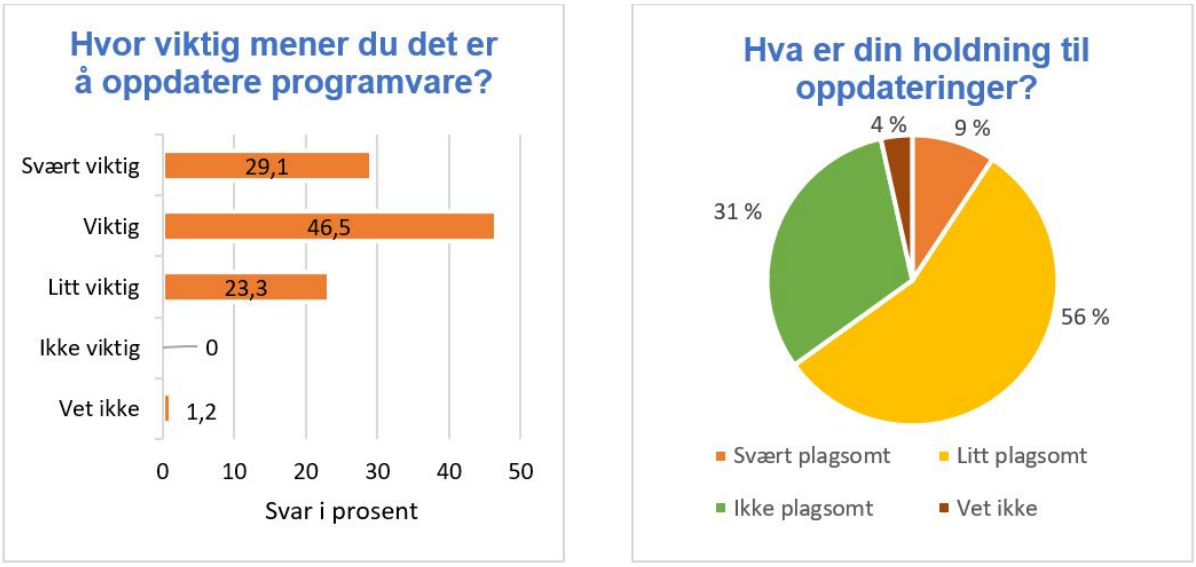


Figur 5.

Svaralternativene om kjennskap til sikkerhetstruslene er med vilje veldig like eller underbegreper, meningen med dette spørsmålet var å teste kjennskap til de ulike sikkerhetstruslene og eventuelt om de selv har opplevd dette.

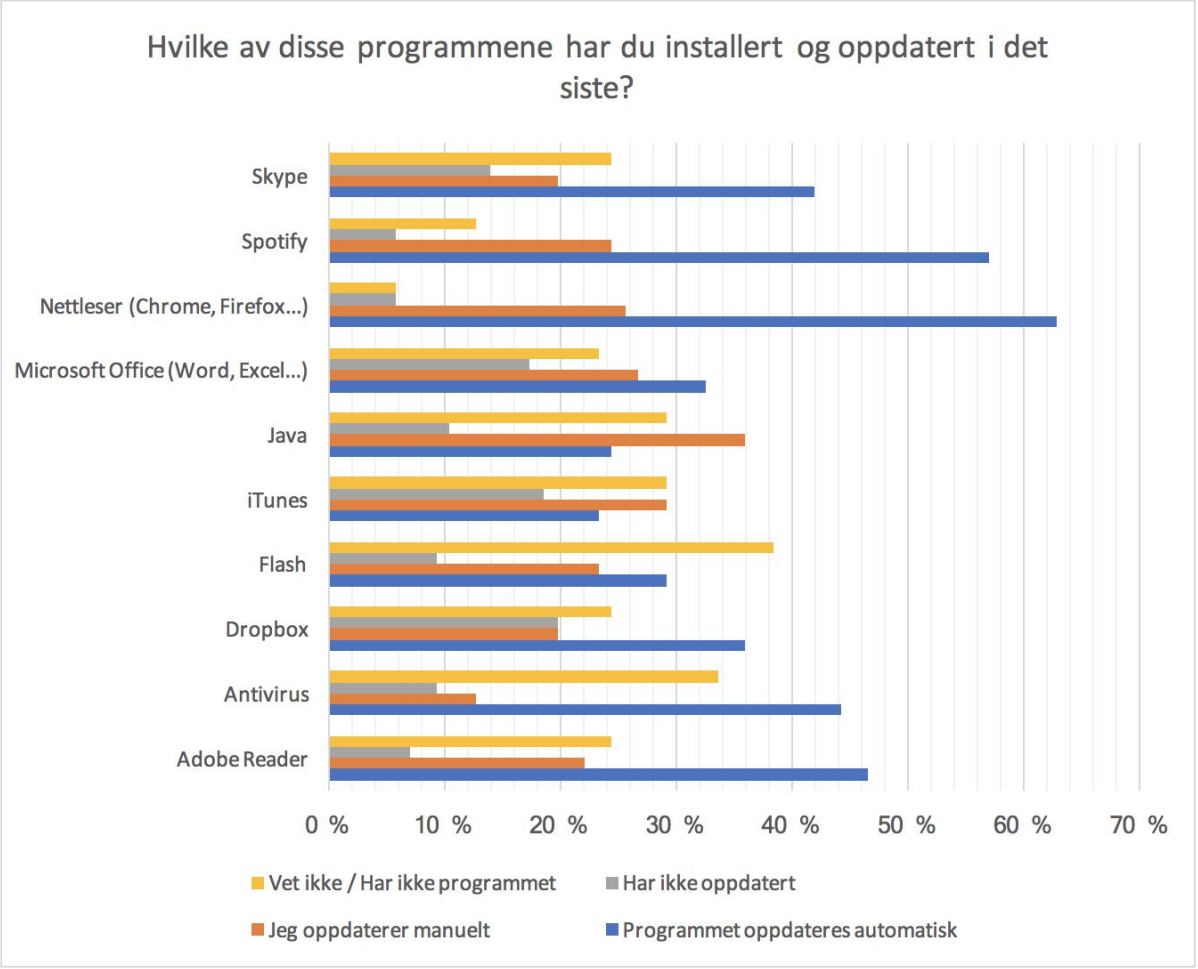
Vi har undersøkt mer i detalj om personene som har oppgitt at de har blitt infisert av datavirus i fortiden har blitt flinkere til å oppdatere i ettertid. Svaret fra disse utvalgte respondentene på spørsmålet "Hvor viktig mener du det er å oppdatere programvare?" er 5%

Vet ikke, 24% Litt viktig, 43% Viktig og 29% Svært viktig. Av respondenter som ikke har blitt infisert av datavirus i fortiden er resultatene på samme spørsmål: 24% litt viktig, 38% viktig og 38% svært viktig. Dermed er det ikke stor forskjell mellom datainteresserte og ikke-datainteresserte infisert av datavirus.



Figur 6. Svar fra alle respondenter.

Vi har også målt hvor mange personer har følgende programmer, og deres oppdateringsvaner tilknyttet til dette. Vi har spurt om de mest kjente programmene, men også programmer med nåværende eller tidligere kjente sårbarheter. Figur 7 viser resultatet fra dette. Det som er interessant å se i svarene er hvordan personer krysser av at de oppdaterer manuelt programmer som har innebygget automatisk oppdatering, dette kan betyr to ting: Enten forstår ikke brukeren at programmet oppdaterer seg selv, eller de bevisst har gått i innstillinger og skrudd den funksjonaliteten av. Det er interessant å se at 12% av respondentene for eksempel manuelt oppdaterer antivirus, selv om det er ikke presisert om det er snakk om selve programvaren eller en definisjonsoppdatering. Eller nesten 30% vet ikke om Java oppdateres automatisk eller manuelt, som er et program kjent for mange sårbarheter gjennom tidene. Vi kan ikke se på disse resultatene ukritisk, fordi respondentenes definisjoner på "oppdaterer manuelt" og "oppdaterer automatisk" kan være forskjellige. Spesielt i tilfeller som Spotify som oppdateres automatisk, men du må selv starte programmet på nytt for å fullføre oppdateringen, vil dette være automatisk eller manuelt?



Figur 7.

Vi har sett på forskjeller mellom personens oppdateringsvaner og operativsystemet de kjører, og her fant vi lite forskjell. De aller fleste oppdaterer innen tre uker, uansett operativsystem. På mobil fant vi at det er androidbrukerne som er tidligst ute med å oppdatere, mens flere iOS brukere utsetter det så lenge som mulig.

Oppsummert virker det som privatpersoner i hovedsak er mye flinkere til å oppdatere enn vi hadde sett for oss før vi gjennomførte undersøkelsen. Det finnes fremdeles et forbedringspotensiale, men programmenes implementasjon av automatiske oppdatering har nok spilt en stor rolle i å luke ut sikkerhetshull og hjulpet brukerne med å holde seg oppdatert, særlig på enheter som eies av personer med mindre teknologisk interesse. Dersom det ikke er mulig å gjøre det automatisk for brukeren, burde programvaren være designet slik at det er så friksjonsfritt for brukeren som mulig. Hvis det er en større oppdatering kan det være lurt at brukeren er informert på forhånd. Ny funksjonalitet kan være omtalt i både

pressemeldinger og media lenge før det blir sluppet. Dermed forventer også brukeren oppdateringen, som igjen bidrar til å senke terskelen for å oppdatere. Det burde være en robust løsning som ikke forstyrrer eller avbryter brukeren i arbeidet sitt mer enn nødvendig (Forbes Technology Council, 2017).

Å oppdatere burde være så risikofritt som mulig og oppdateringen burde testes grundig før den slippes til massene. Det kom frem i undersøkelsen at flere personer var engstelige for forandringer og hadde negative opplevelser knyttet til oppdateringer, som vist i figur 2: ”Jeg er ikke redd for at ting blir verre, jeg VET ting blir verre”

5.2. Utfordringer i organisasjoner

Tidligere var det vanlig og se på informasjonssikkerhet bare som et teknisk anliggende, og en del av IT-avdelingens ansvarsområde. I dag har det kommet lenger opp på agendaen hos flere ledergrupper, fordi man ser nødvendigheten av å inkludere dette når man utformer den generelle strategien for bedriften. Informasjonssikkerhet må tas hensyn til og sees i sammenheng med forretningsmål der fokuset er at det skal leveres verdi for kunder og eiere. For å kunne gjøre dette er det viktig for ledelsen og ha en viss innsikt i risikoene knyttet til informasjonssikkerhet, slik at de kan ta informerte avgjørelser, fordele ressurser på en fornuftig måte og overvåke denne. (Goldschmidt et al., 2010, s. 10)

Bedrifter vil derfor ha forskjellige retningslinjer for hvordan IT-ressurser skal behandles, sikres og oppdateres. Men selv om det er mer fokus på informasjonssikkerhet og risiko enn det var tidligere, så er det fortsatt slik at en stor andel bedrifter ikke har utformet eksplisitte retningslinjer for hvordan de skal arbeide med informasjonssikkerhet. Modenheten i håndteringen av informasjonssikkerhet befinner seg da på et sted mellom nivå 0-2 i forhold til Capability Maturity Model (CMM) (Jøsang, 2017a, s. 26). Ofte er det opp til den ansatte å gjøre de nødvendige sikkerhetstiltakene, som for eksempel å holde datamaskinen oppdatert. Da har bedriften liten eller ingen oversikt over hva som blir gjort eller ikke gjort av tiltak. Tvert imot kan tenkes at mange ansatte bruker jobbmaskinen sin til aktiviteter som øker sikkerhetsrisikoen for bedriften. Det kan for eksempel tenkes at de bruker maskinen på fritiden til å laste ned filmer, installerer ukjent programvare, besøker usikre nettsteder, eller lar andre personer bruke datamaskinen. Dersom ingen har blitt tildelt et ansvar, så er det også

vanskelig å vite hvem man skal stille til ansvar hvis det skulle oppstå en uønsket situasjon. Det er derfor viktig at opplæring og bevisstgjøring av ansatte er en del av tiltakene som blir gjort for å styrke informasjonssikkerheten.

I løpet av 2018 vil også en ny oppdatert EU-lov om informasjonssikkerhet tre i kraft som vil gi mulighet å bøtelegge selskaper som slurver med sikkerhet og persondata. Bøtene vil prise seg opp til 4% av selskapets globale omsetning eller 20 millioner euro, av det beløpet som er størst. Dette er et tiltak som iverksettes for å få flere bedrifter til å fokusere mer på informasjonssikkerheten. (Wakefield, 2016)

Når det kommer til oppdatering av programvare så er dette en viktig del av arbeidet som organisasjoner må gjøre for å holde systemene sine sikret fra angrep. Patch management som vi har vært inne på tidligere er også ifølge Goldschmidt et al (2010, s. 11) en viktig del av arbeidet med informasjonssikkerhet. Gjennom en effektiv distribuering av patcher demper man risikoen for at systemet kan utnyttes. I tillegg til dette mener Goldschmidt et al (2010, s. 11) at organisasjonene må sørge for å overvåke at distribusjonen er vellykket og at alle maskiner mottar og installerer oppdateringene, slik som vi så mangel på i eksempelet med Knight Capital. Alle nye maskiner må bygges med de siste oppdateringene og eksterne maskiner som skal kobles på nettet må være i samsvar med organisasjonens retningslinjer.

I følge Theel (2015) kunne store deler av internettangrep vært unngått med en mer proaktiv *patch management prosess*. Vi har funnet flere undersøkelser som viser til at angrep kunne vært unngått dersom programvaren hadde vært oppdatert. For eksempel skriver Brykczynski og Small (2003, s. 51) i sin artikkel at The Gartner Group rapporterer at ca 90 prosent av "vellykkede" Internet-based angrep utnyttet programvare som ikke var riktig konfigurert eller oppdatert. En annen undersøkelse, som ble underbygget av data som ble samlet inn av HP sikkerhetsteam i 2014, viste at 44 prosent av innbruddene skyldtes sårbarheter som kunne vært fikset, og som var mellom to og fire år gamle (Kerner, 2015). En tredje undersøkelse, gjennomført av BMC og Forbes Insight, antageligvis i 2016, fant at 44 prosent av innbruddene fant sted etter at sårbarhetene og løsningene var identifisert (Williams, 2016) Dette er angrep som kunne vært unngått dersom patchene hadde blitt installert i tide.

Grunnene til at de ikke blir installert i tide er mange. I følge Williams (2016) skyldes dette at ledelsen har vanskeligheter med å prioritere hvilke av systemene som skal fikses først, fordi sikkerhetsteamet og operasjonsteamet har ulike prioriteringer og liten eller generell forståelse av den andres krav. I Kerners artikkel (2015, s. 1) forklarer Jewel Timple fra HP Security at patching er vanskelig av flere grunner. Ofte er distribueringen og installeringen av oppdateringer et svært ressurskrevende arbeid. Det kan for eksempel være at det er et stort antall patcher som skal distribueres utover et stort system, og organisasjonene må også samtidig forsikre seg om at ingenting blir ødelagt som følge av oppdateringene. I sin artikkel nevner Theel (2015, s. 1) disse utfordringene knyttet til patch management; manglende forståelse av hvordan man skal patche, mangel på personell, frykt for å skape uheldige konsekvenser for bedriften, begrensninger som følge av network-bandwidth, vanskeligheter i forhold til å håndtere store og komplekse systemarkitekturer, at de bruker lang tid på å fikse systemene dersom arbeidet med å installere patchen skulle ødelegge noe, scalability issues.

Selv om det virker selvsagt at man skal oppdatere systemene sine så tidlig som mulig, så kan det virke som om det er lettere sagt enn gjort, fordi det er mange faktorer som gjør at dette er utfordrende å gjennomføre i praksis. Studien som det blir referert til i Theels artikkel fant at det var vanlig for enkelte organisasjoner å bruke opp til 200 dager på å installere patchen etter at den ble gjort tilgjengelig. I tillegg til grunnene nevnt ovenfor kan dette også skyldes at man trenger tid til å teste patchen, men kan også være på grunn av manglende støtte eller forståelse for at det bør gjøres raskt. I tillegg kan det å vente med å installere patchen være en bevisst strategi for bedrifter som ikke har budsjett og ressurser til å teste denne selv, da har de en mulighet til å se om de som innfører den opplever problemer med den, før de selv tar risikoen som på å installere. Ulempen med dette er at alle disse forsinkelsene gjør at vinduet for å kunne utnytte sårbarhetene blir større, noe som igjen kan føre til økt motivasjon blant hackere til å lage exploits.

Bedrifter har også utfordringer med at deres infrastruktur som må oppdateres jevnlig, særlig ved bruk av Windows Servere eller lignende. Microsoft lanserer oppdateringer hver andre tirsdag i måneden, for å unngå søndagen i noen tidssoner, og samtidig gi organisasjonene god nok tid for å oppdatere systemene slik at sikkerhetshullene blir tettet før helgen. Noe som er et bevisst valg fra Microsoft, når de fleste store angrep blir utført innen timer fra lanseringen

av oppdatering. Ulempen med denne ordningen er at dette også gjør det mulig for hackerne å planlegge sine angrep, derav begrepet Exploit Wednesday). (Goldschmidt, Dark & Chaudhry, 2010, s. 12-14).

I sin artikkel argumenterer Brykczynski og Small (2003) for å inkludere en systematisk patch management prosess i bedriftens Information Security Management system. De beskriver åtte nøkkelpraksiser ut fra en *life cycle view*. Disse kan sies å overlappe noe med standarden NIST Special Publication 800-40, revisjon 3, men har noen flere punkter som kan være nyttige.

1. Establish policies, procedures, and responsibilities
2. Maintain awareness of IT infrastructure
3. Maintain vulnerability alert resources
4. Monitor vulnerability alerts
5. Assess and respond to vulnerability alerts
6. Test and evaluate patches
7. Install patches
8. Measure and improve the process

Implementasjonen vil variere fra bedrift til bedrift, men prinsippene som følges er de samme. Ved å arbeide med patch management på denne måten vil bedriftene få en tydelig struktur på prosessen, noe som vil kunne stimulere de ansatte til å levere bedre (Brykczynski & Small, 2003, s. 57). Ved å innføre et målesystem vil det være lettere å kommunisere mer objektive resultater til interessentene og ledergruppen, noe som vil styrke beslutningsgrunnlaget og gjøre det lettere for dem å involvere seg i prosessen.

6. Oppsummering

Generelt kan man si at det er utviklerne av programvaren som skal holdes ansvarlig for sikkerheten og funksjonaliteten til programvare (Goldschmidt et. al., 2010, s. 6). En måte å ivareta dette ansvaret på kan være å inkludere oppdateringsprosessen som en del av kravspesifikasjonene når programvaren blir laget. Forbes Technology Council (2017) anbefaler at dette blir gjort, fordi oppdateringer og nye leveranser av programvaren er en

essensiell del av systemutviklingsprosessen og burde dermed ha en sentral rolle i livssyklusen til programmet. Mangel på planer kan føre til frustrerte brukere eller at det går galt, slik som vi så med Knight Capital, og det kan føre til merarbeid for teknikere og supportmedarbeidere. Brukerne liker ikke negative overraskelser, som for eksempel tap av funksjonalitet eller kompatibilitetsproblemer. Etter hvert som ny software blir tilgjengelig vil det bli aktuelt å fase ut gammel programvare. Det kan være både tidkrevende og frustrerende å støtte eller henge bakpå med gammel programvare. En måte er å tvinge brukerne over på ny programvare ved å ikke gi dem noe valg (Forbes Technology Council, 2017).

Bedrifter og organisasjoner som benytter seg av programvare har på sin side et ansvar for å installere oppdateringene fortløpende, som en del av sin *information security management process*, for å unngå og bli utnyttet. Dette betyr at bedriftene bør tenke igjennom, planlegge, implementere og evaluere patch management i bedriftens arbeidspraksis. Her kan standarder eller retningslinjer, som for eksempel NIST Special Publication 800-40, revisjon 3, ”Guide to Enterprise Patch Management Technologies, eller de 8 nøkkelpunktene fra artikkelen til Brykczynski og Small (2003) være til god hjelp når man skal implementere prosessen i bedriften. Når det kommer til privatpersonene så var de flinkere med oppdatering enn våre antagelser som var at folk flest er i hovedsak ikke særlig bevisste på å oppdatere. Vi har i kapittel 5.1 sett på en del tiltak som utviklerselskapene kan gjennomføre for å lette oppdateringene for privatpersoner. I tillegg kan det også være et aktuelt tiltak å innføre mer opplæring i grunnleggende informasjonssikkerhet som en del av skolegangen, der bevissthet rundt oppdatering kan bli inkludert.

Gjennom vår undersøkelse har vi sett at oppdatering av programvare knyttet til informasjonssikkerhet er en prosess som flyter mellom flere aktører. For å ivareta sikkerheten er man avhengig av at disse samarbeider og tar sin del av ansvaret for å bidra til at sikkerheten til sammen blir best mulig.

7. Kilder

Arora, A., Nandkumar, A., & R. Telang (2006) Does information security attack frequency increase with vulnerability disclosure? An empirical analysis. DOI 10.1007/s10796-006-9012-5

Ballintijn, B. (2005) *A case Study of the release Management of a Health-care Information System.* Amsterdam: Centrum voor Wiskunde en Informatica (CWI). Hentet fra https://pdfs.semanticscholar.org/eda6/dadce36418131e472151923c7521adf4f024.pdf

Brykczynski, B., & Small, R. A. (2003). Reducing internet-based intrusions: Effective security patch management. *IEEE software, 20*(1), 50-57.

CERT - Carnegie Mellon University. (2002, 17. januar). "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL. Hentet 23.04.17 fra http://www.cert.org/historical/advisories/CA-2001-19.cfm.

CERT - Carnegie Mellon University. (2001, 25. september). Nimda Worm. Hentet 23.04.17 fra https://www.cert.org/historical/advisories/CA-2001-26.cfm.

Forbes Technology Council. *Forbes*. Hentet 1. mai 2017 fra https://www.forbes.com/sites/forbestechcouncil/2017/02/17/seven-options-for-getting-users-to-update-software/2/#4406b9e814e5

Goldschmidt, C., Dark, M.J., & Chaudhry, H. (2010). Responsibility for the Harm and Risk of Software Security Flaws. DOI: 10.4018/978-1-61692-245-0.ch006

Harris, S., & Maymi, F., (2016) *CISSP All-in-One Exam Guide* (7. utg.). New York: McGraw-Hill Professional

Jøsang A. (2017a, 30. januar). Information Security, Human Factors for Information Security. Hentet fra http://www.uio.no/studier/emner/matnat/ifi/INF3510/v17/lectures/inf3510-2017-l02-isman-humfact.pdf

Jøsang A. (2017b, 6. februar). Risk Management, Business Continuity Management. Hentet fra http://www.uio.no/studier/emner/matnat/ifi/INF3510/v17/lectures/inf3510-2017-l03-risk-bcm.pdf

Kerner, Sean Michael (2015) *Lack of Patching Remains a Top Security Risk, HP Report Finds.* Hentet 1. ai 2017 fra http://www.eweek.com/security/lack-of-patching-remains-a-top-security-risk-hp-report-finds

Khandelwal, S. (2016, 27. november) San Francisco Metro System Hacket with Ransomware; Resulting in Free Rides. *The Hacker News*. Hentet fra http://thehackernews.com/2016/11/transit-system-hacked.html

NIST. (2013) Guide to Enterprise Patch Management Technologies. Hentet fra http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-40r3.pdf

Mell, P., Scarfone, k. & Romanosky, S. (2007) *A complete Guide to the Common Vulnerability Scoring System Version 2.0*. (CVSS). Hentet fra https://www.first.org/cvss/cvss-v2-guide.pdf

Næringslivets sikkerhetsråd. (2016) *Mørketallsundersøkelsen 2016*. Hentet fra 30. April 2017 fra http://nsr-org.no/getfile.php/Bilder/M%C3%B8rketallsunders%C3%B8kelsen/morketallsundersokelsen_2016.pdf

Securities and Exchange Commission. In the Matter of Knight Capital Americas LLC Respondent. Hentet fra https://www.sec.gov/litigation/admin/2013/34-70694.pdf

Selvaraj K. & Gutierrez N.F. (2010). *The Rise of PDF Malware*. Cupertino: Symantec. Hentet 23. april 2017 fra https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/security-response-rise-of-pdf-malware-10-en.pdf

Theel, Amanda (2015) *Lack of Patch Management Leads to Increase in Cybercrime.* Hentet fra https://coar.risc.anl.gov/lack-of-patch-management-leads-to-increase-in-cybercrime/

Veracode (2016). State of Software Security. Hentet fra https://www.veracode.com/sites/default/files/Resources/Reports/state-of-software-security-volume-7-veracode-report.pdf

Wakefield, J. (2016, 14. april). What does shake-up of EU data laws really mean? *BBC*. Hentet fra http://www.bbc.com/news/technology-36037324.

Williams, Wayne (2016). *Data breaches and cyber-attacks are often caused by failing to patch known (and fixable) vulnerabilities.* Hentet fra https://betanews.com/2016/01/12/data-breaches-and-cyber-attacks-are-often-caused-by-failing-to-patch-known-vulnerabilities/