Riskanalys rapport

Riskanalys rapport på Antivirusprogram

Erik Thomasson

Metod

Metoden jag använde för att identifiera riskerna i koden var att kolla specifikt på varje funktion i programmet och se vilka sårbarheter varje funktion har. Jag har också observerat att säkerheten av Virusdatabasen är en väldigt relevant faktor i att göra programmet säkert.

Prioritet	Risk/Sårbarhet	Allvarlighetsgrad (Mycket låg - Mycket Hög)	Allvarlighetsgrad efter åtgärd (Mycket låg - Mycket Hög)	Kostnad	Typ av sårbarhet
1	Ändring av Virusdatabasen	Mycket hög	Hög	Mycket	Ändring av data
2	Viruset inte ligger i början av filerna som ska genomsökas	Hög	Mycket Låg	Ca 4h arbetstid och lite exekveringstid	Mjukvarufel
3	Skyddar endast mot regular files	Medel	Mycket Låg	>24h	Mjukvarufel
4	Virus som inte finns i databasen	Mycket hög	Hög	Konstant arbete	Mjukvarufel
5	Identifierar endast om filen har ett virus, inte flera.	Låg	Mycket Låg	ca 4h	Mjukvarufel
6	För stora filer	Låg	Mycket Låg	1 dag	Mjukvarufel

Adresserade åtgärder

Legitimiteten av Virusdatabasen är den viktigaste aspekten gällande säkerheten för antivirusprogrammet. Den implementerade åtgärd för att gör Virusdatabasen säkrare är att programmet gör ett hashvärde utav innehållet i Virusdatabasen och sparar det värdet i en annan fil. Varje gång Antivirusprogrammet körs så görs ett hashvärde på innehållet utav Virusdatabasen och jämför det med det sparade hash värdet i separata filen. Programmet skriver sedan ut att virusdatabasen har ändrats om de två värdena inte överensstämmer. Det tog ca 2 dagar för att implementera åtgärden men jag prioriterade den eftersom det är den viktigaste sårbarheten att åtgärda.

En fullständig åtgärd för att sänka allvarlighetsgraden signifikant är att förvara filen med hash värdet på en säker plats och att använda sig av ett starkare krypto vid hashning av innehållet i virusdatabasen.

Det är en väldigt stor sannolikhet att filerna blir infekterade av virus om viruset endast behöver befinna sig efter början av filen. Jag löste det genom att jämföra hela filen med värdet från virusdatabasen. Biverkningen av åtgärden är att exekveringstiden blir längre. Innan åtgärden jämfördes första tecknet från filen med första värdet från virus databasens värde och om de inte överensstämde så slutade programmet att jämföra filen med värdet från virusdatabasen. Denna åtgärd kostade inte så mycket tid men det är ändå en viktigt åtgärd därför prioriterade jag denna åtgärd.

Om en fil är infekterad av virus så kan det vara bra att veta vilka virus som filen är infekterad av och inte bara vetskapen av att den är infekterad. Åtgärdens biverkning är att det blir lite extra exekveringstid. Det kostade mig väldigt lite tid med den implementationen av åtgärden med hur koden var konstruerad innan jag skulle implementera åtgärden. Därför prioriterade jag denna åtgärd.

Programmet checkade endast av vanliga filer med Virusdatabasen vilket betyder att andra sorters filer såsom Symbolic link, Socket mm. inte blir checkade. I den nya implementationen är detta löst, programmet checka med alla filer förutom “.” filen som är filen för vilken mapp man är i och “..” som är filen för mappen ovanför den nuvarande mappen.

Förslag på åtgärder för sårbarheter som inte har adresserats

Det upptäcks hela tiden nya virus så att säkerhetsställa att man har alla virusdefinitioner i virusdatabasen är näst intill omöjligt. Den bästa åtgärden för skydda mor så många virus som möjligt är att vara uppdaterad på om nya virus uppkommer. Ständigt uppdatera Virusdatabasen och vara kritisk mot säkerheten av programmet.

Om filerna är för stora kan det kosta mycket exekveringstid i värsta fall så mycket så att programmet inte går att köra. Ett förslag på åtgärd är då att kolla filens storlek innan man läser den och om den är större än en rimlig gräns så öppnar inte programmet den jämför filen med virusdatabasen.

Motivering av prioritet

Den känsligaste delen i programmet är Virusdatabasen. Om inte innehållet i Virusdatabasen stämmer så ger det en falsk implikation att alla filer är virusfria. Alltså är det viktigt att göra virusdatabasen så säker som möjligt mot oönskad ändring av data och därför har jag prioriterat den högst.

Programmet kollar endast om viruset är i början av filen och indikerar inte om viruset ligger någon annanstans i filen. Det är väldigt lätt att ta sig runt det speciellt om man vet att programmet bara kollar i början av filen. Att söka igenom hela filen har jag därför prioriterat högt.

Programmet kollar inte filer som inte är regular files såsom Symbolic link, Directory, Socket mm. Om attackeraren vet om att programmet inte kollar alla filer så är det lätt att använda den sårbarheten. Därför har jag prioriterat den sårbarheten högt upp.

Sårbarheten som alla antivirusprogram har är att skydda sig mot virus som ännu inte finns. Även fast det inte finns en absolut åtgärd som tar bort sårbarheten så är det bästa man kan göra att hålla sig uppdaterad och lägga in nya virusdefinitioner efterhand. Det är ändå viktigt att uppdatera virusdatabasen efterhand för att hålla uppe programmets effektivitet därför har jag prioriterat den åtgärden.

De två lägst prioriterade sårbarheterna är att programmet endast identifierar om filen har ett virus inte flera och om filen är för stor så kollar den , den filen ändå. att programmet inte skriver alla virus som filer har stärker inte säkerheten så mycket på filen utan mer att det hjälper till att återskapa filen. om filerna är för stora så kan det vara ett problem eftersom det kan leda till att man inte kan köra programmet men det är inte en säkerhetsrisk på samma sätt som de andra sårbarheterna som har högre prioritet.

Sammanfattning

Källan man jämför filer med är den mest vitala delen i ett Antivirusprogram. Man bör fokusera på att säkerhetsställa dess legitimitet framför andra säkerhetsaspekter av programmet. Rapporten beskriver vilka de mest signifikanta riskerna jag har hittat för programmet och har därefter gjort en lämplig åtgärd för att sänka allvarlighetsgraden av den sårbarheten.