IT-säkerhet för programmerare

Övningar och instuderingsuppgifter 17/8

Kryptobaserad it-säkerhet

En varning innan in sparkar igång Visual Studio: kryptoteknik är komplicerad, det är lätt att göra fel och det är oftast onödigt att uppfinna hjulet på nytt. Använd färdiga funktioner när du hittar sådana och försök tänka efter om du kan lita på den som producerat koden innan du börjar använda den.

Nedanstående uppgifter görs lämpligen i grupp om du inte har något specifikt själ att arbeta själv. Med rådande temperaturer så kan inlärningen förbättras om inte alla sitter i samma rum samtidigt, hela tiden, därför försöker vi med dessa uppgifter i stället för mer traditionella lektioner.
För att uppfylla kursmålen är avsnitt 1-8 nedan de som är viktigast, men allt eftersom kryptovalutornas blockkedjeteknik vinner fler och fler tillämpningar så kan även den delen vara användbar. Beträffande 10 och 11 är inte meningen att ni skall ställa till dumheter, förstöra för andra, göra olagligheter eller liknande; det är varken skolans eller lärarens uppfattning att sådana aktiviteter skall uppmuntras. Däremot kan det öka förståelsen för den som har tid över och som vill förstå bättre hur det fungerar.

Bra att veta:

När du krypterar inne i ett program är det oftast smidigt att använda de kryptonycklar i form av de datastrukturer som är definierade i dina krypteringsfunktioner. Men så fort du blandar in människor som vill ha verktyg för att behålla sina hemligheter så hamnar du som programmerare i den situationen att du måste omvandla lösenord till kryptonycklar. Den omvandlingen är svår att göra på ett bra sätt. Därför är det sällan du hittar några tydliga råd hur du skall göra. Ofta fungerar det tillräckligt bra genom att använda en hashfunktion för att omvandla ett lösenord till en krypteringsnyckel, men det är svårt att veta om detta verkligen fungerar bra i alla fall i verkligheten.

0. Varning för ”snake oil” undermediciner och egna uppfinningar

Kryptoteknik där säkerheten bygger på att ingen vet hur informationen blandats om vid krypteringen är inte it-säkerhet, det är vidskepelse. Du kan skydda din bostad från inbrott genom att gömma nyckeln under en sten eller genom att hälla klister i låset. Men det är nästan alltid bättre att se till att ha en bra dörr, i bra karmar och med ett bra lås. Ja och du måste naturligtvis se till att ingen får tag i din nyckel, både till dörren och din datakryptering.

Dina algoritmer skall på samma sätt vara välkända, beprövade och väl analyserad av matematiker och säkerhetsforskare runt om i världen.

1. Hashfunktioner

En hashfunktion arbetar ungefär som en krypteringsalgoritm, du utgår från en fil, ett meddelande, ett datablock eller liknande. Sedan flyttar hashfunktionen runt bittarna i dina indata på ett kontrollerat sätt, ungefär som när någon vill fuska i kortspel och blandar leken på ett sätt så att korten hamnar i en viss ordning. Men när en fil krypteras är ju hela den omblandade filen resultatet

av krypteringsfunktionen. Vid en hashning kastar man bort det mesta av utdatat som operationen genererat och man sparar bara en liten bit. När en kryptolog tar fram en hashfunktion är den viktigaste kvalitetsaspekten att en liten förändring av indata ofta (helst alltid) skall generera en ny hash. Vanliga algoritmer är MD5 och SHA-serien (det finns många, SHA står för Secure Hash Algoritm).

What Are Hash Functions, and How Do They Work?

https://medium.com/geekculture/what-are-hash-functions-and-how-do-they-work-3177553e429e

What Is a Hash Function in Cryptography? A Beginner's Guide

https://www.thesslstore.com/blog/what-is-a-hash-function-in-cryptography-a-beginners-guide/

Wikipedia kan vara en bra start här:

https://en.wikipedia.org/wiki/Hash_function

Eller varför inte med lite kod:

Calculate MD5 Hash From a String in C#

https://www.delftstack.com/howto/csharp/csharp-md5-hash/

Microsoft har färdiga funktioner för merparten av saker du vill kunna göra inom detta område:

https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography?view=net-6.0

https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.md5?view=net-6.0

https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.sha256?view=net-6.0

Exempel:

https://www.c-sharpcorner.com/article/compute-sha256-hash-in-c-sharp/

Uppgifter:

Kan två filer som inte är lika ha samma hashvärde?

Tänk efter olika situationer där det kan vara praktiskt att hasha data av olika slag, gör gärna en lista. Om du inte hade hashat din information i uppgiften ovan, finns det några av tillämpningarna där du kunde ha löst problemet på något annat sätt? Hur då?

Pröva att skriva program som hashar data, exempelvis en fil. Gör gärna egna sådana här program och jämför med dina kompisar, får de samma hashvärde på samma indata eller fil?

Jämför olika hash-algoritmer, vilken är snabbast? Vilken ger minst och vilken gör störst hashar?

2. Kryptering, symetrisk

Symetrisk kryptering har använts sedan flera tusen år tillbaka i olika former. Det fungerar bra när det finns en säker väg att byta kryptonycklar mellan de parter som vill kommunicera. Med dagens

elektroniska kommunikation, är det enda vi kan vara säkra på, är att vi inta bör lita på någon och att vi inte kan vara säkra på något. Av praktiska själ väljer vi att hoppas på det bästa…

Den vanligaste symetriska krypteringsalgoritmen idag heter AES

https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

https://sv.wikipedia.org/wiki/Advanced_Encryption_Standard

https://cybernews.com/resources/what-is-aes-encryption/

C#

https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.aes?view=net-6.0

https://www.c-sharpcorner.com/article/aes-encryption-in-c-sharp/

https://tomrucki.com/posts/aes-encryption-in-csharp/

https://www.codeproject.com/Articles/1278566/Simple-AES-Encryption-using-Csharp

3. Kryptering, assymetrisk

Den överlägset vanligaste assymetriska krypteringsalgoritmen heter RSA, det finns andra som Eliptic Curves, men de är inte så vanliga. Därför talar vi om RSA här även om de andra används på liknande sätt så är det mestadels den underliggande matematiken som skiljer. Eliptic Curves utgår från polynomfunktioner, RSA använder primtal. I praktiken behöver ni inte fundera så mycket på matten. Däremot finns det några praktiska saker. Assymetrisk kryptering kräver betydligt längre kryptonycklar än de flesta symetriska algoritmer med jämförbar säkerhet, på samma sätt går den symetriska krypteringen oftast betydligt snabbare (kräver färre programinstruktioner) för att uppnå samma säkerhet. Därför används RSA mest för signering och för att skicka kryptonycklar på ett säkert sätt. I praktiken är det inget användaren behöver tänka på. Användaren vet att hen behöver minst ett nyckelpar (som ofta skickas inpackade i certifikat). När sedan använder sitt cert. för att sätta upp en vpn-förbindelse, eller kanske för att kunna erbjuda HTTPS på sin webbtjänst, då kommer allt att fungera som vi sade på gårdagens lektion, där den ene användaren skickar sina saker till den andre användaren med hjälp av dennes publika nyckel (mottagaren använder sedan sin privata nyckel för att öppna meddelandet). Skillnaden är bara den att när det börjar skickas större datamängder så används RSA för att föra över en nyckel för AES (som ju är den överlägset vanligaste symetriska krypteringen idag; ja eller någon kanske väljer att använda en annan algoritm).

Det blir tyvärr många länkar här. Det här är komplicerat och inte så lätt att varken begripa eller förklara. För den här kursen är egentligen det viktigaste att ni vet hur ni använder den här tekniken, och naturligtvis även varför…

Egentligen är nog det viktigaste att ni förstår ur fantastiskt det är att kunna dela ut en kryptonyckel som gör att andra kan nå dig, utan att någon annan skall kunna ta reda på vad som skickades till dig.

RSA

https://en.wikipedia.org/wiki/RSA_(cryptosystem)

https://sv.wikipedia.org/wiki/RSA

https://cybernews.com/resources/what-is-aes-encryption/

https://don.p4ge.me/rsa-explained-simply/programming

Med färdig kod nedan, blir det lättare i praktiken:

C#

https://www.c-sharpcorner.com/UploadFile/75a48f/rsa-algorithm-with-C-Sharp2/

https://kashifsoofi.github.io/cryptography/rsa-encryption-in-csharp-using-microsoft-cryptography/

https://www.codeproject.com/questions/172951/source-code-for-rsa-algorithm-in-c

https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.rsa?view=net-6.0

Pröva gärna att skriv ett program som använder RSA-algoritmen, spara det och använd det till nästa två uppgifter.

4. Digitala signaturer

Med hjälp av en hash och din privata nyckel kan du skapa en digital signatur. Med sådana kan vi göra bankaffärer on-line, hämta pengar i bankomat och betala i affärer med kort. Plugga på här:

https://en.wikipedia.org/wiki/Digital_signature

https://www.techtarget.com/searchsecurity/definition/digital-signature

Moderna datorer (både sådana vi kallar datorer, men även mobiler, paddor och inbäddade system, som allt från skrivare till ”smarta” lampor) använder ofta signerade programfiler för att säkerställa att ingen mixtrat och ändrat på programfilerna.

https://docs.microsoft.com/en-us/windows/security/trusted-boot

How to verify digital signature in C#

https://stackoverflow.com/questions/56660195/how-to-verify-digital-signature-in-c-sharp

5. Digitala Certifikat

Ett digitalt certifikat är en signerad datafil, nästan alltid på ett format som heter x.509 (när Internationella Teleunionen tog fram de standarder som behövdes för att skapa PKI så fick alla heta saker som började på x.5__, ”x femhundra standarder” som vi brukar säga när vi talar om dem).

Ett sådant certifikat innehåller information om utställaren av certifikatet (certifikatsutställare heter Certificate Authority eller ”CA” på engelska). Giltighetstid, information om certifikatets ägare och så innehåller det nästan alltid kryptonycklar. X.509-standarden är så flexibel att ni i praktiken kan lagra nästan vad som helst i certifikatet, men det är inte så vanligt att certifikaten används för annat.

https://en.wikipedia.org/wiki/X.509

https://www.ssl.com/faqs/what-is-an-x-509-certificate/

Gratiscertifikat kan ni göra hos:

https://letsencrypt.org/

När behövs certifikat?

Hur skapar du certifikat?

Vad händer när ditt certifikat har ”gått ut”? Blir krypteringen sämre?

Hur hittar jag vilka certifikatsutställare som ligger i min webbläsare som standard?

6. PKI – Public Key Infrastructure

PKI är den organisation som Internationella Teleunionen tänkte ut på 1970-talet för att skapa en säker nätmiljö i framtidens digitala värld. Skillnaden är att vi idag har många utställare i stället för en i varje land, som man då planerade.

Med PKI for vi en organisationsstruktur för hur vi distribuerar och verifierar certifikat så att vi kan bygga säkra system som använder certifikat.

Ni kommer att hitta felaktigt information på nätet som säger att certifikaten gör det möjligt att verifiera identiteten hos en person eller ett företag och ibland finns det de som hävdar att man till och med kan använda certifikaten för att avgöra om ett företag tänker lura sina kunder eller inte. Detta är naturligtvis inte sant. Precis som många yngre har haft falska id-kort för att komma in på krogen så kan jag låna ut mitt bankid till vem som helst som i sin tur kan göra spännande saker utan att någon kan veta om det var jag eller någon annan som gjorde dessa dumheter. I avtalet med min bank och med Svensk ID Teknik så står naturligtvis att jag inte får dela mitt id med någon, men vem stoppar mig???

https://en.wikipedia.org/wiki/Public_key_infrastructure

7. HTTPS, SSL/TLS, VPN

Med SSL och dess moderniserade variant TLS så kan vi skapa en krypterad förbindelse, en krypterad TCP-socket, mellan två maskiner på nätet. För att detta skall vara möjligt används nästan alltid certifikat, ibland används certifikat både i klient och server, för att kunna verifiera att klienten är en tillåten sådan. Från början var tanken att detta skulle användas för säkra bankaffärer, men de

flesta banker har valt andra lösningar än så. Men tekniken används ibland för att sätta upp VPN så att båda ändarna på förbindelsen kan verifiera att den är ihopkopplad med någon som åtminstone har rätt nycklar i andra änden.

https://en.wikipedia.org/wiki/Transport_Layer_Security

https://sv.wikipedia.org/wiki/Transport_Layer_Security

https://nordvpn.com/sv/blog/ssl-tls/

8. Nyckelutbyte

När någon vill koppla upp en säker förbindelse (SSL/TLS, VPN, wifi, Blutetooth och i de flesta andra sammanhang) så är den mest säkerhetskritiska operationen att komma överens om vilken kryptonyckel som skall användas. Därför är en vanlig angreppsmetod, speciellt på trådlösa nät, att försöka koppla ned de förbindelser som finns så att en ny uppkoppling kan avlyssnas och analyseras. Innan vi fick någon känd metod för assymetrisk kryptering så användes en metod som heter Diffie-Hellman, eftersom RSA tidigare var patentskyddad runt om i världen (fram till 1998) så använde många Diffie-Hellman eller varianter på denna för att slippa betala för RSA. Den används i många sammanhang fortfarande även om RSA blir allt vanligare.

https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

https://www.comparitech.com/blog/information-security/diffie-hellman-key-exchange/

https://www.codeproject.com/Articles/1219531/Implement-Diffie-Hellman-in-Csharp

9. Kryptovalutor och blockkedjor

Wikipedia

https://en.wikipedia.org/wiki/Cryptocurrency

Kryptovalutor och blockkedjor med C# exempel:

https://towardsdatascience.com/blockchain-explained-using-c-implementation-fb60f29b9f07

10. Fördjupning, Wifi-nät, sårbarheter

https://en.wikipedia.org/wiki/Aircrack-ng

https://sv.wikipedia.org/wiki/Aircrack-ng

https://www.aircrack-ng.org/

https://www.aircrack-ng.org/doku.php?id=getting_started

11. Fördjupning, intrång, hur går det till?

https://www.metasploit.com/get-started

13. PGP/GPG för säker epost (och mycket annat)

Pretty Good Privacy, PGP (eller den moderna uppföljaren GPG, ibland även kallad GnuPG) är ett verktyg som som gör vad det heter. I stället för att arbeta med en central certifikatsorganisation som inom PKI, så använder man en digital motsvarighet till det som vi alla gör med människor runt omkring oss. Jag litar på de jag känner, jag litar på mina kompisars kompisar, jag litar på sådana personer som jag vet många andra litar på. Web of Thrust kallas konceptet för här. För att komma igång med GPG (som idag är den vanligaste versionen) så skapar du ett par nyckelfiler (motsvarar certifikaten inom PKI), den ena med din publika och den andra med din privata nyckel. För att öka din publika nyckels ”värde” så låter du dina kompisar signera den, du kan lägga upp den på en nyckelserver där dina kompisar och deras kompisar (ja och vem som helst) kan signera din nyckel (och därmed tala om att de litar på att nyckeln är just din nyckel). Ibland ordnas det key signing parties där folk träffas för att signera varandras nycklar och för att ha det trevligt tillsammans. Göra sådana saker folk brukar göra på en fest utan kryptonycklar.

När du sedan har dina nycklar ordnade, så kan du signera och kryptera filer och meddelanden. Ibland räcker det med en signatur, som när du uppdaterar ett program på Github. Den som är mer intresserad av att störta en ond diktator, kanske vill både kryptera och signera. Mottagaren vill ju veta inte bara vad du säger utan att det är du som säger det hela så att ingen hamnar i en fälla som diktatorn gjort.

https://en.wikipedia.org/wiki/Pretty_Good_Privacy

https://gnupg.org/

https://www.openpgp.org/

Se om du kan installera GPG i din epostklient, exempelvis kan K-9 Mail på android klara den uppgiften, eller Thunderbird på en vanlig dator.

https://k9mail.app/

https://www.thunderbird.net/sv-SE/

Skall du bara signera och kryptera din mejl, finns ju alternativet PKCS11 (även kallad S/MIME) som är en PKI-baserad lösning. Du måste skaffa ett certifikat. Har du ett sådant skall du kunna installera det i Outlook utan ytterligare program.

Varför är kryptering av epost viktigt för den som värnar om sin personliga integritet?

Varför har krypterad epost inte blivit populär trots att så många säger att personlig integritet är viktigt?

Vad händer på ett företag när någon som krypterat allt sin epost slutar? Eller när personen blivit sjuk eller drabbats av en olycka?