Univerza v Ljubljani
Fakulteta za Računalništvo in informatiko

Podiplomski magistrski program Informacijski sistemi in odločanje

Smo res varni?

Projekt pri predmetu
Kriptografija in računalniška varnost

Predavatelj: **prof. dr. Aleksander Jurišič**

Avtor:
Matej Trampuš
Fakulteta za računalništvo in informatiko, Ljubljana
e-naslov: matej.trampus@fri.uni-lj.si

Povzetek

Članek raziskuje računalniško varnost pri končnem uporabniku. Opisan je napad pri katerem spremenimo delovanje obstoječe aplikacije. Podrobneje je predstavljeno, kako je mogoče prestrezati klice podprogramov, ki se nahajajo v dinamično povezljivih knjižnicah. Kot ilustracija uporabe te tehnike sta izvedena napada na spletni brkljalnik, ki uporablja SSL in napad na program za elektronsko pošto, ki omogoča digitalno podpisovanje sporočil.

Ljubljana, junij 2000

Vsebina

1 Uvod

Uporabniki (pa tudi ponudniki) internetnih trgovin, bank in drugih rešitev za elektronsko poslovanje so ponavadi kar hitro prepričani v varnost rešitve, le da se v njenem opisu dovolj pogosto pojavljajo besede kot kriptografija, SSL, digitalni podpis, digitalni certifikati ipd. Res je, da nam te sodobne tehnologije zagotavljajo visoko stopnjo varnosti, a vendar ne smemo pozabiti, da je veriga močna toliko, kot je močan njen najšibkejši člen.

Veliki sistemi, strežniki in komunikacijske poti so danes praviloma dobro zaščiteni, saj nad njimi bdi cela vojska strokovnjakov, ki so plačani za to, da ne bi šlo kaj narobe. Tako se najšibkejši člen verige pogosto nahaja pri končnih uporabnikih, ki niso dovolj izobraženi, da bi znali ustrezno varovati svoje računalnike, in zato brez pomislekov odpirajo na kupe raznih novoletnih čestitk in ostale navlake, ki jo vsakodnevno dobivajo po elektronski pošti. Na žalost večina današnjih operacijskih sistemov ni zasnovana tako, da bi lahko uporabnika učinkovito ščitili pred tovrstnimi nevarnostmi.

V tem članku se bomo osredotočili predvsem na napade na računalnik končnega uporabnika. Najprej si bomo ogledali napad s spremembo delovanja obstoječe aplikacije. V tretjem razdelku bomo govorili o tem, kako priti do naslovnega prostora neke aplikacije. V četrtem razdelku si bomo podrobneje ogledali kako prestrezati klice v dinamično povezljive knjižnice. Razredi, ki omogočajo prestrezanje takšnih klicev so podrobneje opisani v petem razdelku. S pomočjo teh razredov je enostavno izvesti napad na program za elektronsko pošto in spletni brkljalnik, ter tako ponarediti elektronski podpis ali podatke, ki jih preko varne SSL povezave pošiljamo spletnemu strežniku. Takšna napada sta, skupaj z (zmotnim) prepričanjem uporabnikov, opisana v šestem poglavju. Sedmo poglavje opisuje, kako se zaščititi pred tovrstnimi napadi.

Članek opisuje varnost in napade v družini operacijskih sistemov Windows (Windows 95, Windows 98, Window NT 4.0, Windows 2000). Napade z uporabo dinamično povezljivih knjižnic, ki so glavna tema tega članka, pa je možno brez večjih težav izvesti tudi na drugih operacijskih sistemih (Linux, druge različice Unixa,...).

2 Napad s spremembo delovanja obstoječe aplikacije

V tem članku se bomo osredotočili na izvedbo napada, ki za dosego svojega cilja spremeni delovanje aplikacij, ki jih uporablja končni uporabnik. Pri takšnem napadu končni uporabnik ponavadi sploh ne opazi, da se dogaja kaj čudnega, dokler seveda ni prepozno. Za izvedbo takšnega napada je v večini primerov potrebno najprej namestiti napadalsko kodo na uporabnikov računalnik in nato s pomočjo te kode spremeniti delovanje ene ali več aplikacij končnega uporabnika

Namestitev kode na uporabnikov računalnik je najenostavneje izvesti s pomočjo trojanskega konja. Povprečni uporabniki niso dovolj izobraženi, da bi se zavedali nevarnosti, ki prežijo na njih in brez pomisleka odpirajo vso navlako, ki zaide v njihov elektronski nabiralnik.

Svojo kodo lahko v osnovno aplikacijo vključimo na več načinov. Najpreprostejše je, da na disku zamenjamo ali spremenimo eno od datotek, ki jih aplikacija potrebuje za svoje delovanje. Ker nekateri operacijski sistemi in aplikacije preverjajo integriteto svojih datotek (dolžina datoteke, kontrolna vsota, digitalni podpis....) je takšno zamenjavo enostavno odkriti. Težje pa je odkriti spremembo aplikacije, ko je ta že aktivna in naložena v spomin. Operacijski sistem namreč ne more razlikovati med tem, za kar je bila aplikacija namenjena in

tem, kaj aplikacija dejansko dela. Tudi same aplikacije ponavadi niso napisane tako, da bi preverjale pravilnost svojega delovanja (pa tudi če so, je možno tako kot samo aplikacijo spremeniti tudi preverjanje).

Napad je potrebno sprožiti v pravem trenutku, zato nas zanima, kaj se z aplikacijo, ki jo spreminjamo, dogaja. Če hočemo npr. ukrasti sporočilo, ki ga uporabnik pošilja po elektronski pošti moramo vedeti, kdaj uporabnik pošlje sporočilo in kakšna je vsebina tega sporočila. Sodobna programska oprema je napisana v obliki komponent, ki med sabo komunicirajo preko dobro definiranih in dokumentiranih vmesnikov. Dobro definirani vmesniki omogočajo ločitev in manjšo medsebojno odvisnost delov aplikacije ter lažji razvoj in vzdrževanje programske opreme. Poleg tega, da dobro definirani vmesniki močno olajšajo delo razvijalcem programske opreme, pa močno olajšajo tudi napad na to programsko opremo. Napadalec lahko namreč v vmesnik, preko katerega komunicirajo različni deli aplikacije, vrine svojo kodo in s tem spremlja in spreminja delovanje aplikacije.

Vmesnikov, ki so uporabljeni za komunikacijo med deli aplikacij in so primerni za spremljanje in spreminjanje delovanja aplikacij, je na operacijskem sistemu Windows več vrst. Najuporabnejša mesta za prestrezanje so [1]:

- prestrezanje klicev podprogramov v dinamično povezljivih knjižnicah,
- prestrezanje klicev metod COM (Component Object Model) objektov,
- prestrezanje windows sporočil.

V dinamično povezljivih knjižnicah se po navadi nahajajo najbolj nizko nivojski podprogrami. COM objekti predstavljajo višjo stopnjo abstrakcije. Za svoje delovanje COM objekti pogosto uporabljajo kopico nižje nivojskih podprogramov, ki so pogosto implementirani v obliki dinamično povezljivih knjižnic. Windows sporočila se uporabljajo za komunikacijo med programom in grafičnim uporabniškim vmesnikom. V tem članku se bomo osredotočili na prestrezanje klicev podprogramov v dinamično povezljivih knjižnicah.

3 Kako do naslovnega prostora aplikacije

Sodobni operacijski sistemi vsakemu procesu dodelijo svoj naslovni prostor. S tem preprečijo, da bi neka aplikacija namerno ali nenamerno (hrošč v programu) brala in spreminjala podatke drugih aplikacij, sledila delovanju drugih aplikacij ali jih kako spreminjala. Argument o namernem branju kar hitro odpade, saj je v operacijskem sistemu Windows precej enostavno priti do naslovnega prostora neke aplikacije. Najpogostejši načini so:

- uporaba sistemske funkcije *CreateRemoteThread* ,
- uporaba sistemskih funkcij *ReadProcessMemory, WriteProcessMemory,*
- uporaba sistemske funkcije *SetWindowsHookEx,*
- zamenjava dinamične knjižnice, sprememba sistemskega registra COM razredov,
- uporaba sistemskih gonilnikov.

Sistemski funkciji *ReadProcessMemory* in *WriteProcessMemory* omogočata branje in pisanje podatkov v naslovnem prostoru drugega procesa.

Sistemska funkcija *CreateRemoteThread* omogoča, da v drugem procesu naredimo novo nit in začnemo izvajati kodo na nekem naslovu v naslovnem prostoru drugega procesa. Predhodno moramo poskrbeti, da se na ciljnem naslovu nahaja želena koda.

Za uspešen klic funkcij *ReadProcessMemory WriteProcessMemory* in *CreateRemoteThread* moramo imeti pravice za dostop do ciljnega procesa. Te pravice lahko dobimo precej enostavno - dobimo jih npr., če sami zaženemo ciljni proces.

Sistemska funkcija *SetWindowsHookEx* omogoča, da opazujemo windows sporočila, ki jih dobivajo različni procesi (vsaka aplikacija, ki ima uporabniški vmesnik sprejema tudi windows sporočila). S klicem te funkcije lahko dosežemo, da operacijski sistem v ciljni proces naloži našo dinamično knjižnico in tako dobimo dostop do procesovega naslovnega prostora.

V naslovni prostor ciljnega procesa lahko pridemo tudi tako, da na disku zamenjamo eno od dinamičnih knjižnic, ki jih uporablja proces s svojo dinamično knjižnico. Če ciljni proces uporablja COM, lahko v sistemskem registru popravimo informacije za enega od COM razredov, ki jih proces uporablja in tako dosežemo, da se v proces, namesto izvirne, naloži naša dinamična knjižnica.

Sistemski gonilniki ponavadi tečejo kot del operacijskega sistema in imajo zato dostop do naslovnega prostora kateregakoli procesa. Če uporabnika uspemo prepričati, da namesti naš sistemski gonilnik, lahko tako obidemo praktično vse zaščite operacijskega sistema.

Nekatere aplikacije nam ponujajo še druge načine, kako priti v njihov naslovni prostor. Internetni brkljalnik Netscape Navigator npr. omogoča namestitev dodatkov (*plug-in*), ki razširijo njegovo funkcionalnost (podpora mulimediji,...). Ti dodatki so implementirani v obliki dinamičnih knjižnic, ki jih Netscape Navigator naloži v svoj naslovni prostor.

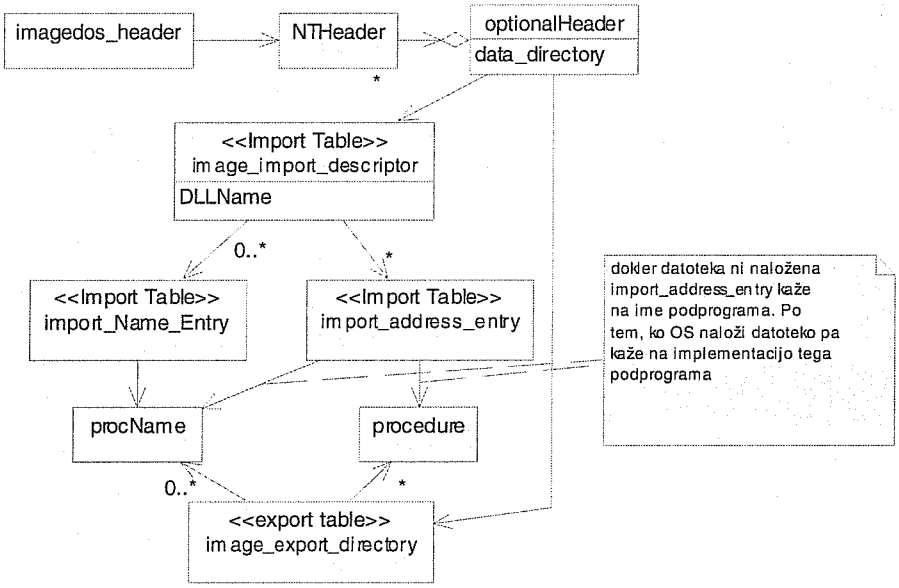
4 Prestrezanje klicev v dinamično povezljive knjižnice

Dinamično povezljive knjižnice so odgovor na vprašanje ponovne uporabljivosti kode na binarnem nivoju. V nasprotju s statičnimi knjižnicami, podprogrami v dinamičnih knjižnicah niso vsebovani v izvršilni datoteki končne aplikacije, ampak se nahajajo v ločeni datoteki. Prednost takšne rešitve je očitna: na disku imamo zmeraj le eno kopijo implementacije nekega podprograma. Če hočemo spremeniti implementacijo podprograma (npr. zaradi odprave hrošča, izboljšanja algoritma....) ni potrebno ponovno prevajati vseh aplikacij, ki uporabljajo ta podprogram, ampak je dovolj, da zamenjamo le to datoteko[1].

V času izvajanja aplikacije operacijski sistem naloži dinamično knjižnico v naslovni prostor aplikacije in poskrbi za povezavo podprogramov iz dinamične knjižnice z ostalimi deli aplikacije.

V operacijskem sistemu Windows so glavne izvršilne datoteke in dinamične knjižnice zapisane v PE (*Portable Executable)* formatu [2],[3]. Datoteka je razdeljena v več razdelkov, v katerih se nahajajo informacije, ki jih potrebuje operacijski sistem za izvajanje izvršilnih datotek. Med drugim se v datoteki nahajata tudi razdelka s seznamom podprogramov, ki jih ta datoteka ponuja drugim datotekam (tabela izvozov) in seznamom podprogramov, ki jih ta datoteka potrebuje od drugih datotek (tabela uvozov). Ti dve tabeli omogočata dinamično povezovanje različnih datotek.

[1] Zaradi nediscipline razvijalcev programske opreme se je v preteklosti velikokrat dogajalo, da nove različice dinamičnih knjižnic niso bile povsem združljive s starimi. Ob namestitvi nove različice so tako nekateri stari programi nehali delovati - nastal je t.i. DLL pekel (*DLL Hell).* Danes obstajajo tudi boljše tehnike za reševanje tega problema. Ena od njih je uporaba COMa (Component Object Model).



Slika 1 PE format datoteke - razredni diagram

Kako je izvedeno dinamično povezovanje? Recimo, da neka izvršilna datoteka za svoje delovanje potrebuje druge dinamične knjižnice. Seznam potrebnih dinamičnih knjižnic je zapisan v tabeli uvozov te izvršilne datoteke. Ko operacijski sistem nalaga izvršilno datoteko, najprej pogleda v tabelo uvozov in naloži vse potrebne dinamične knjižnice. Potem s pomočjo tabel izvozov dinamičnih knjižnic popravi tabelo uvozov osnovne izvršilne datoteke tako, da vanjo vpiše kazalce na implementacijo podprogramov v dinamičnih knjižnicah.

Klic podprograma v dinamični knjižnici se izvrši preko tabele uvozov: program najprej pogleda v tabelo uvozov na kateri lokaciji se nahaja želeni podprogram, nato pa ta podprogram izvede. Če popravimo tabelo uvozov, potem bo program namesto želenega podprograma poklical naš podprogram....

Poleg povezovanja s pomočjo tabele uvozov, lahko aplikacija naslov podprograma v dinamični knjižnici dobi tudi s pomočjo sistemskih funkcij *LoadLibrary* in *GetProcAddress*. Prednost takšnega povezovanja je ta, da aplikacija lahko naloži dinamične knjižnice šele v trenutku, kot jih potrebuje[2].

5 Razredi za prestrezanje klicev

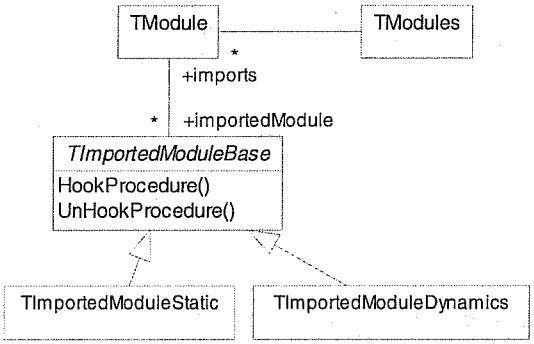
Ta razdelek opisuje razrede, ki so bili izdelani za prestrezanje klicev v dinamične knjižnice. Ti razredi so uporabljeni za napada opisana v razdelku 6.

Razredi omogočajo enostavno prestrezanje klicev. Naslednja tabela prikazuje seznam vseh razredov skupaj s kratkim opisom.

Ime razreda	**Opis**
TModules	Vsebuje seznam vseh dinamičnih knjižnic, ki so naložene v trenutni proces.
TModule	Vsebuje seznam vseh dinamičnih knjižnic, ki jih

[2] Novejše verzije prevajalnikov in povezovalnikov podpirajo tudi t.i. zakasnjen uvoz, ki je kombinacija uvoza s pomočjo tabele uvozov in klica funkcij *LoadLibrary/GetProcAddress*.

	uporablja neka dinamična knjižnica.
TImportedModuleBase	Predstavlja dinamično knjižnico, ki jo uporablja druga dinamična knjižnica. Vsebuje metode, ki omogočajo prestrezanje klicev med drugo in prvo dinamično knjižnico.
TImportedModuleStatic	Specializacija razreda *TImportedModuleBase*. Omogoča prestrezanje klicev podprogramov, ki so našteti v tabeli uvoza.
TimportedModuleDynamic	Specializacija razreda *TImportedModuleBase*. Omogoča prestrezanje klicev podprogramov, katerih naslov je dobljen z sistemko funkcijo *GetProcAddress*.



Slika 2 Razredi, ki omogočajo enostavno prestrezanje klicev med dinamičnimi knjižnicami - razredni diagram

Razredi so implementirani v programskem jeziku Delphi. Uporaba je enostavna. Recimo, da želimo zvedeti, kdaj bo dinamična knjižnica *from.dll* poklicala podprogram *someSub* v dinamični knjižnici *to.dll*. Vse, kar moramo narediti je, naslednje:

```
hookObj.ModuleByName['from.dll'].ImportedModuleByName['to.dll']
      .HookProcedure('someSub',OnCallingSomeSub);
```

in naša metoda *OnCallingSomeSub* bo poklicana tik preden bo klican *someSub*. V metodi si lahko ogledamo parametre poslane podprogramu *someSub*, jih spremenimo, ali pa celo preprečimo klic podprograma.

6 Laži in prevare

V tem razdelku sta prikazana dva primera, ki ilustrirata, kaj je možno narediti s prestrezanjem klicev v dinamične knjižnice. Primera uporabljata razrede predstavljene v razdelku 5. Vsakega od primerov se da enostavno podtakniti na računalnik končnega uporabnika. Kljub temu, da sta primera napisana v objektnem programskem jeziku visokega nivoja (Delphi), je izvršilna koda vsakega od primerov krajša od 100 KB. Če bi za izvedbo uporabili kakšen nižji programski jezik in pazili na dolžino programa, bi bila koda verjetno krajša od 30 KB. Tako majhna izvršilna datoteka je dovolj kratka, da jo lahko skrijemo v trojanskega konja in razpečavamo po elektronski pošti.

6.1 Če uporabljam SSL, sem varen

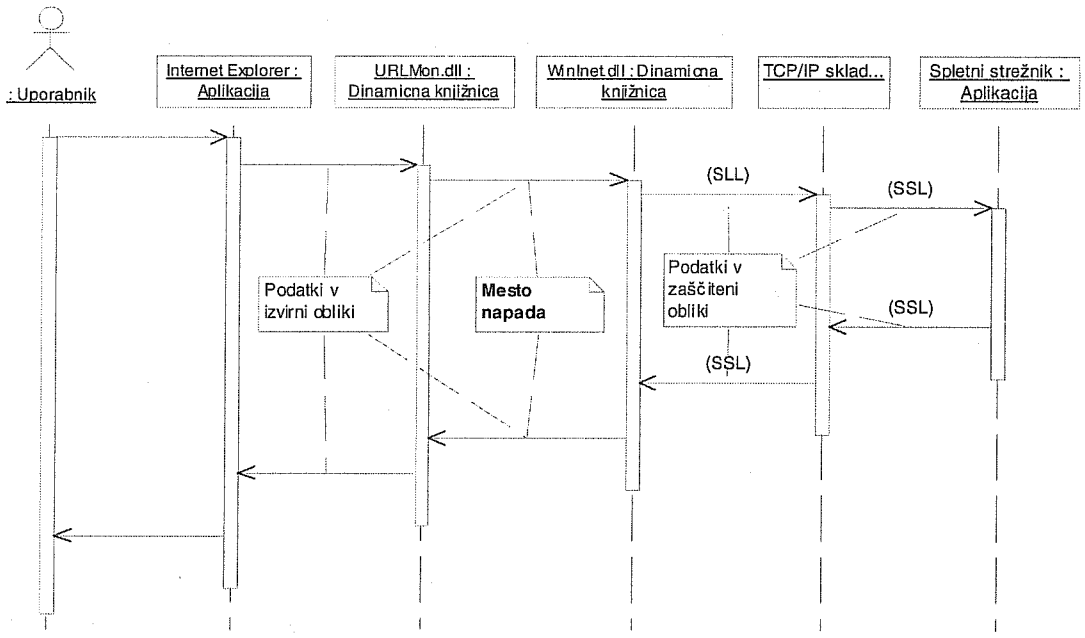
Prepričanje uporabnika: če uporabljam varno SSL povezavo s spletnim strežnikom, potem ne more nihče spremeniti ali ponarediti podatkov, ki sem jih poslal strežniku, ne da bi le-ta to opazil. Prav tako nihče ne more izvedeti, kakšni so ti podatki.

Resnica: na poti med računalnikom uporabnika in spletnim strežnikom ne more nihče spremeniti ali ponarediti podatkov. Lahko pa jih spremeni in ponaredi preden zapustijo računalnik. Uporabnik lahko le slepo zaupa brkljalniku, da bo v varni SSL kanal poslal tiste podatke, ki jih je uporabnik vpisal. Prav tako mora zaupati še vsem ostalim programom, ki so z ali brez njegove vednosti nameščeni na njegovem računalniku.

Oglejmo si primer napada za brkljalnik Internet Explorer.

Kako se prikrasti v naslovni prostor? Kot vse sodobne programe je možno Internet Explorer nadgraditi z množico dodatkov, ki naj bi končnemu uporabniku olajšali deskanje po Internetu. Internet Explorer podpira tako imenovane brkljalniške objekte (*Browser objects*) [4]. To so COM objekti, ki jih brkljalnik naloži v svoj naslovni prostor in obvešča o zanimivih dogodkih, kot so npr. prehajanje uporabnika med različnimi stranmi ipd. Če napišemo svoj brkljalniški objekt, torej brez težav pridemo do želenega naslovnega prostora.

Kaj narediti, ko smo v naslovnem prostoru? Internet Explorer za delo posredno uporablja WinInet - zbirko visoko nivojskih funkcij za delo z HTTP, HTTPS, FTP in drugimi internet protokoli. Za pošiljanje podatkov se v WinInetu uporablja podprogram *HttpSendRequest*. Če prestrežemo klic tega podprograma, imamo dostop do nezaščitenih podatkov, ki jih brkljalnik pošilja v internet. Podatke lahko opazujemo in jih poljubno spreminjamo, še preden so ti zaščiteni z SSL.



Slika 3 Napad na spletni brkljalnik

6.2 Če uporabljam digitalni podpis, sem varen

Prepričanje uporabnika: Če v programu za elektronsko pošto pod sporočilo dodam svoj digitalni podpis [5], potem je naslovnik lahko prepričan, da bere sporočilo, takšno kot sem ga napisal jaz.

Resnica: Če v programu za elektronsko pošto uporabnik pod sporočilo doda svoj digitalni podpis, potem je naslovnik lahko prepričan le, da bere sporočilo takšno, kakršno je bilo v trenutku, ko je bilo podpisano. Med tem, ko je uporabnik v svojem programu za elektronsko pošto izbral *podpiši sporočilo* in trenutkom, ko je bilo sporočilo podpisano, lahko kdo sporočilo tudi spremeni. Uporabnik lahko le slepo zaupa svojemu programu za elektronsko pošto, da podpiše res izvirno sporočilo. Prav tako mora zaupati še vsem ostalim programom, ki so z ali brez njegove vednosti nameščeni na njegovem računalniku.

Oglejmo si primer napada za program Outlook Express, ki za kriptografske funkcije uporablja CryptoAPI [7].

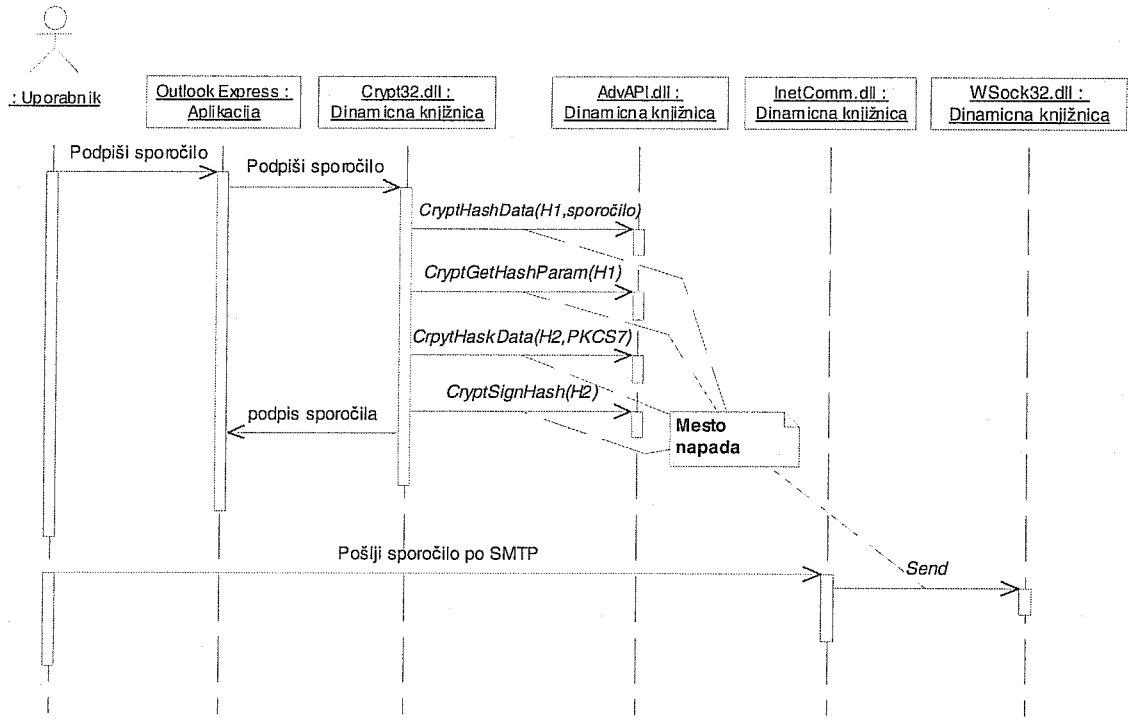
Kako priti v naslovni prostor programa? Preprosto: izberemo si enega iz množice COM razredov, ki jih uporablja Outlook Express. Sistemski register spremenimo tako, da namesto dinamične knjižnice, ki vsebuje implementacijo tega razreda, navedemo ime svoje dinamične knjižnice. Svojo dinamično knjižnico napišemo tako, da implementira vse podprograme, ki jih implementira tudi originalna knjižnica in to na takšen način, da klice podprogramov preprosto posreduje originalni knjižnici. Ker imajo dinamične knjižnice, ki vsebujejo COM objekte praviloma največ štiri podprograme, in so imena ter namen teh podprogramov znani, je to zelo enostavna naloga.

Kaj narediti, ko smo v naslovnem prostoru? Najprej počakamo, da uporabnik poizkuša podpisati kakšno sporočilo. V trenutku, ko uporabnik podpisuje sporočilo, ukrademo ročico na kriptografski kontekst, ki ga uporablja za podpisovanje. Tako dobimo dostop do uporabnikovega privatnega ključa, ki ga hrani eden od ponudnikov kriptografskih storitev (*Cryptographic Service Provider*). Izračunamo izvleček (*hash*) sporočila, ki ga hočemo podtakniti in uporabimo dostop do privatnega ključa za to, da podpišemo svoj izvleček. Izvirno uporabnikovo sporočilo pustimo zaenkrat nedotaknjeno in nadaljujemo normalno izvajanje programa. Tako dobi uporabnik v izhodni mapi podpisano izvirno sporočilo.

Ko Outlook Express poizkuša poslati sporočilo poštnemu strežniku, prestrežemo komunikacijo s poštnim strežnikom in namesto izvirnega sporočila podtaknemo svoje sporočilo. Izvirno PKCS7 pripono [6], ki vsebuje podpis izvirnega sporočila zamenjamo s svojo. To lahko naredimo preprosto tako, da v njej poiščemo in zamenjamo originalni izvleček in podpis z izvlečkom in podpisom našega sporočila.

Naslovniki bodo tako dobili spremenjeno sporočilo, uporabnik pa tega ne bo opazil, saj bo imel v mapi poslanih sporočil podpisano izvirno sporočilo[3].

[3] Uporabnik lahko opazi spremembo sporočila, če pošlje kopijo sporočila še sebi. Tudi ta primer je enostavno odkriti in poskrbeti, da uporabnik namesto spremenjenega "sprejme" izvirno sporočilo.



Slika 4 - Napad na program za elektronsko pošto

Analiza delovanja programa Outlook Express je prinesla še dve zanimivi ugotovitvi: pogovorno okno, ki opozarja uporabnika, da bo aplikacija uporabila njegov privatni ključ, je možno enostavno skriti. To pomeni, da lahko napadalec v primeru, da ima uporabnik privatni ključ shranjen na disku, dostopa do privatnega ključa, ne da bi bil uporabnik na to opozorjen.

Druga ugotovitev se nanaša na dinamično knjižnico *WS2_32.DL*, v kateri se nahaja implementacija TCP/IP sklada. Ta knjižnica namreč s klicem sistemskega podprograma *GetProcAddress* preverja, kje v spominu se nahajajo nekateri njeni podprogrami. Če se dobljene vrednosti razlikujejo od tistih, ki jih pričakuje, knjižnica javi napako. Težko bi rekli, da je to primerjanje namenjeno preprečevanju spreminjanja delovanja programa - verjetno služi kakšnemu drugemu namenu. Kakorkoli že, tudi to preverjanje je dokaj enostavno obiti.

7 Kako se zaščititi

Kako se lahko zaščitimo pred napadom opisanim v tem članku? Operacijski sistem nam pri tem ne more učinkovito pomagati. Lahko sicer zaščitimo kritična mesta datotečnega sistema in nekatere konfiguracijske podatke, ki se nahajajo v sistemskem registru, vendar je danes na žalost večina programov napisanih tako, da od sistema zahtevajo večje pravice, kot pa bi jih dejansko potrebovali. Takšni programi v sistemu s poostrenim nadzorom in zaščito ne delujejo.

Tako je edina učinkovita zaščita, ki nam preostane ta, da ne zaganjamo programov, če nismo povsem prepričani o njihovem izvoru in delovanju. Eden od mehanizmov za potrditev avtentičnosti programov je digitalni podpis, vendar danes le redki proizvajalci programske opreme opremljajo svoje datoteke z digitalnimi podpisi.

8 Zaključek

Napad s prestrezanjem klicev podprogramov, ki se nahajajo v dinamično povezljivih knjižnicah je enostavno izvedljiv in zelo učinkovit. Uporabnik le težko pravočasno opazi, da je nekaj narobe. Danes ne obstaja učinkovita zaščita, s katero bi se povprečno izobražen uporabnik lahko obvaroval takšnega napada - operacijski sistemi in uporabniške aplikacije še niso dosegli stopnje razvoja, ki bi takšno zaščito omogočala. Zanimivo bi bilo raziskati, kaj bi morali dodati obstoječim operacijskim sistemom, da bi ti na eni strani nudili učinkovito zaščito pred tovrstnimi napadi, na drugi pa še vedno omogočali udobno delo končnemu uporabniku.

V prihodnosti bi bilo zanimivo podrobneje raziskati še druge vrste napadov s spreminjanjem delovanja obstoječe aplikacije. Vse več programskih komponent med sabo komunicira z uporabo tehnologij povezovanja objektov kot sta COM in CORBA. S prestrezanjem klicev metod bi lahko vplivali tudi na delovanje objektov, ki se nahajajo na oddaljenih računalnikih.

9 Literatura in viri

[1] In memory patching: three approaches,
http://www.cyberbux.de/fravia_new/stone1.htm

[2] Peering Inside the PE: A Tour of the Win32 Portable Executable File Format,
http://msdn.microsoft.com/library/techart/msdn_peeringpe.htm

[3] The PE file format,
http://fravia.kilrathi.pl/pe.txt

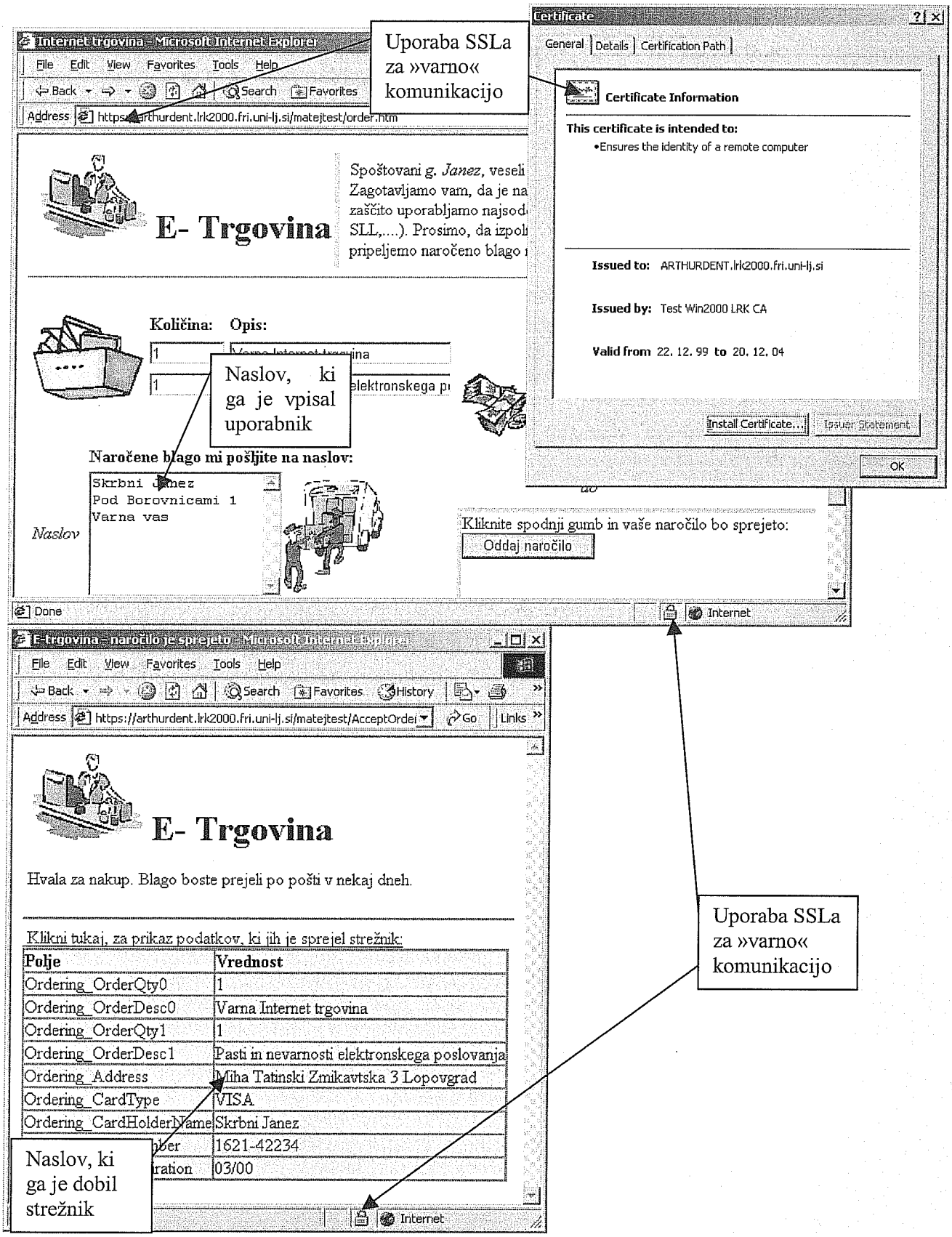
[4] Browser Helper Objects The Browser the Way You Want It,
http://msdn.microsoft.com/library/techart/Bho.htm

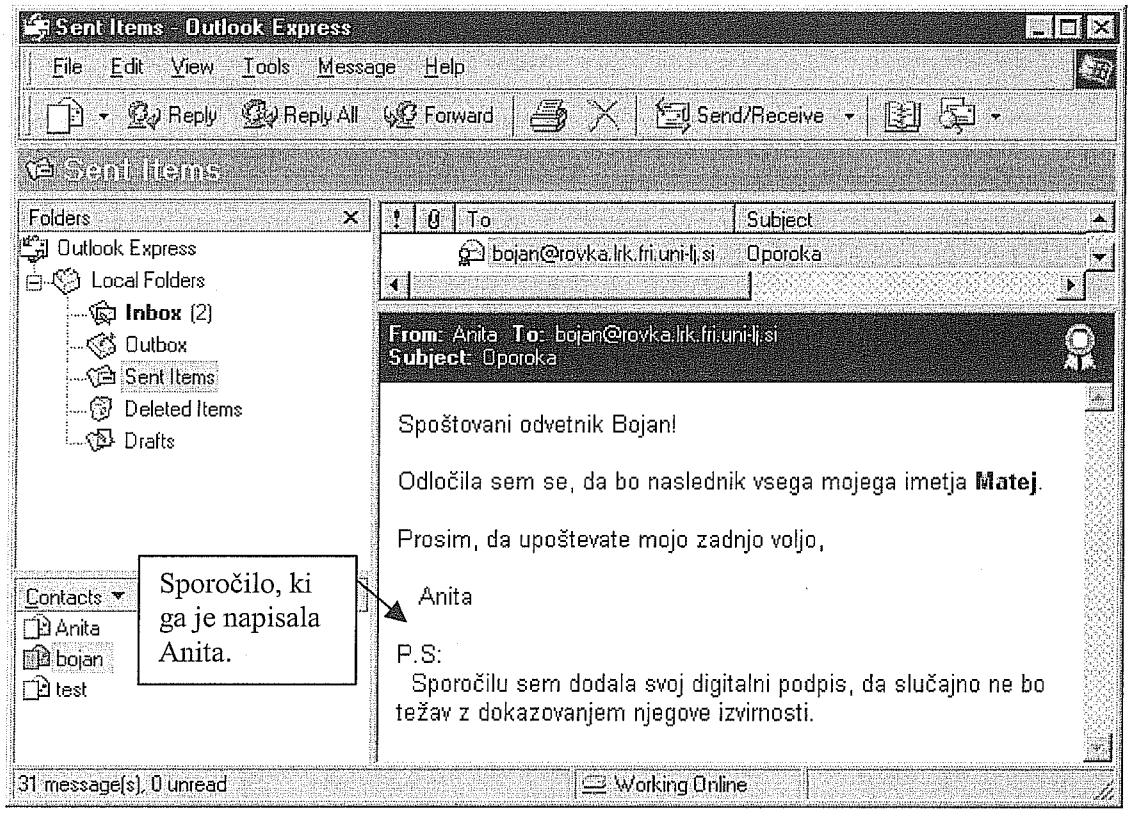
[5] S-MIME Resources,
http://www.rsasecurity.com/standards/smime/resources.html

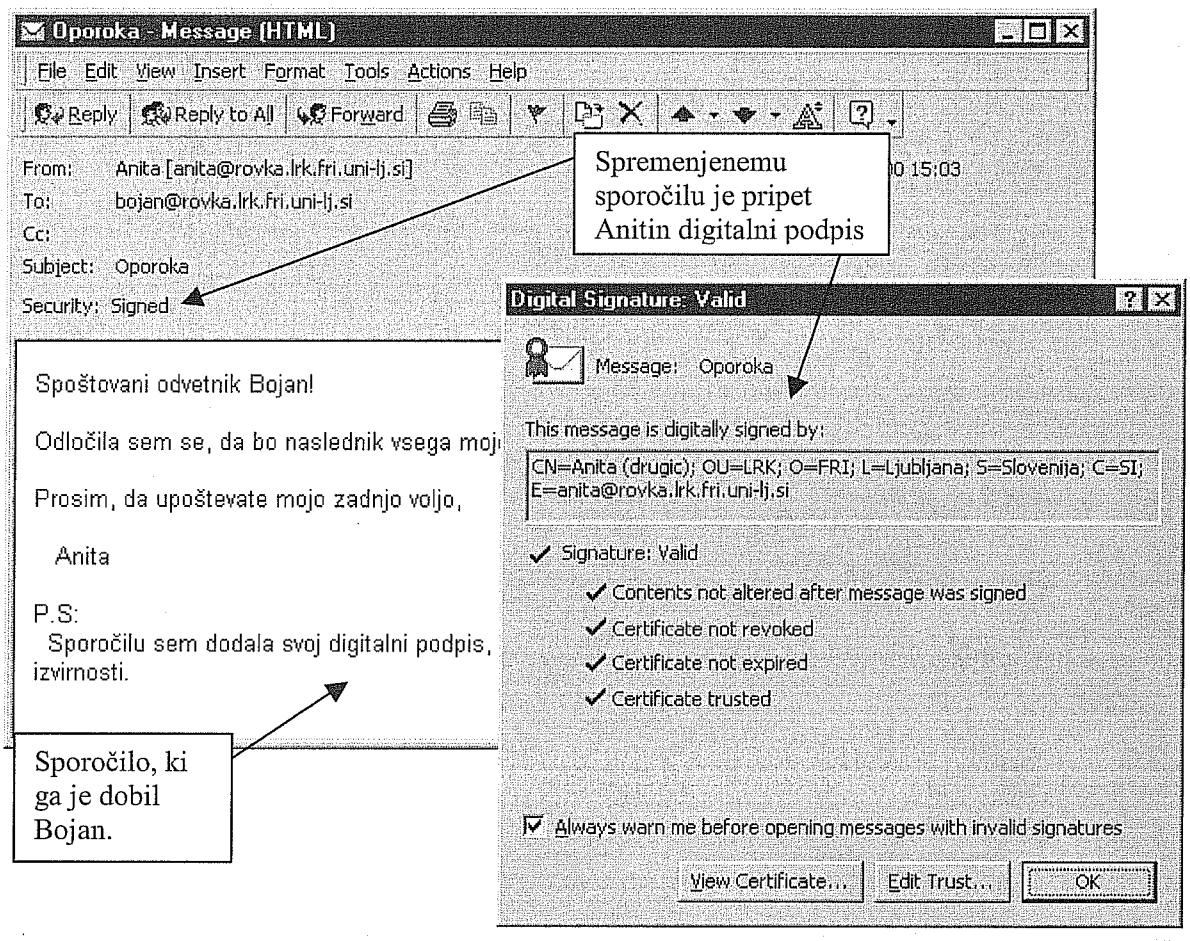
[6] PKCS #7 - Cryptographic Message Syntax Standard
http://www.rsasecurity.com/rsalabs/pkcs/pkcs-7/index.html

[7] Crypto API, 2.0
http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/crypto/portalapi_3351.htm

10 Priloge


Uporaba SSLa za »varno« komunikacijo
Internet trgovina - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address https://arthurdent.lrk2000.fri.uni-lj.si/matejtest/order.htm
E- Trgovina
Spoštovani g. Janez, veseli
Zagotavljamo vam, da je na
zaščito uporabljamo najsod
SLL,....). Prosimo, da izpol
pripeljemo naročeno blago
Količina: Opis:
1
Varna Internet trgovina
1
elektronskega p
Naslov, ki ga je vpisal uporabnik
Naročene blago mi pošljite na naslov:
Skrbni Janez
Pod Borovnicami 1
Varna vas
Naslov
Kliknite spodnji gumb in vaše naročilo bo sprejeto:
Oddaj naročilo
Done
Internet
Certificate
General Details Certification Path
Certificate Information
This certificate is intended to:
•Ensures the identity of a remote computer
Issued to: ARTHURDENT.lrk2000.fri.uni-lj.si
Issued by: Test Win2000 LRK CA
Valid from 22. 12. 99 to 20. 12. 04
Install Certificate...
Issuer Statement
OK
E-trgovina - naročilo je sprejeto - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address https://arthurdent.lrk2000.fri.uni-lj.si/matejtest/AcceptOrder
Go Links
E- Trgovina
Hvala za nakup. Blago boste prejeli po pošti v nekaj dneh.
Klikni tukaj, za prikaz podatkov, ki jih je sprejel strežnik:
Polje Vrednost
Ordering_OrderQty0 1
Ordering_OrderDesc0 Varna Internet trgovina
Ordering_OrderQty1 1
Ordering_OrderDesc1 Pasti in nevarnosti elektronskega poslovanja
Ordering_Address Miha Tatinski Zmikavtska 3 Lopovgrad
Ordering_CardType VISA
Ordering_CardHolderName Skrbni Janez
1621-42234
03/00
Internet
Uporaba SSLa za »varno« komunikacijo
Naslov, ki ga je dobil strežnik


Sent Items - Outlook Express
File Edit View Tools Message Help
Reply Reply All Forward Send/Receive
Sent Items
Folders
Outlook Express
Local Folders
Inbox (2)
Outbox
Sent Items
Deleted Items
Drafts
To
Subject
bojan@rovka.lrk.fri.uni-lj.si Oporoka
From: Anita To: bojan@rovka.lrk.fri.uni-lj.si
Subject: Oporoka
Spoštovani odvetnik Bojan!
Odločila sem se, da bo naslednik vsega mojega imetja Matej.
Prosim, da upoštevate mojo zadnjo voljo,
Anita
P.S:
Sporočilu sem dodala svoj digitalni podpis, da slučajno ne bo težav z dokazovanjem njegove izvirnosti.
Contacts
Anita
bojan
test
Sporočilo, ki ga je napisala Anita.
31 message(s), 0 unread
Working Online


Oporoka - Message (HTML)
File Edit View Insert Format Tools Actions Help
Reply Reply to All Forward
From: Anita [anita@rovka.lrk.fri.uni-lj.si]
To: bojan@rovka.lrk.fri.uni-lj.si
Cc:
Subject: Oporoka
Security: Signed
Spremenjenemu sporočilu je pripet Anitin digitalni podpis
Spoštovani odvetnik Bojan!
Odločila sem se, da bo naslednik vsega moj
Prosim, da upoštevate mojo zadnjo voljo,
Anita
P.S:
Sporočilu sem dodala svoj digitalni podpis,
izvirnosti.
Sporočilo, ki ga je dobil Bojan.
Digital Signature: Valid
Message: Oporoka
This message is digitally signed by:
CN=Anita (drugic); OU=LRK; O=FRI; L=Ljubljana; S=Slovenija; C=SI;
E=anita@rovka.lrk.fri.uni-lj.si
Signature: Valid
Contents not altered after message was signed
Certificate not revoked
Certificate not expired
Certificate trusted
Always warn me before opening messages with invalid signatures
View Certificate...
Edit Trust...
OK