Sigurnost softvera

(2. deo)

Sadržaj

- Reverzni inženjering softvera
- Digital Rights Management
- Razvoj sigurnog softvera
- Softver otvorenog i zatvorenog koda

Reverzni inženjering softvera

Reverzni inženjering softvera.

- *Software Reverse Engineering* (SRE), poznat i kao *Reverse Code Engineering* (RCE).
- Može da se koristi u pozitivne svrhe, kao što su:
  - analiza zlonamernog softvera,
  - proučavanje rada softvera, itd.
- Može da se koristi i u “ne tako pozitivne” svrhe, kao što su:
  - zaobilaženje mehanizama zaštite koji su realizovani u softveru,
  - pronalaženje i zloupotreba nedostataka softvera,
  - varanje u igrama, itd.

Reverzni inženjering softvera

Reverzni inženjering softvera.

- Podrazumevamo da je:
  - napadač onaj koji se bavi reverznim inženjerstvom,
  - napadač ima samo izvršnu (exe) datoteku (nema izvorni kod).
- Namera napadača može da bude:
  - proučavanje softvera kako bi osmislio napad,
  - izmena softvera (češće).
- Razmatraćemo aktivnosti reverznog inženjeringa usmerene na operativni sistem Windows.

Reverzni inženjering softvera

Alati za reverzni inženjering softvera.

- **Disasembler** (*disassembler*), na primer IDA Pro.
  - Konvertuje izvršni (exe) fajl u asembelerski kod – najbolje što može.
  - Nije uvek moguće dobiti dobar (željeni) rezultat.
  - U opštem slučaju, nije uvek moguće od asembelerskog koda, dobijenog disasembliranjem, ponovo dobiti izvršni kod.
- **Dibager** (*debugger*), na primer SoftIce.
  - Omogućava prećenje rada programa (izvršava pojedinačne naredbe, prati vrednost registara, itd).
  - Olakšava analizu – automatizovani postupak.
- ***Hex Editor***.
  - Direktna izmena exe fajla (*patch*).
- Pomoćni alati: Regmon (nadgleda pristup Registry bazi), Filemon (nadgleda pristup datotekama), itd.

Statički i dinamički rezultati.

- Disasembler daje **statičke** rezultate.
  - Dobar je za analizu programske logike.
  - Međutim, ne daje informacije o izvršavanju programa.
  - Teško je pratiti vrednosti promenljivih, registara, grananja u programu, itd.
- Dibager daje **dinamičke** rezultate.
  - Mogu se postaviti tačke prekida (break points).
  - Može se deo koda posmatrati kao “crna kutija”.
  - Disasembler ne daje uvek dobar rezultat za sav kod.
- Disasembler i dibager su neophodni za bilo koji ozbiljniji napad koji se zasniva na reverznom inženjeringu.

Reverzni inženjering softvera

Koja su znanja potrebna?

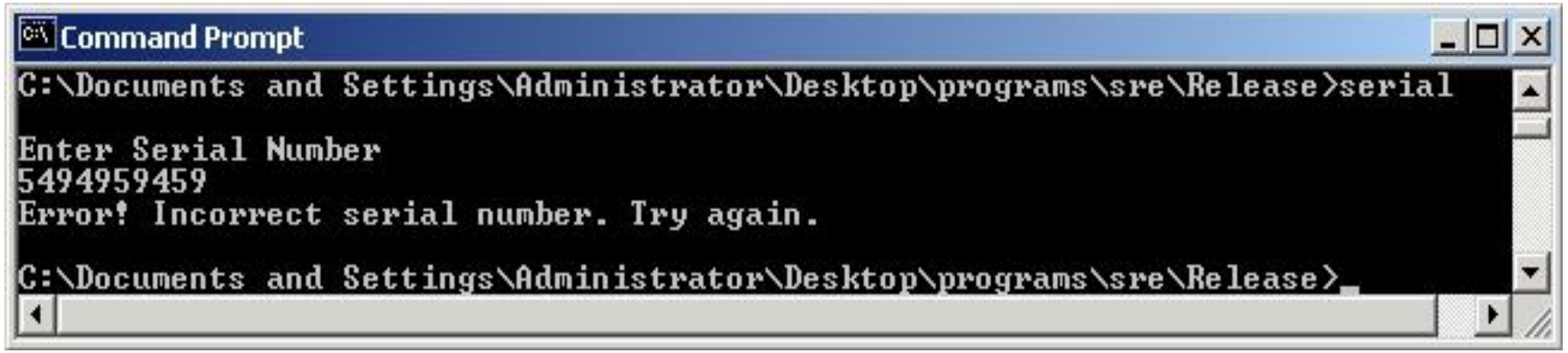
- Potrebna su znanja o asemblerskom kodu softvera koji se analizira.
- Potrebno je iskustvi u radu sa alatima.
  - IDA Pro ima mnogo mogućnosti ali je složen.
  - SoftICE ima veoma obimno uputstvo.
- Potrebno je poznavanje *Windows Portable Executable* (PE) formata datoteka (exe API, DLL, itd.)
- Potrebno je biti strpljiv i uporan, jer je reverzni inženjering softvera težak i iscrpljujući proces!

Primer.

- Analiziraćemo jednostavan primer.
- U ovom primeru je korišćen disasembler i hex editor.
  - Trudi koristi disasembler kako bi razumela kod koji želi da napadne.
  - Trudi, takođe, želi da izmeni kod.
- Napomena: u praksi je, najčešće, potreban i dibager.

Primer.

- Program zahteva unošenje serijskog broja.
- Trudi ne zna serijski broj.
- Da li Trudi može da prevaziđe ovaj problem?



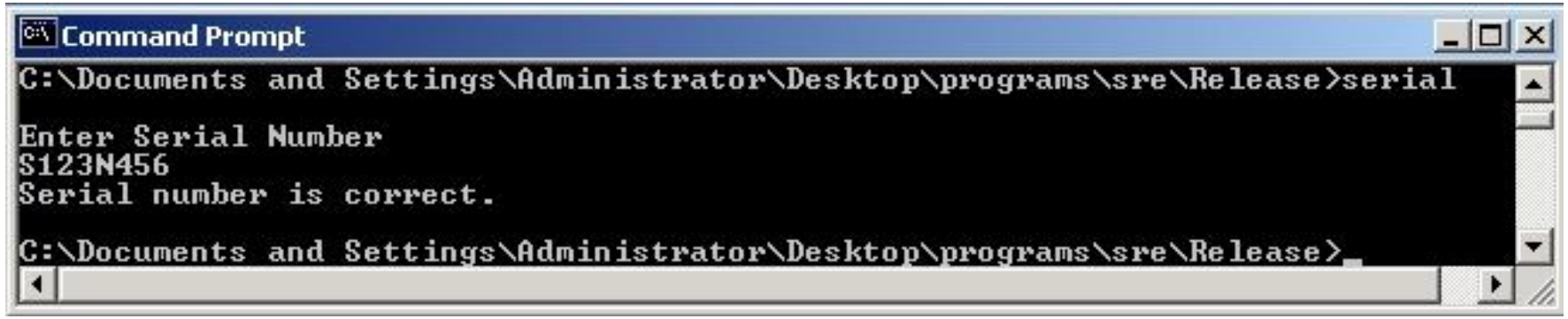
Primer.

- Trudi koristi disasembler.
- Na osnovu linije 0x401022 pretpostavlja da je serijski broj S123N456.

```
.text:00401003    push    offset aEnterSerialNum ; "\nEnter Serial Number\n"
.text:00401008    call    sub_4010AF
.text:0040100D    lea     eax, [esp+18h+var_14]
.text:00401011    push    eax
.text:00401012    push    offset aS       ; "%s"
.text:00401017    call    sub_401098
.text:0040101C    push    8
.text:0040101E    lea     ecx, [esp+24h+var_14]
.text:00401022    push    offset aS123n456 ; "S123N456"
.text:00401027    push    ecx
.text:00401028    call    sub_401060
.text:0040102D    add     esp, 18h
.text:00401030    test    eax, eax
.text:00401032    jz      short loc_401045
.text:00401034    push    offset aErrorIncorrect ; "Error! Incorrect serial number.
.text:00401039    call    sub_4010AF
```

Primer.

- Trudi pokreće program i unosi serijski broj za koji pretpostavlja da je tačan.
- Broj prolazi test, ali se postavlja pitanje da li Trudi može da uradi nešto pametnije.



Primer.

- Trudi ponovo koristi disassembler i uočava naredbu JZ (*Jump if zero*).
- Ukoliko se naredba ne izvrši ispisaće poruku "Error! Incorrect serial number."

```
.text:00401003        push    offset aEnterSerialNum ; "\nEnter Serial Number\n"
.text:00401008        call    sub_4010AF
.text:0040100D        lea     eax, [esp+18h+var_14]
.text:00401011        push    eax
.text:00401012        push    offset aS       ; "%s"
.text:00401017        call    sub_401098
.text:0040101C        push    8
.text:0040101E        lea     ecx, [esp+24h+var_14]
.text:00401022        push    offset aS123n456 ; "S123N456"
.text:00401027        push    ecx
.text:00401028        call    sub_401060
.text:0040102D        add     esp, 18h
.text:00401030        test    eax, eax
.text:00401032        jz      short loc_401045
.text:00401034        push    offset aErrorIncorrect ; "Error! Incorrect serial number.
.text:00401039        call    sub_4010AF
```

Primer.

- Može li Trudi da izmeni program tako da se uvek ispuni uslov za izvršenje naredbe JZ?
- Edituje kod i dobija binarne vrednosti.
- Posmatra adresu 0x401030 neposredno iznad naredbe JZ.
  - `.text: 00401030 test eax, eax`
    `.text: 00401032 JZ`
- Trudi ispituje bite na adresi 0x401030:

Primer.

```
.text: 00401030    test   eax, eax
.text: 00401032    jz     short loc_401405
.text: 00401034    push   offset aErrorIncorect
```

- Naredba `test eax,eax` predstavlja logičku operaciju AND operaciju sadržaja `eax` registra sa samim sobom.
  - Rezultat je 0 ako je sadržaj registra `eax` 0.
  - Ako `test` daje rezultat 0, Izvršava se naradba `jz`.
- Trudi želi da se naredba `jz` uvek izvrši!
- Može li Trudi da izmeni izvršnu datoteku tako da je `jz` uvek tačno?

Primer.

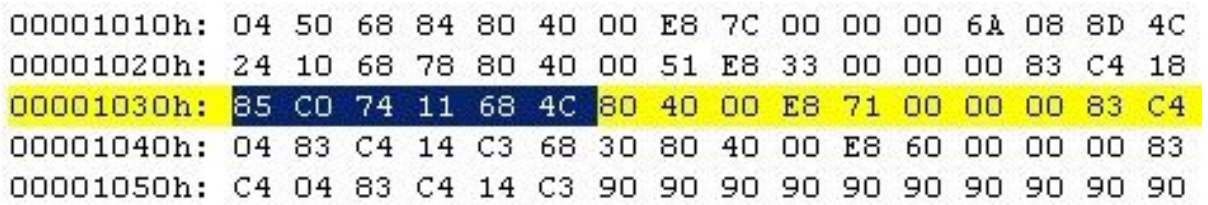
- Može, naredba `xor eax, eax` uvek vraća 0.
- Dakle, Trudi menja naredbu `test eax, eax` naredbom `xor eax, eax`.

```
.text:00401027        push    ecx
.text:00401028        call    sub_401060
.text:0040102D        add     esp, 18h
.text:00401030        test    eax, eax
.text:00401032        jz      short loc_401045
.text:00401034        push    offset aErrorIncorrect ;
.text:00401039        call    sub_4010AF
```

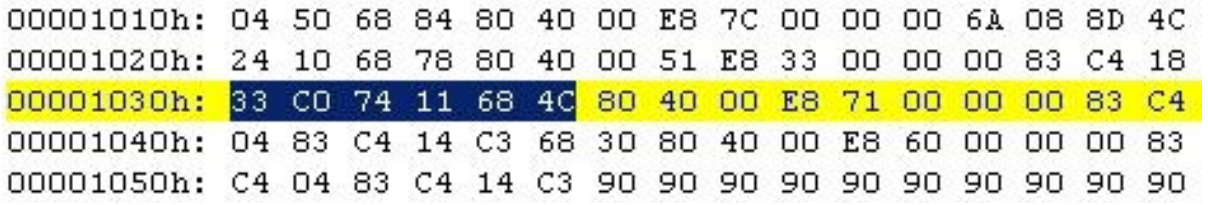
- Da bi to uradila mora da izmeni hex vrednosti u izvršnoj datoteci (pri čemu mora da poznaje vrednost za naredbu `xor`).
- Drugim rečima menja heksadecimalnu vrednost 85 vrednošću 33.

Primer.

- Pre izmene:



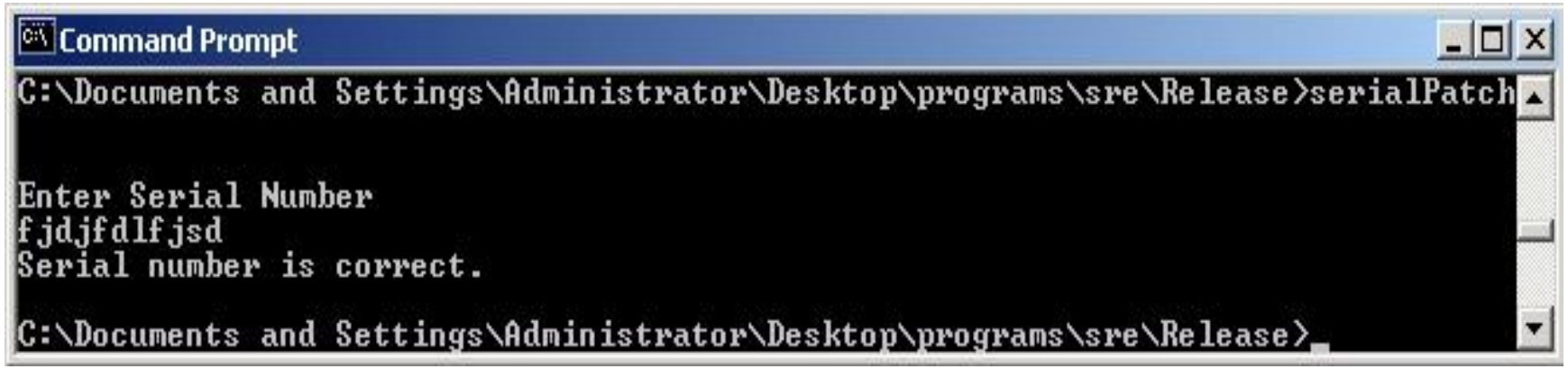
- Posle izmene:



- U ovom slučaju se veličina koda ne menja!

Primer.

- Nakon izmene bilo koji serijski broj prolazi kao tačan.



Primer.

- Ukoliko se disasembliraju izvršne datoteke pre i posle izmene uočava se razlika.

```
.text:00401003                push    offset aEnterSerialNum ; "\nEnter Serial Number\n"
.text:00401008                call    sub_4010AF
.text:0040100D                lea     eax, [esp+18h+var_14]
.text:00401011                push    eax
.text:00401012                push    offset aS       ; "%s"
.text:00401017                call    sub_401098
.text:0040101C                push    8
.text:0040101E                lea     ecx, [esp+24h+var_14]
.text:00401022                push    offset aS123n456 ; "S123N456"
.text:00401027                push    ecx
.text:00401028                call    sub_401060
.text:0040102D                add     esp, 18h
.text:00401030                test    eax, eax
.text:00401032                jz      short loc_401045
.text:00401034                push    offset aErrorIncorrect ; "Error! Incorrect serial number.
.text:00401039                call    sub_4010AF
```

```
.text:00401003                push    offset aEnterSerialNum ; "\nEnter Serial Number\n"
.text:00401008                call    sub_4010AF
.text:0040100D                lea     eax, [esp+18h+var_14]
.text:00401011                push    eax
.text:00401012                push    offset aS       ; "%s"
.text:00401017                call    sub_401098
.text:0040101C                push    8
.text:0040101E                lea     ecx, [esp+24h+var_14]
.text:00401022                push    offset aS123n456 ; "S123N456"
.text:00401027                push    ecx
.text:00401028                call    sub_401060
.text:0040102D                add     esp, 18h
.text:00401030                xor     eax, eax
.text:00401032                jz      short loc_401045
.text:00401034                push    offset aErrorIncorrect ; "Error! Incorrect serial number.
.text:00401039                call    sub_4010AF
```

Reverzni inženjering softvera

Kako se odbraniti od reverznog inženjeringa?

- Nemoguće sprečiti reverzni inženjering u potpunosti, ali je moguće učiniti napad mnogo težim.
  - Koristiti tehnike koje će disasembliranje učiniti manje efikasnim (*anti-disassembly*).
    - Dobijeni asemblerski kod će biti teško razumljiv.
  - Koristiti tehnike koje će debagovanje učiniti manje efikasnim (*Anti-debugging techniques*).
  - Koriste se tzv. *tamper-resistance* tehnike koje se odnose na mogućnost automatske kontrole izmene koda, itd.
  - Maskiranje koda (nepregledan kod) – pisati kod tako da je teško razumljiv, što je suprotno od dobre programerske prakse.

***Anti-dissasemly* tehnike**.

- Ove metode obuhvataju:
  - šifrovanje objektnog koda (ne može se disasemblirati, ali mora postojati kod za dešifrovanje),
  - ubacivanje delova koda koji nema značaja za izvršni program.
  - samo-modifikujući kod, itd.
- Šifrovanje sprečava disasembliranje.
  - Za izvršavanje je neophodno da postoji kod za dešifrovanje!
  - Napadač će ga tražiti.
  - Problem je sličan kao i sa polimorfnim virusima.

***Anti-debugging* tehnike**.

- Koriste se tehnike koje mogu da registruju da li je program pod kontrolom dibagera, na osnovu:
  - pristupa pojedinim registrima,
  - postojanja prekidnih tačaka (*breakpoints*),
  - razlike u memorijskim zahtevima,
  - vremenu izvršavanja, itd.
- Upotreba API funkcija `IsDebuggerPresent()` i `CheckRemoteDebugger()` koje se jednostavno mogu dodati gotovom kodu.
- Dibageri imaju problema sa obradom niti (*threads*).
  - Interakcija niti može da "zbuni" dibager.
- Postoje i mnoge druge metode.
- Ipak, moguće je realizovati dibager čije aktivnosti se teško detektuju.
  - *Hardware-based debugging* (HardICE).

Tamper-resistance.

- Cilj je da se izmena izvršnog fajla učini što težom ili nemogućom.
- Mogu se izračunati heš vrednosti delova koda.
- Ako dođe do izmene, pokazaće se neslaganje heš vrednosti.
- Istraživanja pokazuju da ovo može da bude dobar oblik zaštite, uz minimalno narušavanje brzine izvršavanja softvera.
- Implementacija zaštite mora da se razlikuje od slučaja do slučaja.
  - U suprotnom, napadač može lako da je “zaobiđe”.

Maskiranje koda (*code obfuscation*).

- Cilj maskiranja je učiniti kod teško razumljivim.
  - Kao što je napomenuto, to je suprotno od dobre programerske prakse!
  - Jednostavan primer: *spaghetti code*.
- Mnogo istraživanja kako postići što veći stepen nerazumljivosti.
- Primer:

```
int x,y
…
if((x-y)*(x-y) > (x*x-2*x*y+y*y)) {
…
}
```

- Uslov nije nikada ispunjen, ali napadač gubi vreme na analizu koda.

Maskiranje koda (*code obfuscation*).

- Ova tehnika ima veliku primenu u zaštiti od reverznog inženjeringa.
- Neka istraživanja su pokazala da primena ove tehnike ne može da obezbedi “jaku” zaštitu.
  - “On the (im)possibility of obfuscating programs”, dostupno na adresi http://www.wisdom.weizmann.ac.il/~oded/PS/obf4.pdf
- Iako još uvek daje dobre rezultate u praksi, ne može da se poredi sa kriptološkim rešenjima.

Primer autentifikacije.

- Za autentifikaciju se koristi softver.
- Često se podatak o autentifikaciji beleži kao 1 bit.
- Bez obzira na metod (lozinka, biometrija, itd.) negde u softveru za autentifikaciju, jedan bit određuje da li pristup dozvoljen.
- Ako napadač može da pronađe taj bit, može da zaobiđe i najsloženiji postupak autentifikacije.
- Maskiranje koda može da učini ovu pretragu veoma teškom ili čak nemogućom.
  - Može se kombinovati sa prethodno navedenim tehnikama.

Reverzni inženjering softvera

Kloniranje softvera.

- Neka je napisan neki program.
- Može se svakom korisniku dostaviti identična kopija (klon) tog programa.
- Ako napadač ima uspeha u “razbijanju” jedne kopije, on će moći da isti napad primeni i na sve ostale kopije.
- Ovaj pristup nije otporan na napad tipa BOBE – “*break once, break everywhere*”.
- Kloniranje je uobičajeni pristup u distribuciji softvera.

Metamorfni softver.

- Metamorfizam se koristi kod zlonamernog softvera.
- Može li se iskoristiti i u dobre svrhe?
- Svaka kopija softvera koja se distribuira može da bude različita.
- Moguća su dva nivoa metamorfizma:
  - Kopije koje se distribuiraju imaju različite funkcionalnosti (nema veliku primenu).
  - Svaka kopija ima istu funkcionalnost ali im se interne strukture razlikuju (razmotrićemo ovaj slučaj).

Metamorfni softver.

- Razvoj softvera je uvom slučaju teži, kao i dorada (naknadna ispravka nedostataka, modifikacija, itd.)
- Metamorfizam ne može da spreči reverzni inženjering, ali ga može otežati.
- Metamorfizam može da unapredi otpornost na napade tipa BOBE.

Digital Rights Management

Šta je DRM?

- DRM je skup metoda za ograničavanje korišćenja digitalnih sadržja u cilju zaštite autorskih prava.
- DRM je dobar primer ograničenja primene mehanizama sigurnosti na zaštitu softvera.
- Razmotrićemo sledeće teme:
  - Šta je DRM?
  - Sistem za zaštitu PDF dokumenata.
  - DRM i Internet distribucija audio i video sadržaja.

Digital Rights Management

Šta je DRM?

- Distribuira se digitalni sadržaj.
- Kako zadržati neki oblik kontrole nad primenom i kopiranjem tog sadržaja?
- Primer digitalne knjige.
  - Može da ima veliko tržište.
  - Ali, može da se desi da se proda samo 1 primerak!
  - Lako može da se napravi identična kopija.
  - Velika promena u odnosu na štampana izdanja.
- Slično je sa digitalnom muzikom, video sadržajem, itd.

Digital Rights Management

Osnovni problemi.

- Kako sprovesti definisana ograničenja nakon što se digitalni sadržaj distribuira?
- Primeri ograničenja:
  - zabrana kopiranja,
  - ograničeni broj čitanja/izvršavanja,
  - vremenska ograničenja upotrebe,
  - zabrana prosleđivanja, itd.

Digital Rights Management

Da li kriptografija može da ponudi rešenje?

- Kod distribucije digitalnih materijala, napadač može da bude neko od legalnih korisnika.
  - Drugim rečima, sve što mu je potrebno ima na jednom mestu.
- Kriptografija nije predviđena za ovaj scenario!
- Ipak, kriptografija je neophodna:
  - zbog sigurnog prenosa podataka i
  - radi zaštite od najjednostavnijih napada.
- Napadač, najverovatnije neće pokušati da izvede direktan napad na kripto sistem (to je previše posla).
- Napadač će pokušati da analizom digitalnog materijala (softvera) dođe do ključa.

Digital Rights Management

Trenutno stanje.

- Trenutno su najbolja rešenja koja se zasnivaju na tajnosti primenjenog mehanizma zaštite (*security by obscurity*).
  - Narušeno je osnovno pravilo kriptografije – algoritam zaštite je tajan, što je u suprotnosti sa Kerhofovim principima.
- O zasnivanju DRM na kriptografiji:
  - "*Whoever thinks his problem can be solved using cryptography, doesn't understand his problem and doesn't understand cryptography*." - Roger Needham, Butler Lampson.

Digital Rights Management

Ograničenja.

- Analogno oticanje podataka – prilikom prikaza digitalnog sadržaja, on može da se snimi u analognom obliku (kamera, mikrofon, itd).
  - DRM ne može da spreči ovaj tip napada.
- Ljudska priroda.
  - Potpuna sigurnost u oblasti DRM nije moguća!
  - Kako napraviti nešto što ima praktičnu primenu?
  - Sadržaj utiče na metodologiju zaštite.
- DRM nije problem koji je isključivo tehničke prirode!

Digital Rights Management

DRM zasnovano na softveru.

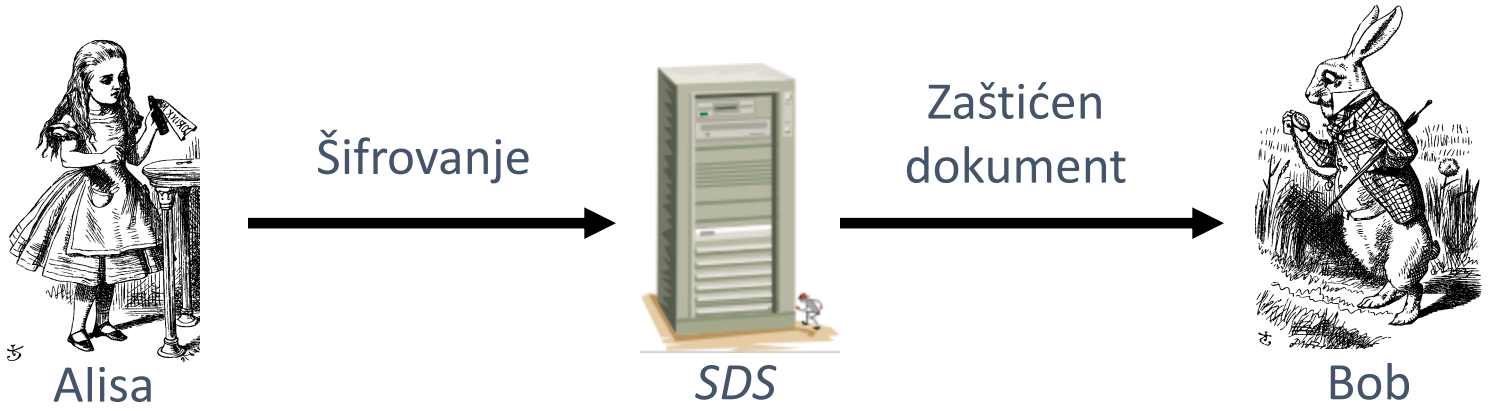
- Ozbiljna zaštita DRM zasnovana na softverskom rešenju nije moguća.
- Zašto?
  - Gotovo je nemoguće sakriti tajnu u softveru.
  - Ne može se u potpunosti sprečiti reverzni inženjering.
  - Korisnici koji imaju potpune administratorske privilegije mogu čak i da zaobiđu implementiranu zaštitu od reverznog inženjeringa.
- Iz ovoga sledi da je najozbiljnija pretnja zaštiti DRM je reverzni inženjering.

DRM za PDF datoteke.

- Raljzmotrićemo rešenje MediaSnap, Inc.
- Razvili su DRM sistem za zaštitu PDF dokumenata.
- Sistem se sastoji iz dve celine:
  - Server – *Secure Document Server* (SDS),
  - Klijent – softver na klijentskoj strani koji se implementira u PDF čitač.

Digital Rights Management

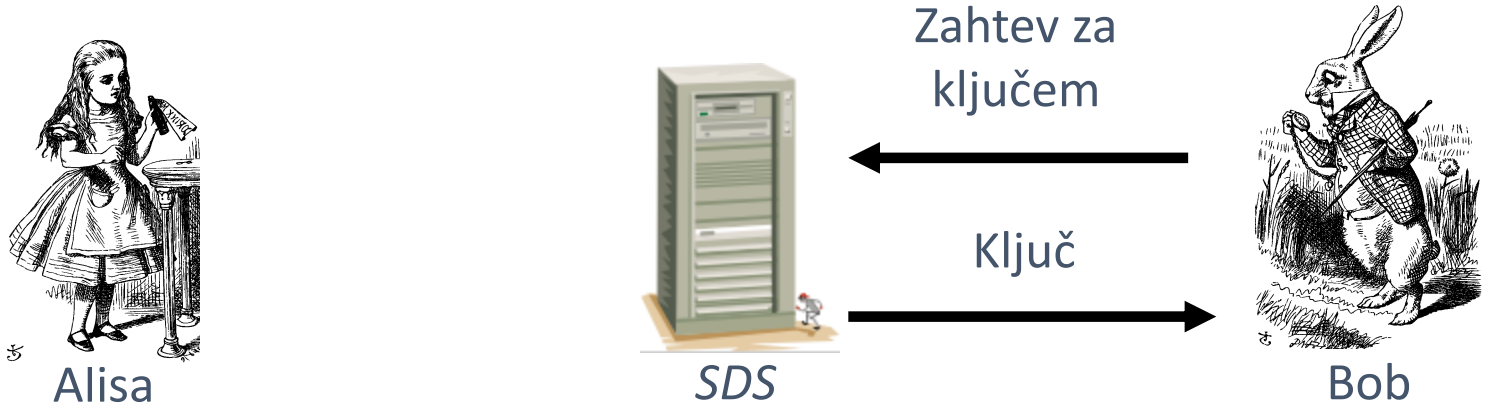
DRM za PDF datoteke.



- Alisa kreira dokument i postavi ga kao prilog u e-pošti.
  - Odabere primaoca i iz padajućeg menija odabere željeni nivo zaštite.
- Poruka se se automatski konvertuje u PDF, potom šifruje i šalje na server (SDS).
- SDS primenjuje željeni nivo zaštite.
- Dokument se šalje Bobu, tako da samo on može da mu pristupi.

Digital Rights Management

DRM za PDF datoteke.



- Bob mora da se autentifikuje na SDS.
- SDS tek tada šalje ključ Bobu.
- Bob sada može da pristupi dokumentu, ali samo preko (klijentskog) DRM softvera.

DRM za PDF datoteke.

- Ograničenja:
  - Na serverskoj strani (SDS) čuvaju se ključevi, podaci za autentifikaciju, itd.
    - Primenjuje se željeni nivo DRM zaštite.
  - Klijentska strana (integrisana u PDF čitač):
    - Zaštita ključeva, autentifikacija korisnika, itd.
    - Omogućiti rad u okruženju u koje ne postoji poverenje.
    - Ostatak diskusije se odnosi na klijentsku stranu.

DRM za PDF datoteke.

- Ograničenja:
  - Na serverskoj strani (SDS) čuvaju se ključevi, podaci za autentifikaciju, itd.
    - Primenjuje se željeni nivo DRM zaštite.
  - Klijentska strana (integrisana u PDF čitač):
    - Zaštita ključeva, autentifikacija korisnika, itd.
    - Omogućiti rad u okruženju u koje ne postoji poverenje.
    - Ostatak diskusije se odnosi na klijentsku stranu.

DRM za PDF datoteke.

- Klijentska strana.
  - Spolješnji nivo zaštite se zasniva na *tamper-resistance* tehnikama.
  - Tehnike maskiranja se primenjuju kao unutrašnji mehanizmi.

DRM za PDF datoteke.

- Klijentska strana – *tamper-resistance*.
  - Kod koji je ugrađen u klijentski softver je šifrovan – sprečava se statička naliza (disasembliranje).
    - U toku rada, dešifruje se deo po deo koda.
  - Primenjene su tehnike koje sprečavaju debagovanje softvra kako bi se onemogućila dinamička analiza.
    - Kontroliše se pristup registrima za debagovanje.
  - Obe tehnike su međusobno zavisne.
  - Softver je pisan na takva način da ga je veoma teško analizirati (tehnika maskiranja).

DRM za PDF datoteke.

- Klijentska strana – maskiranje koda.
  - Maskiranje koda se koristi kod:
    - razmene i čuvanja ključeva,
    - autentifikacije,
    - čuvanja podataka za autentifikaciju,
    - šifrovanja i skremblovanja.
  - Digitalni sadržaj se šifruje AES algoritam.
    - Algoritam je javan.
    - Naknadno se koristi skremblovanje na koje može da se primeni maskiranje.
  - Tehnika maskiranja može samo da uspori napadača.
    - Uporan napadač ima šanse za uspeh!

DRM za PDF datoteke.

- Ostala svojstva.
  - Koriste se heš vrednosti pojedinih delova softvera, odnosno proverava se da li je došlo do izmene.
  - *Anti-screen capture* – zaštita od digitalnog snimanja sadržaja ekrana.
  - *Watermarking*.
    - Teorijski, omogućava praćenje ukradenog sadržaja.
    - Ograničena primena u praksi, posebno ako napadač zna primenjen algoritam.
  - Metamorfizam.
    - Prevencja napada tipa BOBE.
    - Primena kod skremblovanja.

Digital Rights Management

DRM za audio i video zapise.

- Audio i video podaci na Internetu.
  - Često istovremeni prenos, u realnom vremenu ili na zahtev.
  - Ako neko želi da ih naplati, mora da ih zaštiti.
  - Mogu li se zaštiti, tako da ne mogu da se prosleđuju trećoj strani, kopiraju, itd?

Digital Rights Management

DRM za audio i video zapise.

- Napadi na audio i video podatke.
  - *Spoofing attack*.
    - Slično napadu tipa čovek u sredini.
  - Reprodukcija i redistribucija sadržaja.
  - Snimanje nezaštićenog sadržaja.
    - Cilj je da se spreči neautorizovanim korisnicima.
    - Onemogućiti zlonamernom softveru na klijentskoj strani da snimi/prosledi audio i video materijal.

DRM za audio i video zapise.

- Rešenje.
  - Algoritmi za skremblovanje.
    - Algoritmi koji služe za nizak nivo zaštite, nemaju kriptološku vrednost.
    - Postoji monogo različitih algoritama.
    - Primena principa metamorfizma!
  - Neophodan je “dogovor” između servera i klijenta oko izbora algoritma za skremblovanje.
  - Podaci se na serveru:
    - Prvo se skrembluju, potom šifruju.
  - Kod klijenta:
    - Prvo se dešifruju, potom deskrembluju.
    - Kod klijenta: deskrmblovanje je realizovano programski, neposredno pre emitovanja.

DRM za audio i video zapise.

- Algoritmi za skremblovanje.
  - Na serveru postoji veliki skup algoritama za skremblovanje.
    - Recimo N, koji su numerisani od 1 do N.
  - Svaki klijent ima (različit) podskup algoritama za skremblovanje.
    - Na primer: LIST = {12, 45, 2, 37, 23, 31}.
  - LIST se čuva kod klijenta, šifrovan je ključem koji je poznat samo serveru: *E*(LIST, *K*server)

DRM za audio i video zapise.

- Algoritmi za skremblovanje.
  - Na serveru postoji veliki skup algoritama za skremblovanje.
    - Recimo N, koji su numerisani od 1 do N.
  - Svaki klijent ima (različit) podskup algoritama za skremblovanje.
    - Na primer: LIST = {12, 45, 2, 37, 23, 31}.
  - LIST se čuva kod klijenta, šifrovan je ključem koji je poznat samo serveru: *E*(LIST, *K*server).

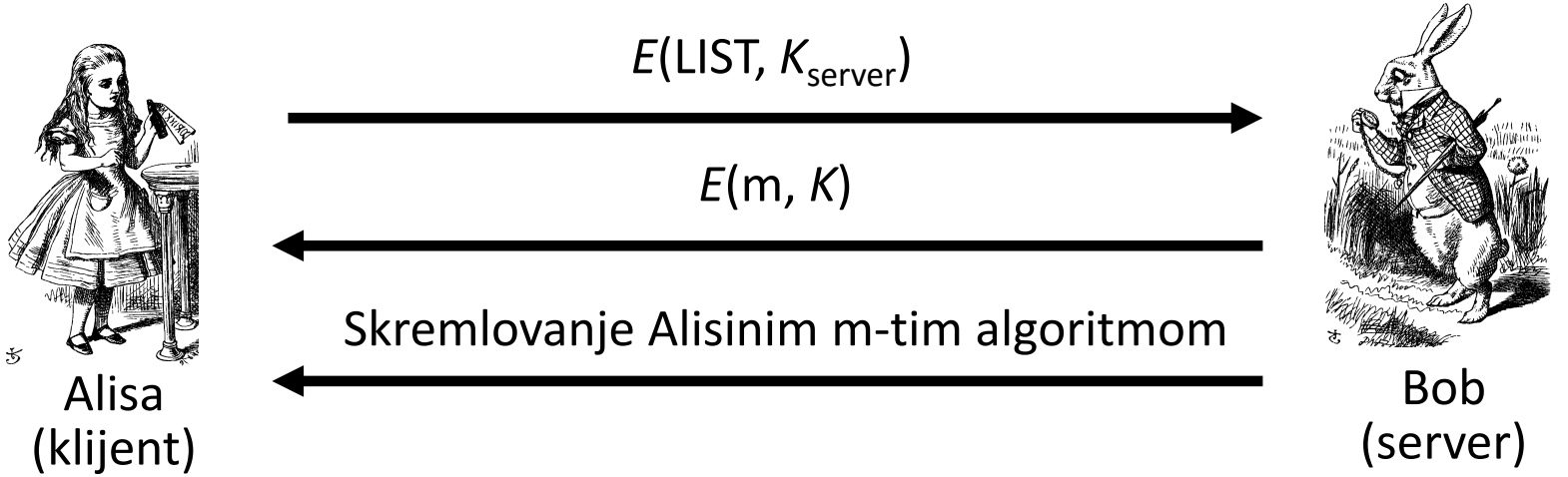
Digital Rights Management

DRM za audio i video zapise.

- Serverska strana.
  - Server mora da skrembluje podatke sa algoritmom koji podržava klijent.
  - Klijen mora da serveru pošalje listu algoritama koje podržava.
  - Server mora da obezbedi zaštićenu komunikaciju za prenos podataka o izabranom algoritmu skremblovanja.

DRM za audio i video zapise.

- Izbor algoritam skremblovanja.



- Server dešifruje listu Alisinih algoritama.
- Odabira jedan i šalje redni broj šifrovano.
- Ključ K je sesijski, pa je potrebna razmena ključa između servera i klijenta.
- Klijent ne može da pročita LIST jer nema K_{server} .

Digital Rights Management

DRM za audio i video zapise.

- Deskremblovanje na klijentskoj strani.
  - Cilj je onemogućiti potencijalnog napadača da dođe u posed podataka.
    - Sprečiti ga da napravi kopiju, itd.
  - Upravljački program (*device driver*) nije javan!
    - Algoritam za skremblovanje je njegov sastavni deo.
    - Deskremblovanje se obavlja neposredno pre reprodukcije.

DRM za audio i video zapise.

- Zašto skremblovanje?
  - Metamorfizam je implementiran na najnižem nivou sistema.
  - Ako je poznato da je neki od algoritama za skremblovanje razbijen (što je realno), server ga neće koristiti.
  - Ako klijent ima mnogo ovakvih algoritama, server može da, preko odgovarajućeg protokola, dostavi klijentu druge algoritme za skremblovanje.
  - Sigurnost ne može da počiva na algoritmima za skremblovanje, zato se šifruje.

Digital Rights Management

DRM za audio i video zapise.

- Zašto metamorfizam?
  - Najbolja je zaštita od reverznog inženjeringa.
  - Napadač neće napasati kriptološki algoritam, pokušaće da nađe ključ.
  - Reverzni inženjering algoritma za skremblovanje može da bude veoma kompikovan.
  - Sigurnost je zasnovana na maskiranju, a pokazuje se kao uspešna u odbrani od napada tipa BOBE.

Plasiraj pa doradi.

- Uobičajen pristup kod razvoja softvera.
  - Razviti proizvod što je pre moguće.
  - Plasiraj ga na tržište bez detaljne analize.
  - Dodaj “zakrpe” za prodati softver kada se uoče nedostaci.
- Sa stanovišta sigurnsoti:
  - loš pristup za razvoj softvera, odnosno
  - najgori mogući pristup po pitanju sigurnosti!

Plasiraj pa doradi.

- Zašto ovaj pristup?
  - Prednost na tržištu:
    - Ko se prvi pojavi može da najviše zaradi.
    - Korisnici obično slede onog ko je prvi plasirao proizvod.
    - Poslodavac neće postaviti pitanje grešaka, sve dok i drugi proizvođači prave slične greške.
    - Korisnici imaju podršku od proizvođača ali i međusobno.

Plasiraj pa doradi.

- Zašto ovaj pristup?
  - Razvoj sigurnog softvera je težak, skup i vremenski zahtevan.
    - Testiranje je skupo i vremenski zahtevno.
    - Lakše je ostaviti korisnicima da traže greške!
  - Nema zakonske odgovornosti.
    - Čak i kad softver ima ozbiljne nedostatke, prizvođač ne snosi zakonske posledice.
    - Da li se neki drugi proizvod prodaje pod ovim uslovima?
    - Može li se ova praksa promeniti u korist kupaca sofrvera?

Plasiraj pa doradi.

- Zabluda: ako redovno preuzimate “zakrpe” softver će biti siguran.
- Zašto je ovo zabluda?
  - Statistika pokazuje suprotno: “zakrpe” često imaju nove nedostatke.

Softver otvorenog i zatvorenog koda

U čemu je razlika?

- Kod softvera otvorenog koda (na primer, Linux) – izvorni kod softvera je dostupan korisniku.
- Kod softvera zatvorenog koda (na primer, Windows) – izvorni kod softvera nije dostupan korisniku.
- Kakve su posledice po pitanju sigurnosti?

Softver otvorenog i zatvorenog koda

Softver otvorenog koda.

- Argument zagovornika ovog pristupa je:
  - Više očiju, veća šansa da se otkrije (sigurnosni) nedostatak.
  - Verzija Kerhofovih principa.
- Da li je argument dobar?
  - Koliko korisnika hoće/može da uoči sigurnosni nedostatak?
  - Koliko korisnika su stručnjaci u oblasti sigurnosti?
  - Napadači, takođe, mogu da analiziraju sigurnosne nedostatke!
  - Programer sa lošom namerom ima mogućnost da podmetne loš kod.

Softver otvorenog i zatvorenog koda

Softver zatvorenog koda.

- Argument zagovornika ovog pristupa je:
  - Sigurnosni propusti nisu direktno vidljivi napadaču.
  - Ovo je oblik sigurnosti zasnovane na maskiranju (*security by obscurity*).
- Da li argument dobar?
  - Zloupotreba nekih propusta ne zahteva poznavanje koda.
  - Mogući su i neki oblici analize zatvorenog koda, mada to zahteva dosta truda i rada!
  - Da li je sigurnost zasnovana na maskiranju zaista sigurnost?

Literatura

1. M. Stamp (2006): *Information Security*. John Wiley and Sons.

Hvala na pažnji

Pitanja su dobrodošla.