Sigurnost softvera

Sadržaj

Uvodne napomene

Zašto softver?

- **Pitanje**: zašto je softver isto toliko važan za sigurnost kao i kriptologija, kontrola pristupa i protokoli?
- **Odgovor**: mehanizmi sigurnosti se veoma često ostvaruju preko softverskih rešenja.
  - Ako je softver predmet napada, sigurnost može biti ozbiljno ugrožena, bez obzira na snagu kriptološkog rešenja, kontrolu pristupa ili primenjene protokole.
  - Softver može da bude loša osnova za primenu sigurnosnih rešenja.

Uvodne napomene

Loš softver – primeri.

- Loš softver je veoma rasprostranjen!
  - Aerodrom u Denveru (SAD).
    - Greške u softveru sistema za opsluživanje prtljaga.
    - Uzrok su kašnjenja otvaranja aerodroma za 11 meseci.
    - Gubici, veći od 1 milion dolara po danu.
  - MV-22 Osprey.
    - Savremena vojna letelica.
    - Izgubljeni životi zbog softverskih grešaka.
  - Organ donor (UK, 2010).
    - Pogrešni organi izvađeni iz 25 donatora u UK.
    - Softver za unošenje informacija o organima koji mogu biti izvađeni iz donator se koristio od 1999. godine i pronađeno je oko 400.000 grešaka.

Uvodne napomene

Kako pojedinci doživljavaju softver?

- “Normalni korisnici”.
  - Greške i propuste najčešće otkrivaju slučajno.
  - Mrze loš softver, ali moraju da nauče da žive i rade sa njim.
  - Moraju da “nateraju” loš softver da radi.
- Napadači.
  - Aktivno traže greške i propuste.
  - Obožavaju loš softver i pokušavaju da iskoriste njegove slabosti.
  - Napadaju sistem koristeći slabosti lošeg softvera.

Uvodne napomene

Odnos broja linija koda i grešaka.

- Gruba procena: 5 grešaka na 1.000 linija koda.
- Iz toga sledi:
  - Tipičan računar: 3.000 izvršnih fajlova, 100.000 linija koda svaki od njih.
  - U proseku 50 grešaka po izvršnoj datoteci.
  - Sledi 150.000 grešaka po računaru.
  - U mreži sa 30.000 čvorova ima 4,5 milijardi grešaka.
  - Ako je samo 10% od njih sigurnosno kritično, a od toga samo 10% podložno spoljašnjem napadu:
    - Onda je “samo” 4.5 miliona sigurnosno kritičnih grešaka kao posledica lošeg softvera u toj mreži!

Uvodne napomene

Analiza rezultata.

- Jednostavna aritmetrika daje:
  - dobre vesti “lošim momcima” i
  - loše vesti “dobrim momcima”.
- Teško je pretpostaviti da se ove greške mogu u potpunosti prevazići.
- Realan cilj: efektivno upravljanje sigurnosnim rizikom u datom okruženju.
- Apsolutna sigurnost najčešće ne može da se postigne.
  - Softver nije izuzetak.

Uvodne napomene

Propusti i greške.

- Nedostaci softvera sa aspekta sigurnosti (*sofware vulnerabilities*) pružaju mogućnost napadaču da povećaju bezbednosi rizik informacionog sistema.
- Ovi nedostaci mogu da se podele u dve grupe:
  - **propuste** (*flaws*) i
  - **greške** (*bugs*).

Uvodne napomene

Propusti i greške i zlonamerni programi.

- Propusti i greške.
  - Nenamerne mane softvera koje se mogu zloupotrebiti.
  - Mogu se podeliti na više grupa:
    - prepunjavanje bafera (*buffer overflow*)
    - nepotpuna kontrola, itd.
- Zlonamerni programi (*malicious software*).
  - Rezultat namere.
  - Vrste:
    - virusi,
    - crvi, itd.

Uvodne napomene

Propusti u programiranju.

- Greška u programiranju se naziva *error* ili *bug*.
  - Nastaje kao posledica rada programera.
  - Razmatraju se greške koje se ne mogu otkriti u toku prevođenja programa.
- U toku izvršavanja programa greška može da dovede program u **neželjeno stanje** (*fault*).
  - Predstavlja interno stanje programa.
- Neželjeno stanje programa može da onemogući izvršavanje programa na očekivani način (*failure*).
  - Može da se registruje u toku rada programa.

Uvodne napomene

Primer.

```
char array[10];
for(i = 0; i < 10; ++i)
  array[i] = 'A';
array[10] = 'B';
```

- Ovaj program ima **grešku** (*error*).
- Greška može da prouzrokuje **neželjeno interno stanje** (*fault*).
- Ako se dogodi neželjeno interno stanje, može doći do problema kod izvršavanja programa (*failure*).
- Termin **propust** ili nedostatak (*flaw*) objedinjuje: *error*, *fault* i *failure*.

Uvodne napomene

Siguran softver.

- Osnovni zahtev softverskog inženjerstva:
  - napisati program koji radi ono što se od njega očekuje.
- Zahtev sigurnosti u softverskom inženjerstvu:
  - softver radi ono što se od njega očekuje i ništa više.
- Nemoguće je realizovati potpuno siguran softver.
- Kako upravljati rizikom?

Uvodne napomene

Sigurnosni propusti.

- Najčešće nastaju nenamerno, ali predstavljaju sigurnosni rizik.
- Razmotrićemo:
  - prepunjavanje bafera i
  - nepotpunu kontrolu.
- Postoje i mnoge druge mane, ali su ove vrlo česte.

Prepunjavanje bafera

Šta je bafer, a šta prepunjavanje bafera?

- Bafer je deo memorije, definisane veličine, namenjen za čuvanje podataka.
- Najčešće je definisan početnom adresom, tipom podataka i veličinom memorije koja mu je dodeljena.
- Prepunjavanje bafera je upis veće količine podataka od predviđenog memorijskog prostora koji je rezervisan za bafer.
  - Ukoloko se pri upisu podataka u bafer ne vodi računa o količini podataka i veličini bafera, može doći do prepunjavanja bafera!

Prepunjavanje bafera

Uobičajeni scenario napada.

- Korisnik unosi podatke preko Web stranice.
  - Ime, datum rođenja, adrese, itd.
- Web stranica šalje podatke serveru koji ima bafer dugačak *N* karaktera (bajtova).
- Server upisuje podatke u bafer, ali pri tom ne proverava dužinu ulaznog podatka.
- Može doći do prepunjavanja bafera.
  - Može doći i do pada sistema.
- U nekim slučajevima, prepunjavanje bafera može da se iskoristi za napad.

Prepunjavanje bafera

Primer.

```
int main() {
  int buffer[10];
  buffer[20] = 37;
}
```

- Pitanje: šta će se desiti u toku izvršavanja programa?
- Odgovor: zavisi od toga šta se nalazi na memorijskoj lokaciji buffer[20].
  - Može se prepisati deo korisničkih podataka ili koda.
  - Može se prepisati deo sistemskih podataka ili koda.

Prepunjavanje bafera

Primer.

- Razmotrimo slučaj softvera za autentifikaciju.
- Informacija o autentifikaciji se čuva u jednom bitu (0 nije, ili 1 jeste autentifikovan).
- Prepunjavanjem bafera može da se prepiše sadržaj tog bita (*flag*) i da se dozvoli Trudi da se predstavi kao Alisa!

Prepunjavanje bafera

Organizacija memorije.

- *Text* čuva kod.
- *Data* čuva statičke promenljive.
- *Heap* čuva dinamičke podatke.
- *Stack* čuva:
  - dinamičke lokalne promenljive,
  - parametre funkcije,
  - povratnu adresa (povratna adresa predstavlja adresu na kojoj će se nastaviti izvršavanje koda nakon izlaska iz funkcije),
  - *stack Pointer* – SP (krajnja adresa stack dela memorije).
- Napomena: memorijske lokacije bafera se nalaze iznad memorijske lokacije na kojoj se čuva povratna adresa!

Prepunjavanje bafera

Prepunjavanje bafera na steku.

- Šta su posledice prekoračenja bafera?
- Program se “vraća” na pogrešnu adresu.
- Moguć je pad sistema.
- Ovo može da bude jedini cilj napada, ali obično nije ...

Prepunjavanje bafera

Prepunjavanje bafera na steku.

- Trudi ima bolju ideju – ubacivanje koda.
- Trudi može da natera sistem da izvrši kod koji ona želi!
- Postavlja povratnu adresu na adresu početka bafera.

Prepunjavanje bafera

Prepunjavanje bafera na steku.

- Problemi za Trudi:
  - Ne zna tačnu lokaciju na kojoj se u steku nalazi povratna adresa.
  - Ne zna tačnu lokaciju na kojoj se u steku nalazi program koji ona želi da se izvrši.
- Međutim, ovo nisu nepremostive prepreke!
  - Trudi će ispred željenog koda staviti više praznih naredbi (NOP).
  - Trudi će postaviti više novih povratnih adresa.
  - Ako bar jedna od povratnih adresa bude na pravom mestu, verovatno će pogoditi ili NOP ili početak željenog koda.

Prepunjavanje bafera

Zaključne napomene.

- Da bi se izveo ovaj napad mora u programu da postoji propust koji omogućava prepunjavanje bafera.
- Ne mogu se svi propusti ovog tipa zloupotrebiti.
- Ako mogu, napadač može da ubaci svoj kod.
- Sistem nasumičnih pokušaja je jedan od modela napada.

Prepunjavanje bafera

Zaključne napomene.

- Alternativna metoda:
  - Trudi ne mora da poseduje izvorni kod programa.
  - Dovoljna je izvršna datoteka.
  - Jedini alat koji joj je potreban je disasembler kako bi odredila povratnu adresu.

Nepotpuna kontrola

Provera ulaznih vrednosti.

- Razmotrimo funkciju: strcpy(buffer, argv[1])
- Prepunjavanje bafera će se dogoditi ako je: len(buffer) < len(argv[1]).
- Program mora da proveri ulazne vrednosti kontrolišući dužinu argv[1].
- Nedostatak ovakvih provera vodi jednom opštijem problemu koji se naziva **nepotpuna kontrola** (*incomplete mediation*).

Nepotpuna kontrola

Provera ulaznih vrednosti.

- Neka ulazni podaci dolaze sa Web stranice.
- Neka ih kontroliše klijent i prosledi serveru.
- Primer – validni ulazni podaci:
  http://www.things.com/orders/final&custID=112&
  num=55A&qty=20&price=10&shipping=5&total=205
- Neka ih server ne kontroliše.
- Zašto bi se kontrolisali na dva mesta?
- Trudi može direktno da pristupi serveru i pošalje:
  http://www.things.com/orders/final&custID=112&
  num=55A&qty=20&price=10&shipping=5&total=25

Zlonamerni softver

Nije ništa “novo”.

- Dizajniran je da naruši bezbednost.
- Tipovi zlonamernog softvera.
  - Virus – pasivno širenje.
  - Crv – aktivno širenje.
  - Trojanski konj – neočekivana funkcionalnost.
  - *Trapdoor*/*backdoor* – omogućava neautorizovan pristup.
  - *Rabbit* – nenamensko trošenje sistemskih resursa.

Zlonamerni softver

Gde se nalazi zlonamerni softver?

- U opštem slučaju, bilo gde.
  - *Boot sector*.
    - Preuzima kontrolu nad celim sistemom.
  - *Memory resident*.
    - Nalazi se učitan u radnoj memoriji.
  - Aplikacije, makroi, ...
  - U bibliotekama funkcija.
  - Kompajleri, dibageri, kvazi “antivirusni” softveri.
    - Oni su posebno “prljavi”!

Zlonamerni softver

Virus.

- Virus je računarski kod koji može da se integriše na postojeći program ili datoteku i da se tako prenosi sa računara na računar.
- Može da bude destruktivan i da pri tome ošteti podatke i softver na zaraženom računaru.
- Za širenje među računarima često je potrebna akcija čoveka, mada postoji mogućnost da se šire i pomoću crva.

“Brain” virus.

- Pojavio se 1986. godine.
- Više neprijatan nego štetan.
- Šta je mogao da uradi?
  1. Postavljao se u *boot* sektor (i na druga mesta).
  2. Pratio je sve pozive programa sa diska kako bi sprečio svoju detekciju.
  3. Na svaku akciju čitanja diska, proveravao je da li se nalazi u *boot* sektoru – u slučaju da nije, idi na prvi korak.
- Nije imao štetne posledice.
- Nije ozbiljno shvaćen.
  - Poslužio kao prototip ostalim virusima.

Zlonamerni softver

Crv.

- Ima mogućnost da se automatski kopira sa jednog računara na drugi.
  - Postupci mogu biti složeni.
  - Preuzima (delimično) kontrolu nad nekim funkcijama računara za prenos podataka.
- Mogu veoma brzo da se šire.
- Nakon distribucije mogući su različiti scenariji napada.

Zlonamerni softver

Morisov crv.

- Pojavio se 1988. godine (autor je tada bio student).
- Internet je bio u povoju.
- Šta je pokušavao da uradi:
  - Da odredi gde može da se širi.
  - Da zarazi što više resursa u mreži.
  - Da ostane neotkriven.
- Moris je tvrdio da se test “oteo kontroli”.
- “Nedostatak” u kodu crva – pokušavao je da ponovo inficira već zaražene sisteme.
  - Veliko zauzeće sistemskih resursa.
  - Sličan efekat kao kod zlonamernog softvera tipa *rabbit*.

Zlonamerni softver

Morisov crv.

- Kako se širio?
- Pokušavao je da pristupi mreži preko:
  - Pogađanja korisničke lozinke.
  - Zloupotrebe prepunjavanja bafera preko nekih servisa u Unix OS (konkretno, fingerd).
  - Zloupotrebe zadnjih vrata u sendmail paketu.
- Nedostaci u ovim aplikacijama su bili poznati u to vreme, ali nisu bili preduzeti pravi koraci da se oni koriguju.
- Internet je bio projektovan za akademsku zajednicu.

Morisov crv.

- Kada se “ostvari” pristup drugom računaru.
  - “*Bootstrap loader*” se šalje novoj žrtvi.
- Sastoji se od 99 linija C koda.
- Na računaru žrtve se prevodi i izvršava.
- *Bootstrap loader* tada prihvata ostatak koda potrebnog za funkcionisanje crva.
- Pri tome je računar žrtve čak autentifikovao pošiljaoca!

Zlonamerni softver

Morisov crv.

- Kako nije detektovan?
- Ako se prenos crva prekine, preuzeti deo koda se automatski briše.
- Kod je bio šifrovan prilikom preuzimanja (*download*).
- Preuzeti kod se brisao nakon dešifrovanja i prevođenja.
- U toku rada crv je često menjao ime.

Zlonamerni softver

Morisov crv.

- Šokirao je Internet zajednicu 1988. godine (Internet je tada bio mnogo drugačiji).
- U to vreme Morisov otac je radio za NSA.
- Moglo je biti mnogo gore – kod nije bio zlonameran.
- CERT (*Computer Emergency Responce Team*) je ustanovljen radi računarske sigurnosti.

Zlonamerni softver

Code red crv.

- Pojavio se u julu 2001. godine.
- Zarazio je više od 250.000 sistema za samo 15 sati.
- Ukupno je zaraženo oko 750.000 sistema.
- Iskoristio je nedostatak prepunjavanja bafera u Microsoft IIS serveru.
- Pratio je saobraćaj na portu 80 u potrazi za serverima na koje bi se proširio.

Code red crv.

- Šta je radio?
- Od 1. do 19. dana pokušavao je da se proširi po mreži.
- Od 20. do 27. dana izvođenje napad tipa DDoS usmerenih na više adresa, među kojima je bila i www.whitehouse.gov.
- Kasnije verzije (više varijanti):
  - Obuhvataju instaliranje zadnjih vrata na zaraženom računaru radi neautorizovanog pristupa.
  - Postojalo je mišljenje da je Code Red mogući “beta test” za “informacioni rat”.

Zlonamerni softver

SQL Slammer.

- Pojavio se 2004. godine.
- Zarazio 250.000 sistema za 10 minuta!
  - Širio se suviše brzo.
  - Potpuno je zagušio Internet saobraćaj.
  - Da je mogao da kontroliše brzinu širenja sigurno bi zarazio mnogo više računara.
- U čemu je tajna uspeha?
  - Ceo crv može da se zapiše u jedan 376 bajtski UDP paket.
  - Mrežne barijere su bile podešene da propuste male pakete, podrazumevajući da ne mogu da budu štetni, a tek nakon njih nadziru konekciju.
  - Pretpostavka je bila da je za napad potrebno mnogo više podataka.
- Slammer je iskoristio zaključke “stručnjaka”.

Zlonamerni softver

Trojanski konj.

- Računarski programi koji nalikuju korisnim programima.
  - Jednim delom to i jesu.
  - Ostatak programa najčešće čini štetu, odnosno nije pod kontrolom korisnika u sigurnosnom smislu.
- Drugim rečima, govorimo o softveru koji pored očekivane ima i dodatne skrivene funkcije.
- Šire se tako što navode korisnike da pokrenu aplikacije misleći da su iz legalnih izvora.

Detekcija zlonamernog softvera.

- Tri uobičajene metode:
  - detekcija potpisa,
  - detekcija promena,
  - detekcija anomalija.

Detekcija potpisa.

- Potpis predstavlja tačnu vrednost niza bita koji su sastavni deo nekog softvera.
  - Može da bude i heš vrednost.
- Pretpostavimo da virus ima potpis: 0x23956a58bd910345
- Moguće je tražiti taj potpis u svim datotekama.
- Ako se nađe datoteka koji sadrži ove podatke, da li smo sigurni da je pronađen virus?
  - Ne, isti potpis mogu da imaju i druge datoteke.
  - Ako je sadržaj datoteke slučajan, mala je verovatnoća lažne uzbune: $1/2^{64}$.
  - Softver nije slučajan, verovatnoća je veća.

Detekcija potpisa.

- Prednosti:
  - Daje dobre rezultate za pretragu poznatog malvera, u slučajevima kada su njegove kopije nepromenjene.
  - Zahteva minimalno angažovanje korisnika / administratora.
  - Potrebno je samo poznavanje potpisa i periodična provera (skeniranje).
- Nedostaci:
  - Ako je datoteka sa bazom potpisa velika, skeniranje može da bude relativno sporo.
  - Datoteka sa bazom potpisa mora da bude ažurna.
  - Ne može se detektovati nepoznati zlonamerni softver.
  - Ne može se detektovati novi tip zlonamernog softvera.
- Do sada je detekcija potpisa najpopularniji metod detekcije zlonamernog softvera.

Detekcija promene.

- Zlonamerni softver mora da se nalazi negde na sistemu.
- Ako detektujemo da se neka datoteka izmenila, postoji mogućnost da je npr. zaražena virusom.
- Kako detektovati promene?
  - Izračunati heš vrednosti datoteka i čuvati ih na sigurnom mestu.
  - Periodično izračunati heš vrednosti i upoređivati ih sa onim koje su sačuvane.
  - Ako se heš vrednosti razlikuju, može se uvesti pretpostavka da je datoteka zaražena.

Detekcija promene.

- Prednosti:
  - Greška u detekciji je svedenena na minimun.
  - Može se detektovati i zlonamerni softver čiji tip nije prethodno bio poznat ili softver koji evolvira.
- Nedostaci:
  - Mnoge datoteke se menjaju – i to često – što za posledicu ima veliki broj lažnih uzbuna.
  - Mnogo posla za korisnika / administratora koji se odnosi na donošenje odluke da li je detektovana promena posledica napada.
- Pitanje: ako se virus ubaci u datoteku koja se često menja, da li će biti detektovan?
- Pitanje: da li je kombinacija detekcije promene i detekcije potpisa bolje rešenje?

Detekcija anomalija.

- Anomalije se detektuju tako što se nadziru aktivnosti na sistemu i traže bilo koje “neobične” aktivnosti ili one koje su “nalik-virusu”.
- Šta je neobično?
  - Izmene datoteka na neuobičajeni način.
  - Ponašanje sistema.
  - Neuobičajene mrežne aktivnosti.
  - Neuobičajeni pristup datotekama.
- Mora se prvo definisati šta je “normalno” ili “uobičajeno”.
- Ono što je normalno je podložno promenama!

Detekcija anomalija.

- Prednosti:
  - Mogućnost otkrivanja nepoznatog zlonamernog softvera i softvera koji evolvira.
  - Nema potrebe za ažuriranjem baze potpisa.
- Nedostaci:
  - Nedovoljno provereno u praksi.
  - Trudi može učiniti da neuobičajeno izgleda uobičajeno (spore ali kontrolisane promene).
  - Mora se kombinovati sa drugim metodama (kao što je detekcija potpisa).
  - Popularna je u IDS sistemima.
- Detekcija anomalija je složen problem!

Polimorfni zlonamerni softver.

- Polimorfni crv je obično šifrovan.
  - Novi ključ se koristi svakoga puta kada se crv preseli na novu lokaciju.
  - Šifrovanje nije složeno (npr. višestruki XOR) zato što poverljivost nije cilj.
  - Crv nema isti potpis.
- Međutim mora da postoji kod da bi se dešifrovao.
  - Detekcija potpisa može da traži baš taj kod.
  - Moguće je detektovati potpis, ali je postupak nešto složeniji.

Metamorfni zlonamerni softver.

- Metamorfni crv mutira pre nego što inficira novi sistem.
- Takav crv može da onemogući detekciju potpisa.
- Mutirani crv treba da ima istu funkcionalnost kao i originalni, ali mora biti dovoljno različit da bi sprečio detekciju.
- Detekcija ovakvog zlonamernog softvera nije generalno rešen problem.

Zlonamerni softver

Metamorfni zlonamerni softver.

- Da bi se umnožio, crv se disasemblira.
  - Slučajne vrednosti se umeću u kod.
  - Naredbe skokova i grananja se rearanžiraju.
  - Ubacuje se beskorisni kod, itd.
- Ponovo se asemblira rezultujući kod.
- Rezultat je crv sa istom funkcionalnošću kao i polazni, ali sa drugačijim potpisom.

Zlonamerni softver

Warhol crv.

- Andy Warhol: “In the future everybody will be world-famous for 15 minutes”.
- Warhol crv je dizajniran da zarazi ceo Internet za 15 minuta.
- Slammer je zarazio 250,000 sistema za 10 minuta.
  - Zagušio je propusni opseg i nije uspeo da zarazi ceo Internet za 15 minuta.
- Može li se dizajnirati crv koji će biti efikasniji od Slammer-a?

Warhol crv.

- Jedan pristup dizajnu:
  - Napraviti početnu listu koja će sadržati skup ranjivih IP adresa.
    - Koje su to adrese, zavisi od konkretnog cilja.
    - Postoje dostupni alati za analizu ranjivosti.
  - U početnom napadu svaka IP adresa sa početne liste će biti zaražena (jer je ranjiva).
    - Sa svake od tih IP adrese može da ispita prethodno određeni deo IP adresnog prostora.
  - Moguće udvostručenje broja zaraženih, bez zagušenja saobraćaja.
- Nije poznata efikasna realizacija ovakvog crva.

Flash crv.

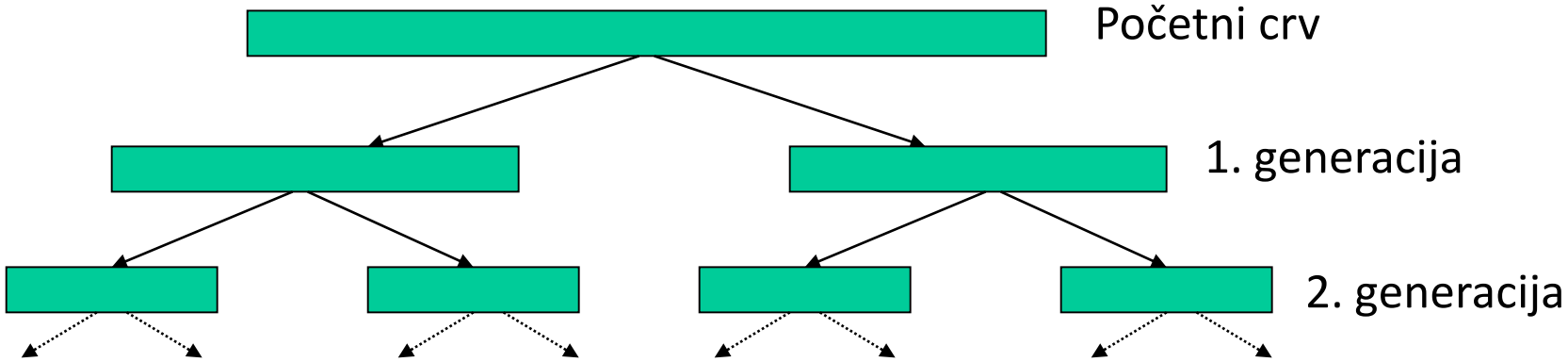
- Da li je moguće efikasnije rešenje od Warhol crva?
- Može li ceo Internet biti zaražen za manje od 15 minuta?
  - Pretraga ranjivih IP adresa je zahtevan posao u bilo kom napadu.
  - Pretraga može da bude ograničena propusnim opsegom.

Flash crv.

- Jedan pristup dizajnu:
  - Odrediti sve IP adrese ranjivih računara.
    - Veoma zahtevan posao, ali postoji dosta alata za ovu namenu.
    - Pretraga zavisi od vrste napada koji treba da se primeni.
  - Zapisati sve te adrese u deo koda crva.
    - Rezultat je relativno veliki crv (oko 400KB)
  - Kad god se crv umnoži, on se smanji.
    - Svaki novi crv napada duplo manje adresa.
  - Za samo nekoliko generacija, crv dostiže optimalnu veličinu.
- Snaga ovog pristupa: vremenski efikasan, nema zagušenja saobraćaja!

Flash crv.

- Optimizacija veličine crva: u svakoj novoj generaciji smanjuje se broj adresa koje crv napada.



Flash crv.

- Procena je da u idealnim uslovima flash crv može da zarazi ceo Internet za 15 seskundi!
- To je mnogo brže nego što čovek može da registruje napad tako da je neophodna automatizovana odbrana.
- Moguća odbrana od flash crva:
  - Angažovati mnogo nezavisnih sistema za detekciju napada (IDS).
  - Treba da postiji master IDS koji će nadgledati pojedinačne IDS.
  - Kada master IDS detektuje neuobičajene aktivnosti, on ih obradi na nekoliko čvorova, i blokira na svim ostalim.
  - Ako je analizirani čvor stvarno napadnut, zaustavljeno je širenje crva, ako ne uvodi se kašnjenje u sistem.

Ostali napadi

Napadi koji ne spadaju u prethodne kategorije.

- *Salami attack*.
- Napad linearizacijom.
- Vremenske bombe (*time bombs*).

Ostali napadi

Salami attack.

- Predstavlja seriju malih (beznačajnih) napada, koji kada se mnogo puta ponove predstavljaju ozbiljan napad.
- Rezultat su, najčešće, namenski pisanog dela programa koji programer integriše u program koji piše za nekog naručioca.
- Žrtve teško mogu da primete da su premet napada.
- Najčešće se omogućava krađa veoma malog novca od svake transakcije, koja se prebacuje na račun programera.

Ostali napadi

Salami attack.

- Primeri:
  - Banka računa kamatu na deponovani novac na svakom računu.
  - Programer od svake izračunate kamate, neki mali iznos (npr. par centi) prebacuje na svoj račun.
  - Klijenti ne primećuju da su oštećeni.
  - Banka teško može da primeti prevaru.
  - Tokom vremena, na računu programera može da se nađe velika količina novca!

Ostali napadi

Salami attack.

- Primeri:
  - Takvi napadi mogu da se urade iznutra.
  - Da li se stvarno događaju?
  - Programer je povećao iznos poreza na platu svakog zaposlenog za nekoliko centi.
    - Razliku je prikazivao kao porez koji je on uplatio.
    - Rezultat je bio, ne samo izbegavnje plaćanja poreza, nego i povraćaj novca!
  - Programiranje automata na benziskoj pumpi tako da prikaže da je izdato više goriva nego što jeste.
    - Predvideli su i da za količine od 20 ili 40 litara automat radi tačno.
    - Inspektori obično sipaju toliko.

Ostali napadi

Salami attack.

- Primeri:
  - Zaposleni je reprogramirao Taco Bell kasu tako da se svaki iznos od $2.99 registruje kao $0.01.
    - Zaposleni je na svakoj prodaji ove vrednosti imao zaradu od $2.98
    - Prilično velika vrednost za ovaj tip napada.
  - Ova vrsta napada može da se odnosi i na tehniku prikupljanja informacija.
    - Prikupljaju se pojedinačni podaci iz mnogo izvora, da bi se dobile objedinjeni podaci o pojedincu ili organizaciji.
    - Sporo ali nije nedostižno.

Ostali napadi

Napad linearizacijom.

- Program proverava unos serijskog broja S123N456.
- Na primer, zbog efikasnosti, proverava jedan po jedan karakter.
- Može li napadač ovo da iskoristi?

```
# include <stdio.h>
int main (int argc, const char *argv[]) {
  int i;
  char serial [9]="S123N456"
  for (i=0; i<8; ++i) {
    if (argv[1][i] != serial [i]) break;
  }
  if (i==8) {
    printf("\nSerial number is correct\n");
  }
}
```

Ostali napadi

Napad linearizacijom.

- Provera korektnog unosa traje duže od pogrešnog!
- Napadač isprobava stringove koji se razlikuju samo po prvom karakteru.
  - Mereći vreme, nalazi da je S tačan karakter.
- Postavi S za prvi karakter i isprobava različite karaktere na drugoj poziciji (S*xxxx).
  - Pronalazi da S1 zahteva najduže vreme, itd.
- Napadač može da otkrije serijski broj, karakter po karakter!

Ostali napadi

Napad linearizacijom.

- Koja je prednost ovog pristupa?
- Neka serijski broj ima 8 karaktera i neka se biraju iz skupa od 128 mogućih vrednosti, što znači da imamo $128^8 = 2^{56}$ mogućih serijskih brojeva.
- Napadač treba da isproba oko 2^{55} vrednosti da bi slučajno došao do serijskog broja, što u prevodu znači – mnogo posla!
- Koristeći napad linearizacijom, posao se svodi na oko $8 \times (128/2) = 2^9$, što je zanemarivo malo u odnosu na klasičan napad potpunom pretragom.

Ostali napadi

Vremenske (logičke) bombe.

- Primer:
  - 1986. godine Donald Gene Burleson je zahtevao od poslodvca da od njegove zarade ne upalćuje porez.
  - Kako ovo nije bilo po zakonu, zahtev je odbijen.
  - Zaposleni je najavio tužbu.
    - Koristio je računare na poslu za pripremanje tužbe.
    - Poslodavac mu je to zabranio.
  - Burleson je razvijao zlonamerni softver.
  - Nakon što je otkriven, njegov softver je pobrisao na hiljade dokumenata važnih za poslodavca.

Ostali napadi

Vremenske (logičke) bombe.

- Primer (nastavak):
  - Kompanija ga nije tužila, bojeći se reakcije javnosti.
    - Ali Burleson jeste, tražio je obeštećenje za gubitak posla!
  - Ipak je kompanija uspela da dobije proces.
    - 1988. godine Burleson je kažnjen sa 11.800$
    - Proces je trajao 2 godine.
    - Koštao je hiljade $.
    - Kazna je bila simbolična.
  - Ovo je bio jedan od prvih slučajeva računarskog kriminala koji je sudski okončan.
  - Ima mnogo sličnih slučajeva.
  - Kompanije uglavnom izbegavaju tužbe.

Literatura

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

Hvala na pažnji

Pitanja su dobrodošla.