Razvoj bezbednog softvera

Informaciona bezbednost

Fakultet tehničkih Nauka, Univerzitet u Novom Sadu

Imre Lendak, 2020

Sadržaj današnjeg predavanja

- Primeri iz prakse
- Nenamerne greške
- Analiza izvornog koda
- Alati za analizu izvornog koda
- Obfuskacija izvornog koda

Bezbedan izvorni kod

HEARTBLEED

Heartbleed

- **DEF:** Heartbleed je slabost OpenSSL biblioteke koja je potencijalno omogućavala hakerima pristup lozinkama na Web serverima ili mobilnim telefonima koji su konektovani na javne bežične mreže
  - Common Vulnerabilities and Exposures oznaka: CVE-2014-0160
- **Izvor:** nenamerna softverska greška, čitanje van granica bafera
- **Mesto:** OpenSSL kriptografska biblioteka, *heartbeat* ekstenzija za održavanje bezbednih komunikacionih kanala
- **Nastanak:** 11. decembar 2011 – push koda sa greška
- **Detekcija:** april 2014. godine
  - Navodno je detektovan od strane Google-ovog tima za sigurnost
  - Neel Mehta (Google) je od Internet Bug Bounty inicijative dobio nagradu od 15,000 USD za odgovorno nalaženje i prijavu problema
- **Korekcija:** 7. aprila 2014. godine

Heartbleed - Posledice

- Afektovani entiteti:
  - ~17% web servera sa TLS implementacijom (uglavnom Red Hat i ostale Linux distribucije)
  - ~50 miliona Andorid 4.1.1 uređaja – za čitanje lozinki sa uređaja je potreban pristup serveru sa zlonamernom implementacijom OpenSSL heartbeat ekstenzije
- Poznate zloupotrebe:
  - 4.5 miliona kartona pacijenata ukradeno od Community Health Systems (drugi najveći na polju plaćene medicinske nege u USA)
- Ponovno kreiranje ~30,000 X.509 digitalnih sertifikata (od ukupno 500,000+)
- Navodno je Agencija za nacionalnu bezbednost SAD (NSA) znala za slabost od početka i čuvala ga kao zero-day koji su koristili po potrebi
  - SAD je (navodno) promenio strategiju i „idu“ na čišćenje svih sličnih problema umesto njihovog čuvanja u tajnosti i zloupotrebe
- Osnovan Core Infrastructure Initiative sa vođstvom Linux Foundation i sa ciljem da finansira vodeće razvojne inženjere ključnih biblioteka
  - Finansira: plate, putovanja, eksterne analize izvornog kod
  - Članovi inicijative: Google, Facebook, Amazon, Dell, Microsoft, itd.

Bezbedan izvorni kod

OS, PROCESI I NITI

Operativni sistem

- **DEF:** Operativni sistem (OS) upravlja računarskim hardverom i softverom i pruža zajedničke servise aplikativnim programima
- **DEF:** Kernel (jezgro) operativnog sistema omogućava komunikaciju sa hardverom preko sledećih servisa:
  - Rad sa ulazno-izlaznim uređajima, npr. tastatura, štampač, monitor
  - Rad sa memorijom
  - Rad sa procesima i nitima
  - Realizacija raznih komunikacionih protokola
  - Upravljanje sa skladištima podataka, npr. fajl sistemom

Proces

- **DEF:** Program je skup instrukcija (tj. programskog koda) u trajnoj memoriji
- **DEF:** Proces je primerak programa koji OS izvršava
- Multitasking je usluga OS koja omogućava da se više procesa izvršavaju naizmenično i dele resurse sistema, npr. CPU, memorija
  - OS može da izvršava više primeraka procesa (isti program)
- Stanje (kontekst) proces u izvršavanju se sastoji od:
  - CPU kontekst, npr. vrednosti registara, adresiranje fizičke memorije
  - Instrukcije, tj. izvršni mašinski kod programa
  - Call stack (pozivi aktivnih pod-rutina),
  - Podaci u radnoj memoriji
  - Deskriptori resursa potrebnih za izvršenje, npr. otvoreni fajlovi
  - Stanje sinhronizacionih promenljivih, npr. mutex
- Preključivanjem procesa na CPU se „sklanja“ kontekst prethodnog i stavlja kontekst novog procesa
  - Preključivanje je „skupa“ operacija

Nit

- **DEF:** Nit (engl. thread) izvršava deo zadatka procesa – takođe se prekljućuje na procesoru
  - Nit je donekle sličan procesu, u sklopu kojeg izvršava jedan zadatak
  - Nit može da deli podatke sa drugim nitima u okviru istog procesa
- **Kontekst niti** se sastoji od sledećih informacija
  - CPU kontekst
  - Podaci za upravljanje nitima, npr. stanje sinhronizacionih promenljivih
- Kontekst (tj. stanje) niti je manji od konteksta procesa
- **Prednosti** upotrebe većeg broja niti:
  - OS je u stanju da brže kreira, prekljućuje i briše niti (u poređenju sa procesom)
  - Povećane performanse zbog konkurentnog izvršavanja na više procesora u modernim računarima
  - Blokirajući sistemski pozivi ne blokiraju ceo proces, samo jednu nit
- **Izazov:** softverski inženjeri moraju da vode računa o konzistentnosti podataka i koda u procesu u kojem se izvršava veći broj niti

Debugging osnove

- Debugging je tehnika otkrivanja grešaka u programskom kodu
- Problemi i greške koje je moguće rešiti na osnovu debugging-a
  - Ispad aplikacije, npr. plavi ekran u Windows OS
  - Slabe performanse aplikacije, npr. sporo izvršavanje
  - Curenje memorije, tj. nekontrolisan rast procesa u memoriji
- Debug je moguć u razvojnom okruženju (npr. Visual Studio) kada je izvorni kod programa dostupan i kada je moguće ponavljanje greške u laboratorijskom okruženju
  - Na ključna mesta u kodu se stavi tačka prekida (*breakpoint*)
  - Izvršavanje programa i duboka analiza toka izvršavanja oko tačaka prekida
- Windows Debugger i slični debugging alati se koriste kada izvorni kod nije dostupan, ili kada se greška ispoljava u okruženju gde nije moguće instalirati razvojne alate i postaviti izvorni kod

Bezbedan izvorni kod

NENAMERNE GREŠKE

Uvod

- **DEF:** Nenamerne greške su slabosti sistema koje se slučajno unose (za razliku od malware)
- Nenamerne greške su slabosti distribuiranih informacionih sistema koje su moguće u sledećim domenima
  - **Fizičke tačke pristupa:** nezaštićen prozor, nedovoljno visoka zaštita od poplave, ljudski element, npr. izostanak adekvatne obuke, neadekvatna provera kandidata koji se postavljaju na osetljiva radna mesta
  - **Elektronske tačke pristupa:** komunikaciona oprema, npr. permit all pravilo na firewall, softver, npr. greška u izvornom kodu koja se koristi od strane crva za „upad“ u sistem

- U sledećih nekoliko slajdova diskutujemo nenamerne greške u softveru

Tipovi

1. Neadekvatna validacija ulaza i parametara
2. Provera softvera
3. Ranjiva autentifikacija
4. Ranjiva autorizacija
5. Ranjivosti u održavanju konfiguracija
6. Ranjivosti u upravljanju sesijama

1. Neadekvatna validacija ulaza (NVU)

- Do greške dolazi kad program primi ulaz van predviđenih granica
- Nevalidan ulaz može da dovede do greške u programu i eventualnog otkaza
- **Specifični primeri:** buffer overflow i overread, SQL Injection
  - Njih analiziramo kroz slajdove koji slede
- **Konkretni primeri iz prakse:**
  - Validacija putanje do fajlova na FTN-ovom sajtu u prostoru za nastavne materijale
  - Datum čija je vrednost 30. februar

1.1 *Buffer overflow*

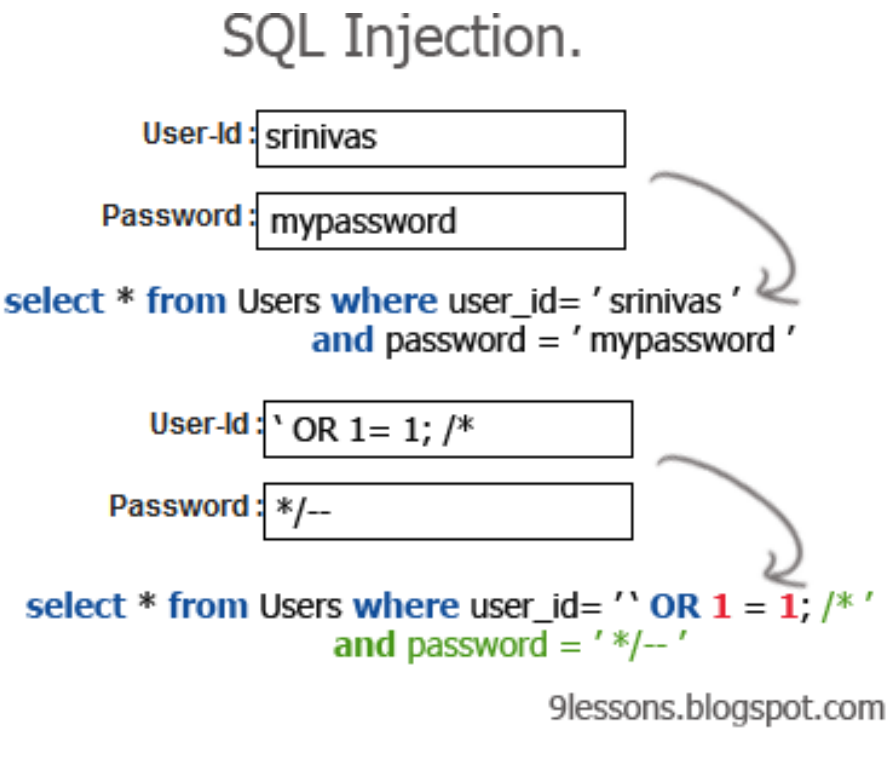
- **DEF:** Bafer (*buffer*) je memorijski prostor zadate veličine
- **DEF:** Buffer overflow slabost u omogućava pisanje van zauzetog bafera
  - Strogo uzevši možemo da smatramo podvrstom neadekvatne validacije
- Neki kompajleri ne generišu kod za proveru veličine bafera zarad boljih performansi koda
- Potencijalna mesta upisa van granica bafera
  - Podaci korisnika
  - Izvršni kod korisnika
  - Podaci operativnog sistema (OS)
  - Izvršni kod OS
- Napadač može da ubaci instrukcije u prostor za sistemski kod koje OS izvrši sa većim privilegijama

1.2 *Buffer overread*

- **DEF:** Buffer overread slabost (u izvornom kodu) omogućava napadaču da čita podatke iz radne memorije van bafera (uglavnom „iza“ bafera)
- Buffer overread je uspešan ukoliko OS ili aplikacije imaju predvidiv način rada sa memorijom, tj. kada napadač poznaje način rada OS i/ili aplikacija sa memorijom
- **Address space layout randomization (ASLR)** je mera bezbednosti OS za otežavanje zloupotrebe slabosti zaštite bafera
  - Prvo uveden u Linux kernel 2001. godine
  - Smešta stek i sl. osetljive sadržaje na slučajne lokacije u radnoj memoriji
  - Značajno otežava napadačima da pogode mesto traženih sadržaja u memoriji
- Ostale tehnike zaštite koje pružaju moderni (Windows) OS: Data Execution Prevention (DEP), Control Flow Guard (CFG)

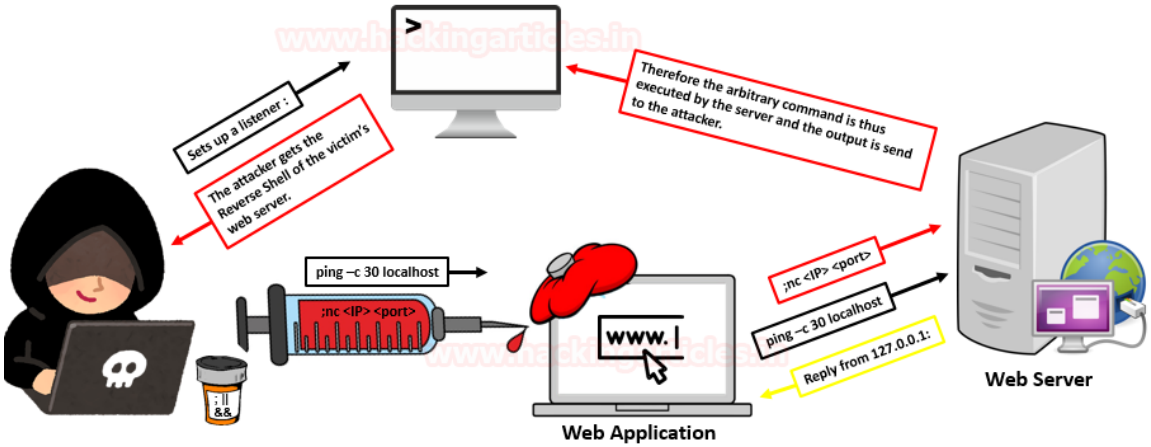
1.3 SQL injection

- Tip napada na data-driven aplikacije
  - Omogućava neželjeno čitanje, modifikaciju ili uništavanje podataka
  - Uglavnom se koristi kao napad na web sadržaje
- Primer: Nelegitiman SQL kod se doda u polje za unos
- **Napomena:** 2020. godine je manje relevantan zbog postojanja jakih counter-a



1.4 (OS) command injection

- Tip napada na aplikacije koje imaju mogućnost posrednog izvršavanja shell komandi
- Neadekvatna provera formata poziva tih aplikacija može dozovoliti napadaču izvršenje neželjenih OS komandi



https://www.hackingarticles.in/comprehensive-guide-on-os-command-injection/

1.5 Cross-site scripting

- DEF: Cross-site scripting (XSS) je nenamerna softverska greška koja afektuje web sajtove
  - XSS omogućava hakerima da ubace skripte koje se izvršavaju u Internet pretraživačima ili na Web serverima
  - XSS je identifikovan od strane Microsoft-ovih inženjera 2000. godine
  - Jedno vreme je (tokom 2000ih) XSS bio najčešći vid napada na Web sadržaje
- Najčešći i najpoznatiji tipovi XSS:
  - **Perzistentni** – ubacivanje klijentskih skripti u sadržaje koji se čuvaju na serveru i prikazuju drugim korisnicima, sa ciljem krađe osetljivih informacija od drugih korisnika, npr. <script> tag na kraju poruke na forumu
  - **Reflektovani (ne-perzistentni)** – ubacivanje skripti u polja za unos podataka u HTML formama i izvršavanje skripti na serverskoj strani
- Zaštitne mere: validacija ulaza, tj. provera postajanja ubačenog koda, isključivanje skripti na klijentskoj strani

https://owasp.org/www-community/attacks/xss/

Neadekvatna validacija parametara

- Nedozvoljena izmena klijentskog keša, npr. cookie manipulation
- Izmena HTTP zaglavlja (header manipulation)
- Izmena URL upita

2. Provera softvera

- **DEF:** Neadekvatna provera (izmena) softvera može dovesti do instalacije ili ubacivanja izvršnog koda na trajnom skladištu ili u radnoj memoriji
- Potencijalne ciljevi napadača su
  - Zaobilaženje koda za proveru licenci
  - Zaobilaženje koda za autentifikacij/autorizaciju
  - Ubacivanje backdoor-ova
- U kontekstu industrijskih kontrolnih sistema je moguća neželjena izmena firmvera industrijske opreme, npr. PLC-ova
- U kontekstu desktop OS su moguće izmene izvršnih fajlova ili biblioteka, kako na disku, tako i u radnoj memoriji
- Metodi zaštite:
  - Pokretanje svih aplikacija pod nalogom običnih korisnika (ne admin ili root)
  - Ažuriranje softvera sa proverenih izvora (proizvođač)
  - Ograničenje fizičkog pristupa računarima i industrijskim uređajima i njihovim (fizičkim) ulazno-izlaznim uređajima

3. Ranjiva autentifikacija

- **DEF:** Ranjivosti podsistema za autentifikaciju mogu da se zloupotrebe i omoguće pristup neautentifikovanim korisnicima (ili napadačima)
  - Npr. postoji login stranica, ali veb-baziran sistem omogućava pristup drugim stranicama i delovima sistema i bez unošenja korisničkog imena i lozinke
- Metodi koje napadači najčešće koriste su:
  - Zaobilaženje login procedura
  - Brute-force napadi na podsistem za autentifikaciju
  - Cookie replay
  - Pass-the-hash napad
- Metodi zaštite su:
  - Više-faktorska autentifikacija
  - Zabrana pristupa nakon višestrukih (npr. 3 do 5) neuspešnih pokušaja autentifikacije
  - Korišćenje slučajnog sadržaja na login skrinu

4. Ranjiva autorizacija

- **DEF:** Ranjivosti podsistema za autorizaciju mogu da se zloupotrebe i omoguće pristup funkcijama računarskog sistema kojima korisnik ne treba da ima pravo pristupa po planu autorizacije
  - Ove ranjivosti uglavnom znače pristup funkcijama koje su predviđene za korisnike sa višim nivoom pristupa, eng. privilege escalation
- Metodi koje napadači najčešće koriste su:
  - Elevacija privilegija
  - Narušavanje tajnosti osetljivih informacija
  - Narušavanje integriteta osetlljivih informacija, npr. izmena „recepta“ u Stuxnet
- Metodi zaštite su:
  - Primena politike najmanjih privilega (eng. least privilege)
  - Zatvaranje korisničkih sesija nakon isteka predviđenog vremena (timeout)
  - Redovna kontrola prava pristupa svih korisnika

4.1 Mesto autorizacije i izvršenja

- **DEF:** Ukoliko se autorizovane operacije snimaju u red čekanja od strane OS, onda je potencijalno moguća zloupotreba u kojoj se autorizovane operacije modifikuju ili fabrikuju
  - Npr. zamena operacije nad fajlom sa kopiranja na brisanje
- Redosled operacija u modernim računarima može da bude drugačiji od očekivanog
  - Konkurentno izvršavanje
  - Optimizacije u operativnom sistemu
  - Optimizacije u samom CPU
- Mera bezbednosti: onemogućiti izmenu autorizovanih operacija (u redovima čekanja)

5. Ranjivosti u održavanju konfiguracija

- **DEF:** Ranjive ili podrazumevane konfiguracije uređaja i softvera omogućavaju neželjen pristup napadačima
- Slabosti koje napadači koriste:
  - Podrazumevana podešenja, npr. prazne ili poznate podrazumevane lozinke veb konfiguracionog panela
  - Pokrenuti bespotrebni servisi koji sadrže slabosti
  - Ne obrisani podrazumevani, bekap i sample fajlovi, aplikacije, sajtovi i skriptovi
  - Slaba podešenja pristupa fajl sistemu
  - Poruke o greškama koje otkrivaju više od potrebnog (narušen *need-to-know*)
  - Neadekvatna primena sertifikata, npr. podrazumevani, self-signed
- Metodi zaštite su:
  - Rigorozni proces upravljanja konfiguracijama koji sadrže detaljne procedure za otklanjanje gore navedenih slabosti
  - Provere konfiguracija pre instalacije i ažuriranja sistema
  - Periodične provere (cele) konfiguracija sistema

5.1 Otkrivanje kritičnih informacija

- **DEF:** Kod slabosti tipa otkrivanje kritičnih informacija su osetljive informacije dostupne na jednostavan način
- Kritične informacije, npr. parovi korisničkih imena i lozinki
  - Zakodirane u kodu aplikacije
  - U konfiguracionim fajlovima u čistom tekstu
- Mere bezbednosti protiv ove klase slabosti:
  - Svi kredencijali treba da su kriptovani i skladišteni u za to predviđenim servisima
  - Dokumentacija ne sme da bude dostupna bilo kome

6. Ranjivosti u upravljanju sesijama

- **DEF:** Komunikaciona sesija je niz zahteva i odgovora koje dva elementa sistema razmene u konačnom vremenskom intervalu
  - Ukoliko se održava stanje sesije, ono sadrži: podatke o autentifikaciji i autorizaciji, opis poslednjih zahteva i odgovora i druge promenljive
- **DEF:** Ranjivosti u rešenjima za otvaranje, održavanje i gašenje komunikacionih sesija mogu dovesti do zloupotreba sistema
- Metodi koje napadači najčešće koriste su:
  - Preuzimanje sesija, tj. session hijacking
  - Ponavljanje sesija, tj. session replay
  - Man-in-the-middle (MitM) napadi
- Metodi zaštite su:
  - Korišćenje slučajnih ključeva sesija koji su jedinstveno asocirani sa korisnicima
  - Praćenje komunikacionih sesija
  - Zatvaranje sesija (eksplicitno ili nakon isteka predviđenog vremena)

Bezbedan izvorni kod

ANALIZA IZVORNOG KODA

Statička analiza

- **DEF:** Statička analiza omogućava detekciju potencijalnih problema na polju performansi, stila i poznatih tipova slabosti **bez izvršavanja izvornog koda** koji je predmet analize
  - Ulaz statičke analize je izvorni kod u originalnom ili nekom prevedenom obliku (objektni kod)
  - Pretpostavka statičke analize je da se izvršava automatski na računaru u posebnom softverskom alatu
  - Engleski naziv: Static Application Security Testing (SAST)
  - Also known as (AKA): White Box Testing
- Statička analiza omogućava kreiranje softverskih metrika, tj. broj linija koda, klasa, metoda, petlji, granjanja, itd.
- Primeri pravila: skrivanje metoda baznih klasa, ponovno bacanje izuzetaka, enumeracije imaju 0 kao vrednost, itd.
- Code review, tj. ručna analiza izvornog koda od strane nezavisnog eksperta (najčešće drugog programera) je podvrsta statičke analize

Dinamička analiza – 1

- **DEF:** Dinamička analiza omogućava detekciju potencijalnih problema na polju performansi, stila i poznatih tipova slabosti **tokom izvršavanja** procesa
  - Engleski naziv: Dynamic Application Security Testing (DAST)
  - Also known as (AKA): Black Box Testing
- Negativni test – ispitivanje ponašanja nakon postavljanja neočekivanih ili nevalidnih ulaznih podataka (npr. 30. februar za datum)
- Ključno je da se kreira dovoljno raznolik „prostor“ ulaznih parametara da bi se „prošao“ što veći broj kombinacija svih grananja izvornog koda
  - Ključna metrika dinamičke analize je stepen pokrivenosti izvornog koda (engl. code coverage)

Dinamička analiza – 2

- Dinamička analiza izvršavanja softvera u kritičnom okruženju treba da se planira pažljivo i ne sme da utiče na izvršavanje osnovnih funkcija ili performanse sistema
  - Često se izvršava u simulacionom okruženju ili na posebnim test sistemima ili replikama sistema
- Dinamička analiza se u zadnje vreme najčešće vrši na Web sadržajima
  - Mnogobrojne (uspešne) kompanije koje su specijalizovane za dinamičku analizu web sadržaja, npr. Veracode, WhiteHat Security

Interaktivna analiza

- **DEF:** Interaktivna analiza omogućava detekciju neželjenih stanja (npr. napad hakera) na osnovu analize ponašanja aplikacije koja se izvršava **preko praćenja zahteva, tokova podataka, korišćenih biblioteka i ostvarenih konekcija**
  - Engleski naziv: Interactive Application Security Testing (IAST)
  - IAST kod se povezuje sa programom koji se analizira i pokreće se zajedno sa njim i ostaje u memoriji
  - Praktična realizacija uglavnom podrazumeva uvezivanje alata za IAST sa procesom preko kog se pruža usluga, npr. sa web serverom ili (Java) virtuelnom mašinom
- **Prednosti** (u odnosu na SAST i DAST): manji broj lažnih pozitivnih detekcija, detekcija u realnom vremenu, smanjena potreba za ekspertima koji analiziraju detektovane anomalije
- **Mane:** primenljiv na „gotovim“ komponentama (u ranim fazama razvoja je otežana primena), potreban DAST alat za kreiranje ulaza i pokrivanje što većeg dela izvornog koda

Bezbedan izvorni kod

ALATI ZA ANALIZU IZVORNOG KODA

FxCop (2018)

- **DEF:** FxCop je Microsoft-ov alat za statičku analizu izvornog koda
  - Prva verzija je izdata sa .NET Framework verzije 1.0 još 2002. godine
  - Moguće njegovo korišćenje kao nezavisnog alata, ili u okviru razvojnog okruženja Visual Studio (Code Analysis)
- Poseduje ugrađena pravila za analizu koda u skladu sa Framework Design Guildelines
  - Pravila podeljena u veći broj grupa
- Umesto originalnog izvornog koda analizira kompajlirani Common Intermediate Language (CIL) kod koji generišu .NET kompajleri
  - Ovo omogućava da alat bude nezavisan od programskog jezika

Security AppScan (2018)

- Proizvođač: IBM
- Funkcionalnost: statička, dinamička i interaktivna analiza
- Podrška za programske jezike: Java, .NET (nema Fortran)
- Ne pruža statičku analizu kao servis, npr. prosleđivanje izvornog koda na IBM server u „oblaku“, proveru koda i slanje izveštaja
- Nema podršku za (dinamičku) analizu aplikacija koje na interfejsima razmenjuju podatke u JSON formatu (2015. god)

- Postoji Trial verzija koja je dostupna na IBM-ovom veb sajtu

HP Fortify (2018)

- **Proizvođač:** HPE
- **Funkcionalnost:** statička, dinamička i interaktivna analiza
- Podrška za programske jezike: 28 programskih jezika (nema Fortran)
- Mobilne platforme: Objective-C, Android, Windows, Blackberry
- Integracija sa IDE, tj. razvojnim okruženjem (npr. Eclipse, Visual Studio)

- Ne postoji probna verzija

Bezbedan izvorni kod

OBFUSKACIJA IZVORNOG KODA

Osnovne definicije

- **DEF:** Obfuskacija je tehnika prikrivanja izvornog koda
  - Mera zaštite izvornog koda koji nije otvoren (npr. osetljiv algoritam)
  - Izvorni kod EXE ili DLL fajlova pisanih na C# jeziku je moguće dobiti dekompajliranjem
- Nivo skrivanja izvornog koda:
  - Otvoren: JavaScript, HTML, Python
  - Virtuelna mašina: Java, C#
  - Binarni fajl: C++, C
- Tehnike obfuskacije:
  - Ubacivanje mrtvog ili prljavog koda (dead code insertion)
  - Preraspoređivanje registara i promenljivih (variable and register reassignment)
  - Promena redosleda funkcija (subroutine reordering)
  - Promena redosleda instrukcija (istruction substitution)
  - Transpozicija koda

Rezime

- Primeri iz prakse
- Nenamerne greške
  - 6 tipova
- Analiza izvornog koda
- Alati za analizu izvornog koda
- Obfuskacija izvornog koda

Hvala na pažnji!