Proqram Təminatı Boşluqlarının Analizi Üsulları

Elşən Bağırov
AMEA İnformasiya Texnologiyaları İnstitutu, Bakı, Azərbaycan
elsenbagirov1995@gmail.com

***Xülasə*— Məqalədə proqram təminatında olan boşluqlar, onların başvermə səbəbləri və təzahürləri araşdırılmışdır. Boşluqların analiz edilməsi üçün istifadə edilən metodlar, testetmə texnologiyaları və proqram alətləri analiz edilmiş və ümumiləşdirilmiş, təklif və tövsiyələr irəli sürülmüşdür.**

Açar sözlər— boşluq, proqram təminatı, fuzzing, boşluqların analizi, testetmə

I. GİRİŞ

Proqram məhsulunun istismarı müddətində mövcud olan boşluqları aşkarlamaq istehsalçının və bədniyyətlinin daim marağındadır. Proqram təminatının təhlükəsizliyi kontekstində boşluqlar dedikdə, konfidensial informasiyanın ələ keçirilməsində və ya proqramın normal işləmə sürətinə təsir edilməsində bədniyyətliyə üstünlük qazandıran səhvlər başa düşülür [1]. Bütün proqram təminatında olan təhlükəsizlik insidentlərinin böyük bir qismi bədniyyətlinin ona məlum olan boşluqları istismar etməsi nəticəsində yaranır. Belə səhvlərin istismarı nəticəsində proqramda qeyri-normal davranış müşahidə edilir. Təhlükəsiz proqram məhsulu dedikdə isə hər hansı bir hücumun baş verməsi anında bu proqramda olan bəzi önləyici funksiyaların işə düşməsi sayəsində həmin hücuma davam gətirən proqram nəzərdə tutulur [2].

Keyfiyyətli proqram məhsulunun təqdim edilməsi üçün proqram təminatı istismara buraxılmazdan öncə testetmə mərhələsinə buraxılır, sifarişçi və ya istifadəçilərin tələblərinə uyğunluğu yoxlanılır və tam işlək vəziyyətə gətirilir [3]. Lakin təhlükəsizliyə tam zəmanət vermək çox zaman mümkün olmur.

Demək olar ki, bütün növ sistemlər və proqram təminatlarında müəyyən tip boşluqlar vardır. Boşluqlar nəticəsində əlyetənliyin pozulması, maliyyə itkisi ilə yanaşı hətta insan tələfatı belə baş verə bilər. Buna görə də proqramda olan boşluqların vaxtında aşkarlanması və aradan qaldırılması, analiz edilib qiymətləndirilməsi sayəsində baş verə biləcək risklərin qarşısını maksimum dərəcədə almaq olar.

Texnologiyaların sürətlə inkişafı nəticəsində əvvəllər məlum olmayan yeni tipli boşluqlar üzə çıxa bilər. Bu səbəbdən sistem və tətbiqi proqramların təhlükəsizliyinin mütəmadi olaraq analiz edilməsi ilə məlum və ya naməlum, qəsdli və ya təsadüfi boşluqların aşkarlanması və aradan qaldırılması olduqca vacib məsələdir.

II. PROQRAM TƏMİNATINDA OLAN BOŞLUQLAR

Proqram təminatında rast gəlinən təhlükəsizlik boşluqları əsasən buferin daşması, etibarsız girişlər, avtorizasiya, autentifikasiya, kriptoqrafik boşluqlar və digər kateqoriyalara aid edilə bilər.

A. Boşluqların yaranma səbəbləri

Proqram təminatında olan boşluqlara səbəb olan səhvlər iki böyük sinfə bölünür:

- istismardan öncə buraxılan səhvlər;
- istismar müddətində buraxılan konfiqurasiya səhvləri.

Gartner-in hesablamalarına görə proqram məhsuluna olan uğurlu hücumların 35%-i proqram təminatınn işlənməsi dövründə buraxılan səhvlərdən qaynaqlanır [4]. İstismardan öncə buraxılan təhlükəsizlik səhvləri iki yerə ayrılır:

- layihələndirmə səhvləri;
- kodlaşdırma səhvləri.

Proqram təminatının funksiyaları düzgün şəkildə layihələndirilmədikdə potensial boşluqlar yarana bilər. Bu tip səhvlər mütləq şəkildə tələblərin analizi, yüksək səviyyəli layihələndirmə və digər mərhələlərdə tapılmalı və aradan qaldırılmalıdır.

Kodlaşdırma səhvləri nəticəsində buferin daşması, eyni zamanda iki və daha çox əməliyyatın icra olunması və autentifikasiya səhvləri, parolların qorunmaması və s. boşluqlar yarana bilər. Layihələndirmə səhvlərindən fərqli olaraq, bu tip boşluqlar proqramın test edilməsi zamanı ixtiyari anda aşkar edilə bilər [4].

MITRE təşkilatı və SANS İnstitutu tərəfindən 25 ən təhlükəli proqramlaşdırma səhvlərinin siyahısı tərtib edilmişdir [5]. Bu səhvlər içində SQL inyeksiyası, əməliyyat sistemi inyeksiyası, klassik bufer daşması, kritik funksiyalar üçün autentifikasiyanın olmaması, avtorizasiyanın olmaması, şifrlənmiş fərdi məlumatların istifadəsi, konfidensial verilənlərin şifrlənməsinin ötürülməsi, yalnış avtorizasiya, nəzarət olunmayan sətir formatı və s. səhvlər yer alır [5].

Konfiqurasiya səhvləri proqram quraşdırılan zaman baş verən səhvlərdir. Gartner uğurlu hücumların təxminən 65%-nin bu tip səhvlərdən qaynaqlandığını müəyyən etmişdir. Bu tip səhvlərin də iki altkateqoriyası vardır [4].

Təhlükəli və lazım olmayan xidmətlər – lazım olan əməliyyatları yerinə yetirərkən tələb olunmayan lakin quraşdırılan bəzi xidmətlərdir. Bu tip boşluqlar çoxfunksiyalı proqram təminatlarında, xüsusilə əməliyyat sistemlərində geniş yayılmışdır.

İnzibatçı girişi səhvləri – istifadəçi hesabları və ya girişin idarə edilməsi prosesləri avtorizasiya edilməmiş istifadəçilərin

DOI: 10.25045/NCSoftEng.2017.68

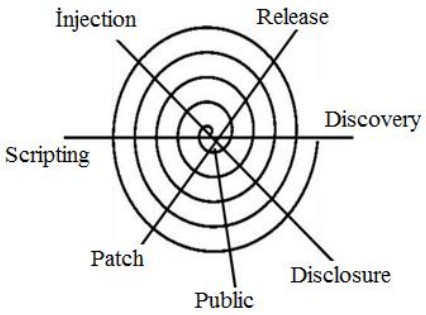
girişi üçün düzgün konfiqurasiya edilməyən səhvlərdir. Bu tip səhvlər ən təhlükəli hesab olunur. Çünki bədniyyətliyə avtorizasiya olunmamış giriş imkanı verir. Nəticədə avtorizasiya olunmamış fəaliyyətlər və ya qanuni istifadəçilərin hüquqlarının pozulması halları müşahidə edilir. Müdaxilələrin aşkarlanması mexanizmləri bu tip boşluqların aşkarlanmasında çox vaxt çətinlik çəkir. Adətən keyfiyyət test edilməsi mərhələsində aşkarlanmalıdır [4].

B. Boşluqların həyat dövrü

Kibertəhlükəsizlik üzrə mütəxəssis Andy Ozment tərəfindən proqram təminatında olan boşluqların həyat dövrü nəzəriyyə olaraq aşağıdakı kimi irəli sürülmüşdür:

- boşluğun yaranma tarixi (injection date);
- boşluq olan proqramın istismara buraxılması tarixi (release date);
- boşluğun ilkin aşkarlanması tarixi (discovery date);
- boşluğun mövcud olması ilə bağlı proqram məhsulu yaradana xəbərdarlıq tarixi (disclosure date);
- boşluğun aradan qaldırılması və mediaya açıqlanması tarixi (public date);
- boşluğun düzəlişinin yayılma tarixi (patch date);
- boşluğun istismarı üçün hazırlanmış ilkin kodun yayımlandığı tarix (scripting date) [6].

Mərhələlər şəkil 1-də iterasiya modeli üzərində əks olunmuşdur.



Şəkil 1. Proqram təminatında olan boşluqların həyat dövrü

III. PROQRAM TƏMİNATININ TƏHLÜKƏSİZLİYİNİ TESTETMƏ TEXNOLOGİYALARI

Proqram məhsulunun təhlükəsizliyinin test edilməsi onun hücumçu kimi boşluqları tapması deməkdir [7]. Proqram təminatında olan boşluqları aşkarlamaq üçün məhsulun növünə görə istifadə olunan testetmə texnologiyaları (ağ qutu, qara qutu, nüfuzetmə və s.) fərqlidir.

Proqram təminatında olan boşluqların istismar müddətində aşkarlanması olduqca böyük xərclər tələb edə bilər. Bu səbəbdən boşluğun erkən müddətdə, yəni, proqram təminatının işlənməsi dövrünün (Software Development Life Cycle, SDLC) mərhələlərində aşkarlanması olduqca əhəmiyyətlidir [8]. Test ardıcıllığı cədvəl 1-də göstərilmişdir.

Qeyd edək ki, proqram məhsulunun keyfiyyətinin təmin edilməsi məqsədilə proqramın işlənmə dövrünün müxtəlif modellərindən istifadə olunmasına baxmayaraq ümumi mərhələlər ardıcıllığı bütün modellərdə gözlənilir [9].

CƏDVƏL 1. PROQRAM TƏMİNATININ İŞLƏNMƏ DÖVRÜNÜN MÜXTƏLİF MƏRHƏLƏLƏRİNDƏ TƏHLÜKƏSIZLIYIN TEST EDILMƏSI

Proqram təminatının işlənməsi mərhələləri	**Təhlükəsizlik testedilməsi mərhələləri**
Tələblərin analizi	Təhlükəsizliyin analizi
Layihələndirmə	Təhlükəsizliyin test edilməsi
Kodlaşdırma və modul testi	Ağ qutu testetməsi
İnteqrasiya testi	Qara qutu testetməsi
Sistem testi	Qara qutu və boşluq testi
Sistemin hazırlıq testi	Nüfuzetmə testi və boşluq testi
Dəstəkləmə	Təsirlərin analizi

Proqram təminatının təhlükəsizlik tələblərinə uyğunluğunun test edilməsi zamanı əsasən altı xassənin təmin edilməsinə fikir verilir:

- konfidensiallıq;
- tamlıq;
- autentifikasiya;
- avtorizasiya;
- əlyetənlik;
- imtinalara qarşı dayanıqlıq [10,11].

A. Ağ qutu testetmə texnologiyası

Ağ qutu testetmə prosesi əsasən proqram təminatının işlənilmə dövründə modullar çoxluğunun birlikdə test edilməsi zamanı icra edilir. Bu, adətən proqram kodunu yazan mühəndis tərəfindən həyata keçirilir. Statik testetmə metodu bu texnologiyanın bir növüdür və burada koda baxış keçirməklə məhsul qiymətləndirilir, leksik, sintaktik və semantik səhvlər tapılır. Kodun icrası zamanı 1-ci növ (false positives) və 2-ci növ səhvlər (false negatives) meydana çıxa bilər. Ona görə də statik testetmə zamanı digər metodlardan da istifadə olunmalıdır [12, 13].

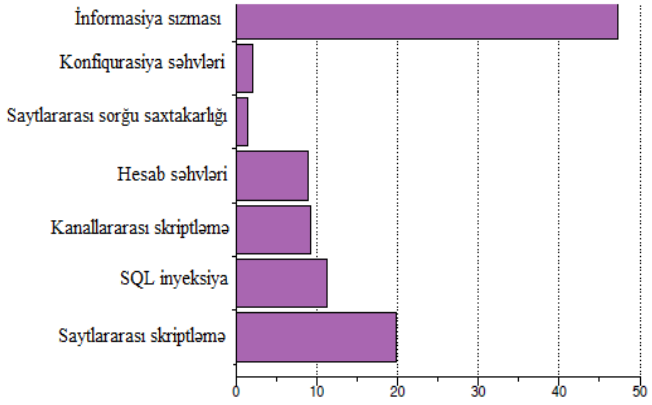
B. Qara qutu testetmə texnologiyası

Qara qutu testetmə texnologiyasının əsas mahiyyəti proqram təminatında olan boşluqları aşkarlamaq üçün məhsulun girişinə dinamik olaraq verilənləri daxil etməklə alınan sorğu cavabını qiymətləndirməkdir. Bura proqramın uzun sətirli mənbə koduna baxış prosesi daxil deyil. Buna görə də testedənin proqramlaşdırma biliyinin olması əhəmiyyət daşımır və təhlükəsizlik üzrə mütəxəssislər tərəfindən aydın başa düşülür. Lakin boşluğun olmasını bildirərkən boşluğun

DOI: 10.25045/NCSoftEng.2017.68

hansı səbəbdən və nə üçün olması məsələsi çox vaxt qaranlıq qalır [14].

Şəkil 3-də McAfee, IBM, HP və Acunetix məşhur dinamik testetmə alətləri istehsalçılarının testetmə məhsulları ilə veb tətbiqlər üzərində aparılmış qara qutu testetmə təcrübəsində hansı tip boşluqların daha çox diqqət çəkdiyi faizlərlə göstərilmişdir [14].



Şəkil 3. Boşluqların paylanma faizləri

Proqram təminatında olan boşluqların aşkarlanması üçün qara qutu texnologiyasının bir növü olan fuzzing metodundan geniş istifadə olunur. Bu metod istifadə olunan proqram alətlərinin müxtəlifliyinə baxmayaraq, aşağıdakı mərhələlərin ardıcıllığı hamısında gözlənilməkdədir:

- hədəfin təyin olunması;
- girişin təyin olunması;
- səhv verilənlərin generasiya edilməsi;
- səhv verilənlərin girişə daxil edilməsi;
- proqram təminatının cavabının müşahidə edilməsi;
- qüsurları qeydə alaraq boşluğun müəyyən edilməsi [7].

Hədəf və giriş istifadəçi tərəfindən təyin edildikdən sonra, fuzzing alətləri boşluqları təyin etmək üçün təsadüfi girişləri və ya mövcud girişləri modifikasiya edərək proqramı icra edir və nəticələri istifadəçiyə bildirir. İstifadəçi həmin boşluğu müəyyən edərək proqram təminatının mənbə kodunda boşluğu aradan qaldırmağa çalışır.

Fuzzing testetmə metodu ilə proqram məhsuluna təsadüfi giriş simvolları sorğusunu daxil edərək proqramın fərqli cavab reaksiyası verməsi nəticəsində onda olan məlum və naməlum boşluqları aşkarlamaq mümkündür. Proqramçıların nəzərindən qaçan və olduqca qarışıq olan, boşluqların aşkarlanmasında səmərəliliyi ilə fərqlənən bu metodun istifadə edilməsi çox sadədir.

Fuzzing metodunun əhəmiyyətli olmasına baxmayaraq, istifadəsi zamanı xeyli vaxt tələb etməsi halı müşahidə edilir. Ümumiyyətlə, testetmə nəticəsində yalnız tipik səhvlərin aşkar olunması əsas problemdir. Eyni zamanda, bu metod daha çox veb əsaslı tətbiqi proqramlar üçün hazırlanmışdır [15].

C. Nüfuzetmə testləri

Nüfuzetmə testləri potensial boşluqlarla əlaqədar olan riskləri qiymətləndirmək üçün bədniyyətlilərin istifadə etdiyi əsl hücumları simulyasiya etməyə əsaslanır. Bu tip təhlükəsizlik testləri proqram təminatında olan boşluqların istismar edilməsi ilə həyata keçirilir və adətən yüksək mütəxəssis bilikləri tələb olunmur. Lakin bir çox təşkilatlar nüfuzetmə testini həyata keçirmək üçün kənardan mütəxəssis dəvət edirlər. Testetmə zamanı mütəxəssis sifarişçi tələblərinin analiz edilməsindən başlayaraq bir neçə mərhələni ardıcıl yerinə yetirir [16].

D. Təhlükəsizliyin model əsasında test edilməsi

Model əsasında testetmə zamanı proqram təminatının strukturu və davranışı əsasında bir model qurulur və testetmə prosesi model üzərində aparılır. Proqram təminatının davranışı giriş və çıxış ilə uzlaşaraq bir neçə müxtəlif diaqramlarla təsvir oluna bilər. Test edilmiş proqram təminatının təsviri dəqiq verilir. Proqram təminatının testetmə modellərində sonlu avtomatlar, unifikasiya edilmiş modelləşdirmə dili (Unified Modeling Language, UML) modeli, Markov zəncirləri və s. geniş şəkildə istifadə olunur [7].

E. Proqram təminatının təhlükəsizlik tələblərinə uyğunluq üzrə sertifikatlaşdırılması

Proqram təminatının işlənməsi dövrünün geniş şəkildə istifadə olunmağa başlandığı dövrlərdə keyfiyyətli proqram məhsulu üçün modellər yaradılırdısa, son bir neçə ildə isə əlavə olaraq təhlükəsizlik tələbi də irəli sürülür. Proqram və sistem təhlükəsizliyinin dəyərləndirilməsi üzrə ilk standart “Narıncı Kitab” adı ilə tanınan Etibarlı Kompüter Sistemlərinin Qiymətləndirilməsi Meyarları (Trusted Computer System Evaluation Criteria, TCSEC) standartı olmuşdur [17].

ISO/IEC 15408 (Ümumi meyarlar) standartında təhlükəsizliyin verifikasiya edilmiş test edilməsi məsələlərinə toxunulmuşdur. Standart üç hissədən ibarətdir (giriş, funskional təhlükəsizlik tələbləri, təhlükəsizliyin təsdiq edilməsi tələbləri) [17, 18]. Üçüncü hissədə təhlükəsizliyin qiymətləndirilməsi 7 səviyyəyə (Evaluation Assurance Levels, EAL) bölünmüşdür:

- EAL 1: Funksional testetmə;
- EAL 2: Struktur testetmə;
- EAL 3: Metodik testetmə və yoxlama;
- EAL 4: Metodik layihələndirmə, testetmə və baxış;
- EAL 5: Yarı-formal layihələndirmə və testetmə;
- EAL6: Yarı-formal verifikasiya edilmiş layihələn-dirmə və testetmə;
- EAL7: Formal verifikasiya edilmiş layihələndirmə və testetmə.

Burada səviyyələr artdıqca testetmə səviyyəsinin tələbi də uyğun olaraq artaraq yüksək səviyyəli layihələndirmə, ilkin kodun analiz edilməsi, boşluq analizləri və digər yoxlamalar həyata keçirilir [17].

DOI: 10.25045/NCSoftEng.2017.68

IV. PROQRAM TƏMİNATI BOŞLUQLARININ ANALİZİ ÜÇÜN PROQRAM ALƏTLƏRİ

Açıq kodlu və kommersiya məqsədli qara qutu testetmə alətləri istehsalçılarının geniş yayıldığına görə düzgün vasitənin seçilməsi problemlidir. Proqram məhsulunu test etmək üçün alət seçərkən əsasən aşağıdakı meyarlara baxılır [19].

- asan istifadə olunma;
- hesabatlılıq;
- doğruluq;
- testetmənin əhatəsi və tamlıq;
- boşluq bazasının tutumu və yeniliyi;
- proqramın qiyməti.

Proqram təminatı boşluqlarıının test edilməsinin açıq kodlu proqram alətlərinə misal olaraq FlowFinder, Splint, FindBugs, Owasp ZAP, Spike, RIPS PHP, VisualCodeGrepper, SPI Dynamics, PMD, SonarQube, Agnitio; kommersiya məqsədli proqram məhsullarına isə Veracode, SAINT, Buro Intruder, bugScout, Codacy, Cxsast kimi proqramları göstərmək olar.

Fuzzerlərə Peach Fuzzer, beSTORM, Zzuf, WebScarab, UniOFuzz, Powerfuzzer kimi proqram vasitələrini misal göstərmək olar.

NƏTİCƏ

Proqram təminatının təhlükəsizlik səviyyəsini yüksəltmə istiqamətində müxtəlif yanaşmalar və boşluqları aşkarlamağa xidmət edən müxtəlif növ proqram məhsullarına baxarkən daha çox veb-tətbiqlərin təhlükəsizliyinə üstünlük verildiyi aydın olur. Lakin veb-tətbiqlərdən başqa digər proqram təminatlarının da (əməliyyat sistemlərinin, xidməti proqramların, proqramlaşdırma dillərinin və s.) təhlükəsizliyi üçün geniş şəkildə tədbirlərin görülməsi vacibdir.

Proqram məhsulunun işlənmə dövrünün ən vacib hissələrindən biri olan boşluq təmizlənməsi strategiyasına baxdığımız texnologiyalarla yanaşı həm də mükafatlandırma proqramlarının da geniş şəkildə daxil edilməsi boşluq aşkarlanmasında koordinasiyanın inkişafına yol açа bilər. O cümlədən, proqram məhsulunun təhlükəsizliyi test edildikdən sonra onun standartdlara cavab verməsini sübut edən sertifikatlaşdırmaların sayı artırılmalıdır. Təhlükəsizlik spesifikasiyalarının formal verifikasiyası üçün qəbul edilmiş İSO/İEC 15408 standartında olan təhlükəsizlik tələblərinin ödənildiyi kütləvi şəkildə yoxlanılmalıdır.

ƏDƏBİYYAT

[1] M. Dowd, J. McDonald, J. Schuh, “The art of software security Assesment: Identfying And Preventing Software Vulnerabilities”. Addison Wesley Professional. 2006, 1200 p.

[2] R. Siciliano, “Software Security Incidents Cost an Average $300,000”, http://www.infosecisland.com/blogview/13075-SoftwareSecurity-Incidents-Cost-an-Average-300000.html, 2011.

[3] W. Du, A. P. Mathur, “Vulnerability testing of software system using fault infection”. Department of Computer Seciences, Purdue University, USA, 1998, 20 p.

[4] J. Pascatore, Taxonomy of Software Vulnerabilities. Gartner Research. 2003, 4 p.

[5] 2011 CWE / SANS Top 25 Most Dangerous Software Errors, http://cwe.mitre.org/top25/

[6] O. Andy, “Vulnerability Discovery & Software Security”, University of Cambridge Computer Security Group & Cambridge Labaratory Computer Security Group & Magdalene College, 2007, 139 p.

[7] M. K. Ehmer, “Different approaches to black box testing technique for finding errors”, Inernational Journal of Software Engineering & Applications, 2011, v.2. No 4. p. 31-40.

[8] What is Security testing: Complete Tutorial, http://www.guru99.com/what-is-security-testing.html

[9] A. Mishra., D. A. Dubey, “Comparative study of different Software Development Life Cycle Models in different scenarios”, International Journal of Advance Research in Computer Science and Management Studies, 2013, v.1. No 5. p. 64-69.

[10] M. Felderer et al. “A Classification for Model-Based Security Testing”, The Third International Conference on Advances in System Testing and Validation Lifecycle, Austuria: 2011, pp. 109-114.

[11] F. Michael et al. “Security testing: A survey”, Advances in computers, 2016, v. 101, pp. 1-43.

[12] J. J. Tevis, J. A. Hamilton, “Methods For The Prevention, Detection And Removal Of Software Security Vulnerabilities”, ACM Southeast Conference. 2004, pp. 197-202.

[13] M. Graff, and V. K. Wyk, Secure Coding: Principles and Practices. O'Reilly and Associates, Sebastopol, CA, 2003, 167 p.

[14] J. Bau et al. “State of the art: automated black-box web application vulnerability testing”, Stanford University, Stanford, CA, 14 p.

[15] S. Bekrar et al. “Finding Software Vulnerabilities by Smart fuzzing”, 2011 Fourth IEEE International Conference on software Testing, Verification and Validation, 2011, p. 427-430.

[16] G. Weidman, Penetration testing: A Hands-On Introduction to Hacking. San Francisco: No Starch Press, 2014, 495 p.

[17] B. Erkut, K. Mehmet, B. Hayretdin, A. Erdem. “Güvenli yazılım geliştirme modelleri ve Ortak Kriterler standartı”, 4. Ulusal yazılım mühendisliği sempozyumu-UYMS’09., 2009, p. 11-17.

[18] A. Ariffuddin. “The common criteria ISO/IEC 15408-The insight, some thoughts, questions and issues”, SANS İnstitute, 2001, 11 p.

[19] C.C. Michael, W. Radosevich, Black box security testing tools, Cigital Inc. , 2007, v. 9, p. 31.

DOI: 10.25045/NCSoftEng.2017.68