Yazılım güvenlik testi: bir sistematik literatür haritalaması

Burcu Yalçıner[1], Sena Sönmez Çiçek[2], Esra Şahin[3], Vahid Garousi[1, 4]

1: Yazılım Mühendisliği Araştırma Grubu (HUSE)
Bilgisayar Mühendisliği Bölümü, Hacettepe Üniversitesi, Ankara, Türkiye
2: TÜBİTAK MAM Enerji Enstitüsü, Gebze, Kocaeli, Türkiye
3: Ulaşım, Güvenlik, Enerji ve Otomasyon Sistemleri, ASELSAN A.Ş., Ankara, Türkiye
4: Maral Yazılım Mühendisliği Danışmanlık ve Ar-Ge Corp., Calgary, Kanada

burcuyalciner@cs.hacettepe.edu.tr, sena.sonmez@tubitak.gov.tr
esrasahin@aselsan.com.tr, vahid.garousi@hacettepe.edu.tr

Özet. Mobil uygulamalar, internet ağı uygulamaları, bilgisayar uygulamaları ve sunucu uygulamaları gibi yazılım sistemlerinin güvenlik testleri oldukça önemli faaliyetlerdir. Yazılım güvenlik testi araştırma alanı olgunlaştığı ve bu alanda yapılan çalışmaların sayısı çok fazla olduğu için, bu çalışma alanının mevcut durumunu sistematik olarak sınıflandırmak önem teşkil etmektedir. Bu çalışmada, yazılım güvenlik testi konusundaki yerleşik bilgiler sistematik haritalama çalışması (İngilizcede: systematic mapping study, SM) ile sınıflandırılmıştır. Çalışma kapsamında üç haritalama sorusu kümesi oluşturulmuş, araştırma için anahtar sözcükler seçilmiş, dâhil etme ve hariç tutma kriterleri tanımlanmış ve bir sınıflandırma şeması sistematik olarak geliştirilmiştir. Seçilen birincil çalışmalar bu şemaya göre haritalandırılmıştır. Yazılım güvenlik testi alanında yayınlanmış son makale havuzumuz 67 makaleden oluşmaktadır. Çalışmamız hala devam etmekte olduğu için makale havuzunun %65-70 kadarı analiz edilmiştir. Yapılan analizlerin sonuçlarının bazıları şu şekildedir: (1) Yazılım güvenlik testi alanında, birincil çalışmalarda sunulan katkıların türleri değişmektedir ve en fazla sunulan iki katkı türü, 35 makale ile metotlar/teknikler katkısı ve 13 makale ile de araç katkısı olmuştur; (2) Makalelerde kullanılan araştırma metotlarını gözlemleyerek, 25 makale tarafından kullanılan doğrulama araştırmasının araştırmacılar arasında en çok tercih edilen yöntem olduğu gözlenmiştir.

Anahtar sözcükler: Yazılım testi; Yazılım güvenlik testi; yazılım güvenlik değerlendirmesi

Software security testing: a systematic literature mapping

Abstract. Security testing of software systems such as mobile applications, web applications, computer applications and server applications has become an essential activity. As the area of software security testing (SST) has matured and the number of studies has increased, systematically categorizing the current state-of-the-art is important. In this paper, we classify the established knowledge in this area through a systematic mapping study. In the scope of the study, three sets of mapping questions are posed, research keywords are selected, inclusion and exclusion criteria is defined, a classification schema is devel-

oped and refined systematically, and the selected primary studies are mapped according to this schema. Our final pool of papers consists of 67 papers. As our work is in progress, 65%-%70 of our final pool of papers has been analyzed. Some of the results of analysis are the following: (1) In the software security testing area, the types of contribution presented in primary studies vary and the top two leading facets are methods/techniques presented in 35 papers and tool presented in 13 papers; and (2) By investigating types of research methods used in papers, we observed that validation research used by 25 papers is the most preferred method among researchers.

Keywords: Software testing; software security testing; software security assessment

1 Giriş

Yazılım testleri maliyeti yüksek olan ve oldukça çaba gerektiren işlemlerdir. 2010 yılında yapılan bir çalışmaya göre [1], test işlemleri için (yazılım, donanım ve servislerin testi dâhil) dünya çapındaki maliyetler 79 milyon Euro değerinde ve bu değerin 2014 yılında 100 milyon Euro'ya ulaşması beklenmekteydi. Diğer yandan Robert Siciliano [2] tarafından 2011 yılında verilen bilgilere göre, 150'den fazla kuruluşun katıldığı çalışmada tekbir uygulamaya ait güvenlik vakalarını iyileştirmek için harcanan para ortalama 300.000 dolar değerindedir. Tek bir uygulama için olan bu maliyet binlerce uygulama düşünüldüğünde çok yüksek rakamlara ulaşacaktır. Bu bilgilerde açıkça göstermektedir ki güvenlik açıklıklarının neden olduğu maliyetler, açıklıkları önceden tespit edip düzeltmek için gerekli maliyetlerden kat kat fazladır. Durum bu şekilde olunca yazılım güvenlik testleri; yazılım güvenlik tehditlerini ve açıklıklarını ortaya çıkarmak, oluşabilecek güvenlik açıklıklarını önceden tahmin edebilmek ve güvenlik açıklıkları nedeniyle ortaya çıkan zararları azaltmak için oldukça önemli hale gelmiştir.

Yazılım güvenliğinin testine verilen önem arttıkça bu alanda yapılmış çok farklı konulara odaklanan çalışmalar da artmış ve bunun sonucunda genel çerçeveyi görme ihtiyacı doğurmuştur. Bu nedenle, bu çalışmada yazılım güvenlik testleri konusunda literatürde yer alan çalışmalar sistematik haritalama yöntemi ile analiz edilmiş ve sınıflandırılmıştır. Yazılım mühendisliğinde popüler olan sistematik haritalama yöntemi, geçmişte birçok makalede kullanmıştır, Örneğin, Vahid ve arkadaşları tarafından yazılım test kod mühendisliği[3], UML tabanlı yazılım performans mühendisliği[4], web uygulamaları testi[5] ve grafiksel kullanıcı ara yüzü testi gibi konularda sistematik haritalama çalışması[6] yapılmıştır.

Çalışmanın bir parçası olarak 4 kategoriden oluşan haritalama soruları ve yazılım güvenlik testleri ile ilgili 190 birincil çalışmadan oluşan bir makale havuzu oluşturulmuştur. Çalışma kapsamına giren birincil çalışmaları ayırt edebilmek için dâhil etme (seçme)–hariç tutma kriterleri belirlenmiştir. Dâhil etme kriterlerini sağlayan 67 çalışma seçilmiştir. Sistematik haritalama çalışmasının devam ediyor olması nedeniyle şu an için analiz edilen 45 tane birincil çalışma, bu sistematik haritalama çalışmasına konu olmuştur.

Çalışma şu şekilde düzenlenmiştir. Bölüm 2'de yazılım güvenlik testleri alanı ile ilgili bilgiler ve ilgili çalışmalar, Bölüm 3'de araştırmanın hedef ve yöntemi, Bölüm 4 ve 5'te araştırma sonuçları ve ileriye yönelik çalışmalar anlatılmıştır.

2 Alan özeti ve ilgili çalışmalar

İncelenen çalışmalar içerisinde bu alanda yapılmış sistematik literatür tarama (SLT) ve sistematik literatür haritalama (SLH) çalışması bulunamaması sebebiyle bu çalışmanın literatürdeki boşluğu doldurması beklenmektedir. Çalışmanın bundan sonraki kısımlarının daha iyi şekilde anlaşılması için Bölüm 2.1'de güvenlik testleri ile ilgili genel bir bilgi verilmiş, Bölüm 2.2 de ise yazılım güvenlik testleri ile ilgili yapılmış ikincil çalışmalar listelenmiştir.

2.1 Yazılım güvenlik testleri

Yazılım güvenliği, herhangi bir saldırı anında sistemin doğru çalışabilmesidir[7]. Yazılım güvenlik testleri ise yazılım ürünlerinin ya da yazılım sistemlerinin güvenli olup olmadığını kontrol etmek, herhangi bir saldırı anında verilerin ve sistem fonksiyonelliğinin korunduğundan emin olmak, olası açıklıkları önceden görmek ve önlem almak için yapılan bir çeşit yazılım testidir. Yapılan güvenlik testlerinin; "gizlilik", "bütünlük", "doğrulama", "yetkilendirme", "erişebilirlik" ve "inkâr edememezlik" olmak üzere 6 temel güvenlik konseptini kapsaması beklenir[8].

Yazılım testleri yapılırken belli başlı aktivitelerin sırayla gerçekleştirilmesi beklenir. Bu aktiviteler, Garousi ve arkadaşları[3] tarafından yapılan çalışmada aşağıdaki gibi sınıflandırılmıştır.

- Test senaryosu tasarımı: Mühendislik hedeflerini, insan tecrübelerine dayalı kriterleri karşılayan test gereksinimlerini ya da test senaryoları oluşturulur.
- Test kodlama ve dokümanlama: Manuel ya da otomatik test kodları yazılır.
- Testi çalıştırma: Test senaryoları test edilen yazılımın (Software Under Test (SUT)) üzerinde yürütülür ve sonuçları kayıt edilir.
- Test değerlendirme: Test sonuçları değerlendirilir.
- Test sonuçlarını raporlama: Test kararları ve hataları geliştiricilere raporlanır.

Test edilecek yazılımın ürününe göre test teknikleri ve bu tekniklerin uygulanma metotları farklılık gösterebilir. Yapılan güvenlik testleri sistemin hem güvenlik fonksiyonelliğini hem de güvenlik açıklık seviyesini görmeyi amaçlar. Bu nedenle yazılımlara mümkün olan her yoldan sızmayı amaçlayan sızma testleri yapılır[9]. Sızma testleri, yazılımın iç işlevlerini, kodlarını test etmek için yapılan beyaz kutu testleri; yazılımın bütününe yönelik işlevlerin doğru yapıldığını emin olmak için yapılan kara kutu testleri ve kara kutu testlerine benzerlik gösteren gri kutu testlerinden oluşur. Bu testler sonucunda görülen açıklıklar raporlanır ve düzeltilmeye çalışılır.

2.2 Yazılım güvenlik testlerindeki ikincil çalışmalar

Yazılım güvenlik testi alanında yapılan 20 adet ikincil çalışma [10-29] raporlanmıştır. İkincil çalışmalar, sistematik literatür tarama (SLT), sistematik literatür haritalama (SLH), genel bir görüş veren normal makale ve literatür taraması (İngilizcede: survey) olmak üzere dört kategoride incelenmiş ve Tablo 1'deki gibi kategorize edilmiştir.

Table 1. - Güvenlik test alanında mevcut ikincil (secondary) çalışmalar

Yılı	Makale ismi	Çalışmanın tipi	Ref.
2002	A Comparison of Publicly Available Tools for Static Intrusion Prevention	Makale	[10]
2003	Why security testing is hard	Makale	[11]
2004	Agents of responsibility in software vulnerability processes	Makale	[12]
2005	A portal for software security	Makale	[13]
2005	Black Box Security Testing Tools	Makale	[14]
2006	Dynamic software security testing	Makale	[15]
2006	Software security and SOA Danger, Will Robinson!	Makale	[16]
2007	Software security assurance: A state-of-art report (sar)	Lit. Taraması	[17]
2007	Software Development and Related Security Issues	Makale	[18]
2008	Taxonomies of Attacks and Vulnerabilities in Computer Systems	Lit. Taraması	[19]
2008	Automating software testing using program analysis	Makale	[20]
2008	Learning from Software Security Testing	Makale	[21]
2009	Automatic Testing of Program Security Vulnerabilities	Makale	[22]
2010	Research on Software Security Testing	Makale	[23]
2010	State of the Art: Automated Black-Box Web Application Vulnerability Testing	Makale	[24]
2011	A Classification for Model-Based Security Testing	Makale	[25]
2012	Software Vulnerability Discovery Techniques: A Survey	Lit. Taraması	[26]
2012	Study: Penetration Testing Model	Lit. Taraması	[27]
2012	SAGE: White-box Fuzzing for Security Testing	Makale	[28]
2012	Security Testing Is Not All The Same: A Reference Taxonomy	Makale	[29]

3 Araştırmanın hedef ve yöntemi

Bu çalışmada, [31]'de anlatılan Amaç-Soru-Ölçüt (Goal-Question-Metric, GQM) yöntemi kullanılarak yazılım güvenlik testi alanı ile alakalı bütün bilimsel çalışmalar toplanarak, bu çalışmaların sistematik haritalandırılması yapılmıştır. Toplanan bütün birincil çalışmalar, çalışmalarda kullanılan araştırma tiplerine, literatüre ne tür bir katkı sağladıklarına, yazılım test alanı ile ilgili özelliklerine ve yazılım güvenlik testi alanı ile ilgili özelliklerine göre incelenerek her birinin gerekli verileri elde edilip sınıflandırılmıştır.

Araştırmanın tipi keşifçi (exploratory) olduğu için amacımız yazılım güvenlik testi alanındaki mevcut durumun ne olduğunu öğrenmek, gelecek çalışmalar için bu alandaki yerleşik bilgi ile ilgili verileri toplayıp sınıflandırmaktır.

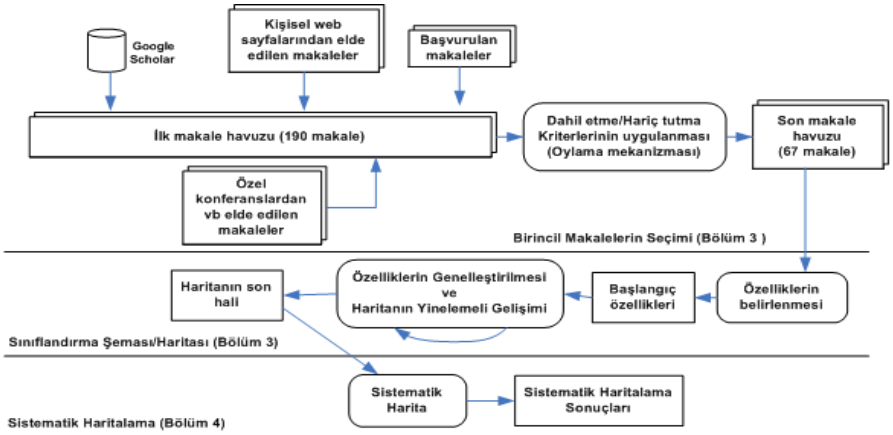
Bu bölümde yazılım güvenlik testi alanında sistematik haritalandırma oluşturmak için yaptığımız araştırma yöntemi hakkında genel bir bilgi verilecektir. Daha sonra, çalışmamızda kullandığımız Amaç-Soru-Ölçüt (GQM) yönteminden bahsedilecek ve son olarak da GQM yöntemi kullanılarak oluşturulan haritalama sorularından (HS, "Mapping Questions") bahsedilecektir.

3.1 Yönteme genel bir bakış

Yapmış olduğumuz sistematik haritalandırma çalışmasında Petersen ve arkadaşları tarafından önerilen metotlar ve süreçler takip edilmiştir [30]. Kullandığımız protokol Şekil 1'de gösterilmektedir. Bu protokol, bilimsel çalışmaların seçilmesi, haritalandırmanın kurulması ve sistematik harita olmak üzere üç bölümden oluşmaktadır.

Sistematik harita çalışmamıza Google Scholar arama motorunu kullanarak yaptığımız makale seçimi adımı ile başladık. Sadece Google Scholar arama motorunu kullanıp, IEEE Xplore, ACM Digital Library, Microsoft Academic Search, CiteSeerX, Science Direct vb. arama motorlarını kullanmamamızın sebebi, Google Scholar kullanarak diğer arama motorları ile bulabileceğimiz makalelere erişmemizin mümkün

olması ve makale havuzumuzu oluştururken aynı makaleleri tekrar havuza dahil etmemektir. Daha sonra kişisel web siteleri, bulduğumuz makalelerin başvurduğu referans makaleler ve özel konferanslardan vb. elde edilen makaleler de araştırılarak makale seçimi adımına devam ettik. İlk makale havuzumuzdaki makalelere uyguladığımız dâhil etme ve hariç tutma kriterlerinin sonucu elde ettiğimiz makalelerden oluşan son makale havuzumuzu oluşturduktan sonra, haritanın kurulması adımında sistematik olarak sınıflandırma şemamızı/haritamızı geliştirdik. Daha sonraki adımlarda, son makale havuzumuzdaki makaleleri, üç haritalama sorusu grubunu kullanarak geliştirdiğimiz Şekil 1'de görülen sistematik sınıflandırma haritamıza yerleştirdik.



Şekil. 1 SLH çalışmasında kullanılan protokol

3.2 Hedef ve haritalamanın soruları

Sistematik haritalandırma çalışmamızın amacı doğrultusunda bu alanda yapılan dört adet bilimsel çalışmadan faydalanarak hedeflerimizi belirledik. Bu çalışmamızın hedefleri; yazılım güvenlik testi alanındaki mevcut durumu anlamak amacıyla bu alanda yapılan bütün bilimsel çalışmaları toplamak, incelemek, sistematik olarak haritalandırmak, gelecekte bu alanda yapılabilecek araştırmaları belirlemek ve istatistiksel sonuçları göstermektir. Daha sonra bu hedeflerimiz doğrultusunda üç adet haritalama sorusu (HS) kümesi oluşturduk. Haritalama sorularımız aşağıdaki gibidir:

- **HS 1:** Bilimsel çalışmaların literatüre sağladığı katkılar nelerdir?
- **HS 2:** Bilimsel çalışmalarda kullanılan araştırma metotlarının tipleri nelerdir?
- **HS 3:** Bilimsel çalışmalarda gösterilen test aktivitelerinin türleri nelerdir?
- **HS 4:** Bilimsel çalışmalarda hangi test senaryosu (test case) üretme metotları kullanılmıştır?
- **HS 5:** Bilimsel çalışmalarda kapsanan güvenlik açıklığı türleri nelerdir?

3.3 Kaynakların (birincil çalışmaların) arama ve seçimi

Yazılım güvenlik testi alanında yapılan birincil çalışmaları arayıp bulmak amacıyla anahtar sözcük tabanlı bir araştırma yapılmıştır. Araştırma için kullanılan anahtar

sözcükler; yazılım güvenlik testi, yazılım güvenlik değerlendirmesi, yazılım saldırıya açıklık testi, yazılım saldırıya açıklık değerlendirmesi, yazılım sızma testi, yazılım sızma değerlendirmesi sözcükleridir. Daha sonra aşağıdaki arama yapısı oluşturulmuştur.

"Software" OR "Program" AND
"Security" OR "Penetration" OR "Vulnerability" AND
"Testing" OR "Assessment" (SOR)

Bu anahtar sözcükler ve arama yapısı, Google Scholar arama motorunda kullanılarak ilk makale havuzumuzu oluşturan 190 tane birincil çalışma bulunmuştur. Bu makaleler sistematik olarak oluşturulan haritalama şemasına göre haritalandırılmıştır. İlk makale havuzundaki makalelerden hangilerini çalışmamıza dâhil edip hangilerini dâhil etmeyeceğimizi belirlemek amacıyla dâhil etme kriterleri(çalışma kapsamımızla alakalı olan makaleler ve çalışmanın doğruluğunu içeren makaleler) ve hariç tutma kriterleri (İngilizce' den farklı dille yazılmış makaleler ve bütününe ulaşamadığımız makaleler) belirlenmiştir. Dâhil etme ve hariç tutma kriterleri aramızda oluşturduğumuz bir oylama sistemi ile uygulanarak son makale havuzumuz 67 makale olarak elde edilmiştir. Herkes tarafından ulaşılabilen, çevrim içi bir depolama alanı olarak oluşturduğumuz haritalama şemamızın linki: https://goo.gl/LJX5QY

3.4 Sistematik haritanın gelişimi ve veri-çıkarma planı

Bu bölümde sistematik haritanın gelişimi ve veri çıkarma planı ile ilgili bilgi verilecek, sistematik haritanın yinelemeli gelişimi anlatılacak, geliştirilen ve çalışmamızda kullanılan sistematik haritamızın son hali Bölüm 5.2'de gösterilecektir.

Seçilen makaleleri sistematik olarak sınıflandırmak amacıyla bir harita geliştirildi ve belirlenen haritalama soruları ve bu sorulardan elde edilen metrikler sistematik haritayı oluşturmak için kullanıldı.

Yazılım güvenlik testi alanındaki bütün birincil çalışmaları kaydetmek amacıyla paylaşımlı bir tablolama programı (çevrim içi Google Docs hesaplama programı) kullanıldı. Her bir birincil çalışma için aşağıdaki bilgiler bu tabloya kaydedildi..

Sistematik harita oluşturulduktan sonra, yüksek kaliteli bir sistematik haritalama çalışması yapmak için yinelemeli bir yöntem izlendi. İlk olarak veri çıkarmak için makalelerin başlıkları, özeti, giriş ve sonuç bölümleri incelendi. Daha sonra bu bölümlerin yetersiz görüldüğü makaleler baştan sona detaylı bir şekilde okunarak gerekli veriler çıkarıldı. Sınıflandırmamızdan emin olmadığımız zaman da diğer araştırmacılara danışıldı ve bu şekilde gerekli veriler elde edildi. Son olarak da, her bir araştırmacı tarafından çıkarılan veriler diğer araştırmacılardan biri tarafından gözden geçirildi.

Tablo 2'de sistematik haritanın son hali gösterilmektedir. Metriklerle ilgili tabloya makaleleri incelerken farklı olarak elde ettiğimiz metrikleri yerleştirmek amacı ile Diğer adlı bir alan eklenmiştir. Bu Diğer alanındaki veriler, yeni bir metrik kategorisi yaratmak için kullanılmaktadır.

Tablo 2. Sistematik Haritanın Son Hali

Haritalama Soruları	Özellikler	Metrikler

HS 1	Makale türleri-Katkı Yönü	Test metotları/teknikleri, test aracı, test modeli, metrik, süreç, vaka çalışması, diğer.
HS 2	Makale türleri-Araştırma Yönü	Çözüm önerisi, Zayıf deneysel çalışma, Güçlü deneysel çalışma, Deneyim çalışması, Felsefi çalışma, Düşünce çalışması, Diğer.
HS 3	Test Aktivitelerinin türleri	Kritere Dayalı Test Senaryosu Tasarımı, Kişi Bilgisine Dayalı Test Senaryosu Tasarımı, Test Otomasyonu, Test Çalıştırma, Test Değerlendirme, Test Planlama ve Yönetimi, Diğer.
HS 4	Test senaryosu üretme metotları	Gereksinime Dayalı Test (Requirement-Based Testing including Model-Based Testing), Mutasyon Testi (Mutation test), Rastgele Test (Fuzzing), Diğer.
HS 5	Güvenlik açıklıklarının türleri	Bellek Taşması (Buffer overflow - BOF), SQL Enjeksiyon (SQLI), Siteler Arası İstek Sahteciliği (Cross-site Request Forgery), Dizi Formatlama Hatası (Format String Bug-FSB), Siteler Arası Betik Yazma (Cross-site Scripting - XSS), Diğer

4 Sonuçlar

4.1 HS 1 - Katkı türü yönünden akademik çalışmalar

Bu soru ile bilgi çıkarımı yapılmış 45 birincil çalışma, katkı türlerine göre sınıflandırılmıştır. Bu katkı türleri belirlenirken Petersen ve arkadaşlarının önerdiği katkı türlerinden yararlanılmıştır. Katkılarına göre bir çalışma birden fazla sınıflandırmaya dâhil olabilmektedir. Örneğin [11]'de yazılım güvenlik açıklıklarını bulmak ve modellemek için hem metot/teknik hem de araç katkısı yapılmıştır. Şekil 2'de katkı türlerine göre akademik çalışma sayıları gösterilmektedir. İncelenen çalışmaların 36'sı (%80) metot / teknik katkısı, 13'ü (%28) araç katkısı, 9'u model, 2'si metrik ve 1'i süreç katkısı yapmıştır. 4 tane vaka çalışması yapılmıştır. Bunun yanı sıra [33]'da mutasyon operatörleri, [23]'de ise dizi biçimlendirme hatalarına (Format String Bugs) yönelik bir eklenti sunulmuştur.

Makale Türü-Katkı Yönleri	**Makale sayısı**	**Referanslar**
Metot /Teknik	36	[2,4,5, 7,9-11,13-16, 18-21,23,24,26-29,32,33,35,36,38-41,44,46-48,50,52,53]
Araç	13	[2,8,13,15,19,20,21,26,34,35,40,48,54]
Model	9	[2,6,14,30,31,38,43,50,54]
Vaka Çalışması	4	[3,17,29,30]
Metrik	2	[19,29]
Diğer	2	[23,33]
Süreç	1	[43]

Şekil. 2. Çalışmaların katkı yönü

4.2 HS 2 – Araştırma türü yönünden akademik çalışmalar

HS 2'yi cevaplamak için birincil çalışmalar şekil 3'te görülen 7 farklı araştırma yöntemine göre, her çalışma sadece bir araştırma yöntemi türüne dâhil olacak şekilde sınıflandırılmıştır Şekil 3'de de görüldüğü üzere 24 çalışma (%50) yeni bir teknik önerip bunu deneysel bir çalışma yaparak doğruladığını gösteren Doğrulama Çalışması (Zayıf Deneysel Çalışma), 14 çalışma çözüm önerisi, 5 çalışma ise haritalama soruları veya hipotezlerle çalışmalarını ortaya koyan Güçlü Deneysel Çalışmadır. Bunun dışında birer tane düşünce çalışması ve filozofik çalışma yer almaktadır.

Makale Türü-Araştırma Yönleri	**Makale sayısı**	**Referanslar**
Doğrulama Çalışması (validation research)	24	[4,5,8,13,15,16,18,20,23,24,27,29,30,32,33,34,39,43,46,47,48,50,52,53]

Çözüm Önerisi (solution proposal)	14	[3,7,9,10,11,14,17,19,21,28,31,36,44,54]
Güçlü Deneysel Çalışma (evaluation research)	5	[2,6,26,35,41]
Felsefi Çalışma	1	[38]
Diğer	1	[40]

Şekil. 3. Çalışmaların araştırma yönü

4.3 HS 3 - Test aktivitesi türleri

Yazılım sistemlerinde güvenlik hataları bulmak amacıyla birçok farklı test aktivitesi yapılmıştır. Şekil 4'de birincil çalışmalarda yapılmış test aktiviteleri gösterilmektedir. En çok yapılan test aktivitesi 31 çalışmada yer alan test çalıştırmadır (Test execution). Bunu 24 çalışma ile test otomasyonu takip etmektedir. 22 çalışmada belirli kriterlere bağlı test senaryosu üretimi, 18 çalışmada ise test değerlendirilmesi (oracle) yapılmıştır. 7 çalışmada kişi bilgisine dayalı test senaryosu üretimi yapılmıştır. Sadece 2 çalışmada test planlama ve yönetimi gerçekleşmiştir. Bunların yanı sıra 4 çalışmada bu kategorilerin dışında farklı test aktiviteleri gerçekleştirilmiştir.

Test Aktiviteleri	**Makale sayısı**	**Referanslar**
Test Çalıştırma	31	[3,4,5,6,8,9,10,13,14-18,20,21,23,24,26,32,33,34,35,41,43,44,47,48,50,52-54]
Test Otomasyonu	24	[2,3,5,6,8,10,11,13,15,17,20,21,24,26,32-,35,38,39,41,43,47,52]
Kritere Dayalı Test Senaryosu Tasarımı	22	[5,6,7,8,10,14,15,16,20,24,26,27,30,31,32,33,38,41,43,50,52,53]
Test Değerlendirme (oracle)	18	[5,6,15,17,20,21,24,26,28,29,32,34,35,36,41,46,48,50]
Kişi Bilgisine Dayalı Test Senaryosu Tasarımı	7	[9,17,18,23,40,47,48]
Diğer	4	[5,19,21,38]
Test Planlama ve Yönetimi	2	[13,27]

Şekil. 4. Yapılan Test Aktiviteleri

4.4 HS 4 – Test senaryosu (test-case) üretme metodu türleri

Bu soru ile test senaryosu üretme metotları belirlenmeye çalışılmıştır. İncelen 45 çalışmada öne çıkan 3 test senaryosu metodu bulunmuştur; Gereksinime Dayalı (Model Bazlı Testi de içerir), Mutasyon Testi ve Rastgele Test. Bunların yanı sıra 15 çalışmada farklı test senaryosu üretme metotlarıyla karşılaşılmıştır. Şekil 5'de test senaryosu üretme metotları ve ilgili sayılar verilmiştir. En çok öne çıkan metot 17 çalışma (%37) ile modele veya gereksinime dayalı test senaryosu üretme yöntemidir. 15 çalışmada ise belirlenen kategoriler dışında test senaryosu üretme yöntemleri kullanılmıştır. Örneğin [7]'de Tehdit Ağacı Dolaşma (Threat Tree Traversal) algoritmasından yararlanan bir teknik ile genel yazılım hatalarına yönelik test senaryoları üretilmiştir.

Test Senaryosu Üretme Metodu	**Makale sayısı**	**Referanslar**
Gereksinime-Modele Dayalı Test	17	[5,9,13,14,19,24,26,27,28,30,31,38,40,41,43,47,52]
Diğer	15	[3,4,5,7,9,11,17,20,23,34,35,39,44,48,54]
Mutasyon Testi	12	[5,6,8,15,21,32,33,41,44,46,50,53]
Rastgele Test (Fuzzing)	9	[2,5,6,9,10,16,18,36,52]

Şekil. 5. Test senaryosu üretme metodu

4.5 HS 5 – Kapsanan güvenlik açıklığı türleri

Yazılım sistemlerinin farklılığına da bağlı olarak, güvenlik açıklıklarına neden birçok farklı türde hatalar bulunmaktadır. Bu haritalama sorusu ile literatürde üzerine çalışılmış açıklık türlerinin neler olduğu ve hangi açıklık üzerinde ne kadar çalışıldığı belirlenmeye çalışılmıştır. İncelen 45 çalışma arasında 13 çalışmada SQL Enjeksiyon açıklığı, 11 çalışmada Bellek Taşması (BOF) açıklığı, 8 çalışmada Siteler Arası Betik Yazma (Cross-site Scripting) açıklığı, 2 çalışmada dizi formatlama hatası (Format String Bug), 1 çalışmada ise Siteler Arası İstek Sahteciliği (Cross-site Request Forgery) açıklığı incelenmiştir. 21 çalışma ise "Diğer" kategorisine dahil edilmiştir. Bu çalışmaların bazılarında herhangi bir açıklık türü belirtilmemiştir ve güvenlik açıklıkları genel olarak ele alınmıştır. Bazılarında ise belirlenen türlere dâhil olmayan farklı açıklıklar ele alınmıştır. Bunlardan bazıları şöyledir; SOAP Enjeksiyonu [38], kimlik denetleme (authentication) açıklıkları [4], genel bulut uygulamaları açıklıkları [52]. Çalışmamızın ilerleyen aşamalarında "Diğer" kategorisinde bulunan çalışmalar incelenip belirli bir sayının üzerinde karşılaşılan açıklıklar yeni bir tür olarak haritaya eklenecektir.

Güvenlik Açıklığı Türleri	Makale sayısı	Referanslar
Diğer	21	[2,3,4,6,7,10,14,18,19,28,30,34,36,38,41,43,44,46,50,52,54]
SQL Enjeksiyon (SQLI)	13	[3,7,8,11,17,20,21,24,31,32,35,39,46]
Bellek Taşması (Buffer overflow - BOF)	11	[2,3,5,7,13,15,16,33,40,47,48]
Siteler Arası Betik Yazma (Cross-site Scripting - XSS)	8	[3,7,8,21,24,41,46,53]
Dizi Formatlama Hatası (Format String Bug-FSB)	2	[23,40]
Siteler Arası İstek Sahteciliği (Cross-site Request Forgery)	1	[3]

Şekil. 6. Güvenlik açıklık türleri

4.6 Geçerliliğe tehditler ve onları ortadan kaldırmak

Bu bölümde, standart bir kontrol listesi temel alınarak [4], çalışmamıza sınır teşkil edebilecek olası geçerliliğe tehditlerden ve bunları nasıl azaltıp, ortadan kaldırmaya çalıştığımızdan bahsedilmektedir. İçsel geçerlilik (internal validity), yapısal geçerlilik (construct validity), sonuç geçerlik (conclusion validity) ve dış geçerlik (external validity) olmak üzere dört tip geçerliliğe tehdit gözlenmiş ve düzeltilmiştir.

İçsel geçerlilik: Bir çalışma ve bu çalışmadan çıkarılan verilere dayalı bir nedensel sonucun garanti edildiğini yansıtan, bilimsel çalışmaların bir özelliğidir. Bu çalışma kapsamında olabildiğince eksiksiz bir birincil çalışma havuzu oluşturulmaya çalışılmış, farklı anahtar sözcükler ve bunların birleşimi ile bir araştırma sözcük kümesi belirlenip ilgili çalışmalar toplanmıştır. Dâhil etme ve hariç tutma kriterleri belirlenmiştir. Araştırmacıların kişisel değerlendirmelerinin etkilerini minimize etmek amacıyla çalışmayı yürüten araştırmacılar ile oylama mekanizması kullanılmıştır.

Yapısal geçerlilik: Çalışmanın nesnelerinin, çalışmanın arkasındaki teoriyi ne ölçüde temsil ettiğiyle ilgilidir. Bölüm 5'te verilen, bir sınıflandırma şemasına göre çalışmalardan bilgi çıkarımları yapılmış ve bu şemaya göre adreslenmiştir. Çalışmalar yazarlar arasında pay edilmiş ve bilgi çıkarımları yapıldıktan sonra farklı yazarlar tarafından yapılan gözden geçirmeler ile farklılıklar ortadan kaldırılmaya çalışılmıştır.

Sonuçsal geçerlik: Bir çalışmadan, titiz ve tekrarlanabilir bir prosedür ile doğru sonuçların elde edilip edilmediği ile ilgilenir. Bölüm 6'da, doğrudan verilerden çıkarılan grafikler ve grafiklerdeki verilerin haritalama tabloları paylaşılmış ve veriler ve sonuçlar arasında izlenebilirlik sağlanmıştır. Buna ek olarak, haritama çalışmamızın verilerine çevrim içi bir depolama alanından erişilebilmektedir.

Dış geçerlilik: Bir çalışmanın sonuçlarının ne ölçüde genelleştirilebileceği ile ilgilenir. Bu sistematik haritama çalışmasının sonuçları yazılım mühendisliği alanındaki yaklaşımlara göre değerlendirilmiştir. Bu nedenle, bu çalışmada sunulan sınıflandırma haritası, sunulan veriler ve belirtilen sonuçlar sadece verilen kapsamda geçerlidir.

5 Sonuç ve gelecek çalışmalar

Yazılım güvenlik testi alanına ilginin giderek artması ve bu kapsamda bu alanda yapılan çalışmaların sayısının günden güne artması sebebiyle yazılım güvenlik testi alanında ikincil bir çalışma olarak sistematik literatür haritalama çalışması yapma ihtiyacı oluşmuştur. Bu kapsamda yaptığımız SLH çalışmamızda, yazılım güvenlik testi alanında yapılmış olan bilimsel çalışmalardan birkaç tanesini inceleyerek, bu alandaki mevcut bilgiyi anlamak ve bu bilgiden yola çıkarak haritalama soruları kurup SLH çalışmamız için gerekli verileri elde etmek üzerine odaklandık. Daha sonra bu alandaki bütün birincil çalışmalar toplanarak, dâhil etme ve hariç tutma kriterlerinin uygulanması sonucu elde edilen son makale havuzumuzdaki 45 tane makalenin verilerini çıkardık. Bu alandaki mevcut bilgiyi çalışmaların literatüre sağladığı katkılar, kullandıkları araştırma yöntemlerinin türleri, kullandıkları test aktivitelerinin türleri, test durumu üretme metotları ve güvenlik açıklıklarının yönünden elde ettiğimiz metrikler sayesinde sınıflandırdık.

Çalışmamız hala devam etmekte olup, makale havuzumuzdaki tüm makalelerin verileri çıkarılarak SLH çalışmamızı en kısa sürede bitirmeyi planlamaktayız. Daha sonra bu alanda bir sistematik literatür değerlendirme (Systematic Literature Review, SLR) çalışması yapılabilir. Ayrıca endüstride mevcut olan ticari yazılım güvenlik test araçlarını, yazılım güvenlik testi alanında yapılan bilimsel çalışmalarda önerilen metotlar/teknikler ile birleştirmek amacıyla bu alanda daha fazla çalışma yapılması gerekmektedir.

Kaynaklar

1. Pierre Audoin Consultants (PAC), "Software testing spends to hit Euro 100bn by 2014," https://www.pac-online.com/software-testing-spends-hit-eur100bn-2014-press-release, 2014.
2. Robert Siciliano, "Software Security Incidents Cost an Average $300,000" http://www.infosecisland.com/blogview/13075-Software-Security-Incidents-Cost-an-Average-300000.html ,2011
3. V. Garousi, Y. Amannejad, A. B. Can, "Software Test-Code Engineering: A Systematic Mapping", Information and Software Technology, Volume 58, 2015, Sayfa 123–147
4. V. Garousi, S. Shahnewaz, and D. Krishnamurthy, "UML-Driven Software Performance Engineering: A Systematic Mapping and Trend Analysis," in Progressions and Innovations in Model-Driven Software Engineering, V. G. Díaz, J. M. C. Lovelle, B. C. P. García-Bustelo, and O. S. Martínez, Eds., ed: IGI Global, 2013.
5. V. Garousi, A. Mesbah, A. Betin-Can, and S. Mirshokraie, "A Systematic Mapping Study of Web Application Testing," Elsevier Journal of Information and Software Technology, vol. 55, pp. 1374–1396, 2013.

6. I. Banerjee, B. Nguyen, V. Garousi, and A. Memon, "Graphical User Interface (GUI) Testing: Systematic Mapping and Repository," Information and Software Technology, vol. 55, pp. 1679–1694, 2013.
7. G. McGraw, B. Potter. " Software Security Testing", Journal IEEE Security and Privacy, Volume 2, Isssue 5, 2004, Sayfa 81-85
8. Security testing tutorial, http://www.tutorialspoint.com/software_testing_dictionary/security_testing.htm
9. OWASP Testing Guide Versiyon 4. https://www.owasp.org/index.php/Testing_Guide_Introduction#Testing_Techniques_Explained
10. J. Wilander, M. Kamkar. "A Comparison of Publicly Available Tools for Static Intrusion Prevention", 7th Nordic Workshop on Secure IT Systems, 2002, Karlstad,
11. H. H. Thompson, "Why Security Testing Is Hard", The IEEE Computer Society, 2003
12. A. Takanen, P. Vuorija¨rvi, M. Laakso, J. Ro¨ning, "Agents of responsibility in software vulnerability processes", Ethics and Information Technology 6: 93–110, 2004.
13. N. R. Mead, G. Mcgraw. "A Portal for Software Security", The IEEE Computer Society, 2005
14. C. C. Michael, W. Radosevich, "Black Box Security Testing Tools", Cigital, Inc. ID: 261, Version: 9, 2005
15. M. R. Stytz, S. B. Banks, "Dynamic Software Security Testing", The IEEE Computer Society, 2006
16. J. Epstein, S. Matsumoto, G. Mcgraw, "Software Security and SOA: Danger, Will Robinson!", The IEEE Computer Society, 2006
17. K. M. Goertzel,T. Winograd, H. L. McKinley, L. J. Oh, M. Colon, T. McGibbon, E. Fedchak, R. Vienneau, "Software Security Assurance", State-of-the-Art Report (SOAR), 2007
18. J. Zadeh, D. DeVolder, "Software Development and Related Security Issues", IEEE, 2007
19. V. M. Igure, R. D. Wılliams,"Taxonomıes Of Attacks And Vulnerabılıties In Computer Systems", IEEE Communıcatıons Surveys, 2008, Volume 10, No. 1
20. P. Godefroid, P. Halleux, A. V. Nori, S. K. Rajamani, W. Schulte, N. Tillmann, M. Y. Levin,"Automating Software Testing Using Program Analysis", The IEEE Computer Society, 2008
21. I. A. Tøndel, M. G. Jaatun, J. Jensen, "Learning from Software Security Testing", IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW'08), 2008
22. H. Shahriar, M. Zulkernine, "Automatic Testing of Program Security Vulnerabilities", 33rd Annual IEEE International Computer Software and Applications Conference ,2009
23. G. Tian-yang, S. Yin-sheng, F. You-yuan, "Research on Software Security Testing", World Academy of Science, Engineering and Technology, 2010
24. J. Bau, E. Bursztein, D. Gupta, J. Mitchell, "State of the Art: Automated Black-Box Web Application Vulnerability Testing", IEEE Symposium on Security and Privacy, 2010
25. M. Felderer, B. Agreiter, P. Zech, R. Breu, "A Classification for Model-Based Security Testing", The Third International Conference on Advances in System Testing and Validation, 2011
26. B. Liu, L. Shi†, Z. Cai, M. Li, "Software Vulnerability Discovery Techniques: A Survey", Fourth International Conference on Multimedia Information Networking and Security,2012
27. "Study: A Penetration Testing Model", Federal Office for Information Security
28. P. Godefroid, M. Y. Levin, D. Molnar, "SAGE: Whitebox Fuzzing for Security Testing"
29. J. Kates, "Securıty Testıng Is Not All The Same: A Reference Taxonomy, Data Securıty Management", Data Securıty Management
30. K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," presented at the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2008.
31. Van Solingen, R., Basili, V., Caldiera, G., & Rombach, H. D. (2002). Goal question metric (gqm) approach. Encyclopedia of software engineering.