Bukalemun: Bilişim Güvenliğinde Yeni Bir Sızma Test Platformu

Emrah Ayyüzlü, Emre Özer

***Abstract* – Penetration tests are considered as one of the most important issues in information security. These tests, when applied in a regular basis, establish a process to ensure the reliability and consistency of information systems. These tests are performed by modeling and mimicking the attacks encountered in the real world, helping the security vulnerabilities of IT systems to be observed more clearly. This paper offers an alternative method to other regular approaches used in the penetration tests. With its dynamic and innovative nature, the penetration testing module - subject to the content of this article - called Chameleon, provides the basis of this alternative approach.**

***Index Terms* — Bukalemun, Chameleon, Sızma Testleri , Penetration tests.**

***Özet* – Sızma testleri, bilişim güvenliğinin en önemli konularından birini oluşturur. Sızma testleri, bilişim sistemlerinin güvenilirliğinin ve tutarlılığının düzenli olarak incelenmesini sağlayan bir süreç tesis eder. Bu testler ile mümkün olduğunca gerçek dünyada karşılaşılabilecek saldırıların modellenerek gerçekleştirilmesi sağlanır. BT (Bilişim Teknolojileri) bünyesindeki güvenlik açıkları daha net bir biçimde gözlenir. Bu bildiri, sızma testlerinde kullanılan yaklaşımlara alternatif bir yöntem sunmaktadır. Bukalemun adı verilen sızma testi modülü dinamik yapısı, kolay kullanılabilirliği, esnekliği ile konusunda yenilikçi bir yaklaşım sunmaktadır.**

I. GİRİŞ

GÜNÜMÜZDE bilişim teknolojileri hızla gelişmekte ve hayatımızda her geçen gün daha çok yer almaktadır. Bu gelişme paralelinde, yeni ortaya çıkan güvenlik açıkları ile bilgiye yetkisiz erişim gibi bazı dezavantajları da beraberinde getirir. Bu durumda bilginin yetkisiz değiştirilmesi, görüntülenmesi, silinmesi ve neticesinde kritik hatalar, hizmet kesintileri yaşanması söz konusu olur. Firmaların organizasyon bütünlüğü ve işleyişi dışında kullanıcıların ve\veya müşterilerin kişisel bilgilerine ulaşılabilir ve bu bilgiler kötü niyetli amaçlar doğrultusunda kullanılabilir.

Sızma testi, kurumların bilişim teknolojileri üzerine inşa ettikleri altyapı, yazılım, donanım ve uygulamalara öngörülen yöntemler kullanarak keşif yapılması ve bu müdahaleler sonucunda sistemsel ve organizasyonel açıkların tespit edilip bu açıkları kullanarak sistemlere sızılmaya çalışılması ve bunun sonucunda oluşabilecek zararların tespit edilip raporlanmasıdır [1].

Sızma testi, kurumların bilişim teknolojileri üzerine inşa ettikleri altyapı, yazılım, donanım ve uygulamalara öngörülen yöntemler kullanarak keşif yapılması ve bu müdahaleler sonucunda sistemsel ve organizasyonel açıkların tespit edilip bu açıkları kullanarak sistemlere sızılmaya çalışılması ve bunun sonucunda oluşabilecek zararların tespit edilip raporlanmasıdır [1].

Sızma testi sonucunda oluşturulan raporlar, sistemlerdeki ve organizasyonun iç süreçlerindeki güvenlik zafiyetlerini detaylandırır. Kurum, bu doğrultuda sistemlerini ve organizasyonunu geliştirir.

II. SIZMA TEST MODÜLLERİNDEKİ TEMEL PROBLEM

Bilişim sektöründe, yeni hizmete giren ve mevcuttaki ürünler ile ilgili yeni güvenlik açıkları tespit edilmektedir [2]. Bulunan açıklar ile ilgili çok farklı atak algoritmaları geliştirilmektedir. Sızma testi farklı adımlardan oluşan bir süreçtir [3]. Sızma testi sürecindeki adımlara destek veren, kendini ispatlamış birçok açık kaynak kodlu, lisanslı uygulama bulunmaktadır [4]. Bu uygulamalardan öne çıkanları amaçları doğrultusunda örnek vermek gerekirse;

a. **İstismar Altyapı Uygulaması**: Metasploit[5]
b. **Flood Uygulaması**: Hping[6]
c. **Port ve Açıklık Tarayıcı Uygulamaları**: Nmap[7], Nessus[8]
d. **Ağ trafiği izleyen uygulamalar:** Wireshark [9]
e. **Uygulama Güvenlik Testi Uygulamaları**: Webinspect[10], Acunetix[11]

Bu uygulamalar sadece belirli sistem veya platformlardaki belirli amaçlara hizmet vermektedir[12]. Bu açıdan, sızma testlerinin güncel güvenlik açıklarını tespit edebiliyor olması elzemdir [13].

Sızma testi modülü geliştiren bir firma herhangi bir güvenlik açığı tespit ettiğinde, açıklığı bulan eklenti, sızma test modülüne hızlı bir şekilde entegre edilebilmelidir. Geleneksel yöntemler ile bu işlem zorlu ve maliyetli bir süreçtir [14]. Yeni bir açık bulunduğunda bu açık ile ilgili öncelikle analiz çalışması yapılması ve ardından bulunan açığı tespit eden eklentinin projelendirilip, geliştirilmesi ve hızlı bir şekilde sızma test modülüne entegre edilmesi gerekmektedir.
Günümüzde iletişim teknolojilerinin hızlı bir şekilde gelişmesi sonucunda birçok gönüllü kullanıcı her gün keşfedilen sistem açıkları hakkında haber almakta ve

Emrah Ayyüzlü Kuveyt Türk Katılım Bankası Teknoloji Mimarı
Emre Özer Kuveyt Türk Katılım Bankası Kurumsal Mimari Müdürü

çalışmalar yapmaktadır. Gönüllü kullanıcılar tarafından tespit edilen bu açıkların sistemler üzerinde var olduğunu doğrulayan uygulama kütüphaneleri geliştirilmekte ve bunlar milyonlarca gönüllü kullanıcı tarafından test edilip en performanslı ve güvenli hale getirilmektedir. Fakat gönüllü kullanıcıların geliştirdikleri uygulamalar dağınık halde bulunmaktadır [15]. Bu makalede önerilen ve bilgi güvenliğinde yeni bir sızma test platformu olarak geliştirilen Bukalemun'un asıl hedefi dağınık halde bulunan bu kod parçacıklarını tek bir çatı altında toplayarak kullanılmasını sağlamaktır.

III. SIZMA TESTİ MODÜLÜ BUKALEMUN

Bukalemun C# üzerine kurulan, sızma testi için kullanılan uygulamaları bir arada kullanmamızı sağlayan, kolay kullanımı hedefleyen, kendine özgü bir programlama (scripting) dili olan bir platformdur. Amacı farklı araçları basit bir önyüzde bir arada kullanmayı sağlamaktır. Sızma testi uygulamaları Bukalemun ortamında tanıtıldıktan sonra artık Bukalemun 'un bir fonksiyonu gibi kullanılmaya başlanır. Bukalemun uzak makinalardaki uygulamalarla farklı bağlantı tipleriyle iletişime geçip, uygulamanın varlığını ve sağlık durumunu kontrol eder. Bukalemun bize üç tip bağlantı tipi sunmaktadır. Bunlar aşağıdaki gibidir:

a. SSH
b. Windows Connection (WMI, Powershell, …)
c. Database

Belirtilen bağlantı tipleri doğabilecek gereksinimler içerisinde genişletilebilir yapıdadır. Bukalemun'da tanımlı bağlantı tiplerini destekleyen ve dışarıdan girdi (input) alabilen bütün uygulamalar kolay bir şekilde Bukalemun script dilinde tanımlanabilmektedir.

Tanımlanan bütün uygulamalar artık platformun bir fonksiyonu haline gelir. Oluşan bu fonksiyonlar, Bukalemun proje tanımlama alanında Bukalemun söz dizinine uygun bir şekilde yazılarak kullanılırlar.

Bukalemun, oluşturulan projeleri çalışma zamanında C# dilindeki karşılıklarına dönüştürerek çalıştırır.

Platformda, iki farklı fonksiyon tipi bulunmaktadır. Bunlar; iç fonksiyonlar ve dış fonksiyonlardır. İç fonksiyonlar platforma ait iken dış fonksiyonlar kullanıcıların Bukalemun'a sonradan tanımladıkları, kullanılmak istenen araçların veya sistemlerin komutlarını ve bağlantılarını içeren fonksiyonlardır. Bu fonksiyonlar kullanılmak istenen uygulamaya ait komutları içerdiği için kullanılan uygulamaya direk bağlıdırlar. Bu yapı, bir dezavantaj gibi görünüp Bukalemun 'un entegre olunan uygulamalara bağımlı bir görüntü çizse de, avantaj olarak aynı işi yapan farklı sistemlerin güçlü yönlerini kullanmamızı sağlamaktadır. Örneğin; Sızma testlerinin yapılabilmesi amacıyla kullanılan birçok uygulama bulunmaktadır. Bu uygulamalardan biri port tarama işini iyi yaparken diğer bir uygulama SQL injection işlemini iyi yapmaktadır. Bukalemun, her iki sistemi orkestra ederek, ana güvenlik sızma orkestrasyon görevini devralır ve kurumlara esneklik kazandırır.

Dış fonksiyonları Bukalemun'a entegre etmek için Bukalemun platformunda yer alan altı adım izlenir. Dış fonksiyonlar yerel veya uzak makinalardaki uygulamaları kullanıyor olabilir. Uzak bir makinada veya yerel bir makinada çalışan araç veya sistem için dış fonksiyon tanımlama aşağıdaki adımlarla gerçekleştirilir.

IV. BAĞLANTI İŞLEMLERİ İÇİN ANAHTARLAR TANIMLAMA

Uzak bir makinadaki bir uygulamanın komutunu çalıştırmamız için, ilk önce belirtilen makinaya bağlanmamız gerekmektedir. Bağlantı işleminin güvenli olması için bir anahtar gerekmektedir. Bukalemun'da anahtar tipi olarak kullanıcı adı ve parola kullanılmaktadır. Bukalemun'da uzak bir makinaya bağlanmak için uzak makinada tanımlı ve erişim yetkisi olan bir kullanıcıya ihtiyaç vardır. Bukalemun'a girilen bir kullanıcı hesabı, platform içindeki farklı uygulamaların bağlantıları için de kullanılabilir.

V. BAĞLANTI TİPLERİNİN BELİRLENMESİ

Bukalemun uzak bir makinaya bağlanıp bir komut çalıştırmak için farklı bağlantı tiplerinden birini kullanabilir. Bu bağlantı tipleri gereksinimler çerçevesinde genişletilip Bukalemun'a iç fonksiyon olarak eklenebilir. Bukalemun şu an için birçok bağlantı tipini desteklemektedir:

a. SSH
b. Database (ODBC, OLE DB)
c. TCP/IP üzerinden NetBIOS
d. PowerShell Remoting

Her tip bağlantı tipinin farklı tanımlaması olabilmektedir. Yalnızca her bağlantı tipi için ortak olan tanımlama kullanılacak olan anahtardır. Anahtar seçimi tanımladığımız anahtarlardan seçilerek yapılır.

VI. KULLANILACAK UYGULAMANIN TANIMLANMASI

Yerel veya Uzak makina üzerinde çalışan uygulama tanımlanır. Bu kısma gelene kadar tanımladığımız anahtar ve bağlantı tipini kapsayan bir isim vermektir. Burada vereceğimiz isim dış fonksiyonlar için bir isim uzayı (namespace) görevi görecektir. Bu yüzden benzersiz olmalıdır.

VII. KOMUT TANIMLANMASI

Kullanılacak uygulamanın bağlantı tipi ve kullanılacak anahtar tanımlandıktan sonra entegre edilen uygulama üzerinde çalıştırılacak komutlar tanımlanır. Her bağlantı tipine göre entegre edilen uygulama üzerinde çalışacak komut farklı olacaktır.

Örnek:

ssh: nmap –sS ip
http/https:
www.google.com.tr/search=word

Entegre edilen uygulama üzerinde çalışacak komutlar, entegre edilen aracın kendisine ait komutlarıdır. Komutlar tanımlandığında Bukalemun için artık bir dış fonksiyon halini alır. Fonksiyonun ismi komut tanımlamasında komuta verilen isimle aynıdır.

Örnek:

Tanımlama:
Anahtar=root
Bağlantı Tipi= SSH
Araç= BackTrack
Komut ismi = PortScan
Komut = nmap –sS ip
Komut Parametreleri =
1. parametre: nmap=>static
2. parametre: -sS => static
3. parametre: İp=>string

Chameleon:
PortScan(ip:"127.0.0.1");
Veya
PortScan(3:"127.0.0.1");

Parametresi statik olan fonksiyon (tanımlanan komut) dışardan değer alamaz. Bir parametrenin dışardan değer alması için Bukalemun'un desteklediği değişken tiplerinden biri olması gerekmektedir. Fonksiyonlara dışarıdan değer girilirken, parametre ile girilen değer aynı tipte olmalıdır.

VIII. STATÜ TANIMLANMASI

Tanımlanan komutların durumlarının (komut çalıştırılmaya başlandığında) nereden kontrol edileceğinin belirlendiği kısımdır. Bir programın çalışma durumu farklı yerlerden kontrol edilebilir.
Aynı program üzerinde oluşturulan komut kalıplarının her birinin durumları farklı yerlerden kontrol edilebilir. Kullanıcı, bu farklı tipteki durum kontrollerini sisteme tanıtabilmelidir.
Kullanıcı bazı komut kalıpları için durum kontrolü kısmını boş da geçebilir. Kullanıcı boş geçtiği dış fonksiyonun kontrolünün yapılmadığı kullanıcıya bildirilir. Kullanıcı isterse bu hatırlatmayı iptal edebilir. (Bunu bir daha gösterme).
Durum kontrolünün yapılabileceği alternatif kaynak ve yöntemler aşağıdaki gibidir:

A. Process

Çalıştırılan bütün işlemlerin process id'leri alınarak işlemlerin çalışıp çalışmadığına, sonucunun başarılı olup olmadığına bakılır.

B. Database

İşlemin durumunu ve başarılı sonuçlanıp sonuçlanmadığını anlamak için veri tabanı tablolarına bakılabilir. Örneğin; işlemin çalışıyor olup olmadığını anlamak için bir tablo üzerinde belirli satır ve sütundaki değerin güncel olup olmadığına bakılabilir. Aynı işlem için işlemin sonlanıp sonlanmadığını belirlemek için belirli bir tabloya kayıt atılıp atılmadığına bakılır.

C. Kütük (Registry)

Microsoft ortamında çalışan programların durumunu kütükten okuyabiliriz. Kullanıcı durumunu okuyacağı programın dizinini (RegKey) ve okuyacağı yerin tipini (DWORD, Text, Binary) girerek durumunu kontrol edilir.

D. XML, W3C, CSV

İşlemlerin durumuna belirli bir dizinin altındaki dosyaya yeni bir veri eklenmesiyle veya belirli bir dizindeki, tag'lar arasındaki verinin güncellenmesiyle bakılabilir. İşlemin durumu, farklı durumlar için aynı dosyada veya farklı dosyalarda farklılık gösterebilir. İşlemin çalışması güncellenmeyle, sonlanması dosyaya veri eklenmesiyle kontrol edilir.
Kullanıcılar n tane durum kontrolü oluşturabilir. Kontrollerin hangisinin kullanacağını seçebilirler.

IX. ÇIKTI TANIMLANMASI

Kullanıcılar kalıplarını oluşturdukları komutların çıktılarının ne tipte olabileceğini, nerelerden okunabileceğini sisteme tanıtabilir. Bir komutun birden fazla çıktısı olabilir. Bu çıktılar ihtiyaca göre arttırılabilir.
Bir komutun çıktısı olmayabilir veya çalışma sırasında çıktı kısmı boş olabilir. Bu tip komutlar sisteme esnek bir biçimde tanıtılabilmektedir. Komutların geri dönüş değeri tipleri aşağıdaki veri tiplerinden biri olabilir:

a. Tam Sayı
b. Ondalıklı Sayı
c. String
d. Void (Geri dönüş değeri yok)
e. Array

Kullanıcı varsayılan olarak yukarıda belirtilen geri dönüş değerini sisteme tanıtabilmektedir. Kullanıcı geri dönüş değeri ile ilgili hiçbir işlem yapmadığında geri dönüş değeri olmadığı varsayılacaktır.
Kullanıcı oluşturduğu komut kalıpları için kendi çıktılarını oluşturup çalışma zamanında istediğini kullanabilir. Kullanıcı şu şekilde bir çıktı oluşturabilir: 'Komut sorunsuz olarak tamamlandıktan sonra işlemin sonucunu belirli bir dizindeki dosyadan oku'
Kullanıcı çıktı tanımları farklı tiplerde olabilir:

A. Database

Komut çıktısı belirli bir veri tabanındaki tablodan okunabilir.

B. XML, W3C, CSV, Text

Kullanıcı belirli bir dizinin altındaki dosyadan çıktıyı okumayı isteyebilir. Burada verinin okunacağı yer dosyanın tipine göre değişebilir. Yazı formatı için dosyanın en altındaki veriyi oku derken, XML için belirli tag'lar arasında bulunan veriyi oku denilebilir.

C. Kütük (Microsoft)

Kullanıcı kütükte okuduğu bilgileri çıktı olarak ayarlayabilir.

Kullanıcı çıktı olarak yukarıda belirtilen seçeneklerle (bu seçenekler ihtiyaca göre artabilir) n tane çıktı oluşturabilir.

X. PROJE TANIMLAMA

Dış fonksiyonlar projeler içerisinde çalıştırılır. Proje kısmı Bukalemun dilinde tanımlamaların yapıldığı ve komutların çalıştırıldığı kısımdır. Proje tanımlarken farklı araçlara uygulamalara dış fonksiyonlar bir arada kullanılabilir.

Proje kısmında Bukalemun script dilinde yazılan kod çalıştırılabilir veya ileriki bir zamanda çalıştırılmak üzere kaydedilebilir.

Örnek:

```
Chamelon Projenin_ismi{
        var @ip String;
        @ip = "127.0.0.1";
        #PortScan(ip:@ip);
}
```

Proje tanımlandıktan sonra kullanıcının yazdığı Bukalemun komut dizinini çalıştırması için ilk olarak projeyi derlemesi (compile) etmesi gerekmektedir. Projenin derlenmesi, Bukalemun'nun söz dizeminin uygun bir şekilde yazılıp yazılmadığını kontrol etmek için yapılmaktadır. Kullanıcı isterse projeyi sağlık kontrolü (HealthCheck) işlemine de tabi tutabilir. Sağlık kontrolü işlemi projede yer alan dış komutların ve ona ait olan diğer birimlerin sağlık durumlarını kontrol eder. Tanımlanan aracın doğruluğuna ve çalışıp çalışmadığına bakar. Belirlenen bağlantı tipi kullanılarak araca erişim yapılabiliyor mu? Tanımlanan anahtar belirtilen makinada tanımlı ve yetkili mi? Bu ve buna benzer kontrolleri yaparak projenin sorunsuz çalışmasına katkıda bulunur.

XI. BUKALEMUN PLATFORMUNUN DİĞER SIZMA TESTİ MODÜLLERİNE GÖRE AVANTAJLARI

Sızma testi gerçekleştiren diğer modüller belirli platform ve uygulamalara yöneliktir. Bukalemun farklı platformlarda farklı şekillerde hizmet veren sızma testi modüllerini veya sızma testinin alt işlemlerini yapan uygulamaları tek bir çatı altında toplayıp etkin bir şekilde yönetmemizi sağlar.

Piyasada var olan sızma testi modüllerinin her biri farklı dallarda kendilerini ispatlamış durumdadır. Bir uygulama, sızma testi adımlarından biri olan port tarama işlemini çok iyi yaparken diğer bir uygulama ise SQL injection işlemini çok iyi yapmaktadır. Fakat bütün sızma test adımlarını, işlemlerin hepsini etkin bir şekilde bir arada sunan ve bütün platformları kapsayan bir modül bulunmamaktadır. Bu modüllerin kullanımı sırasında, kullanılan her bir modülün uzman kişilerine ihtiyaç duyulmaktadır. Bu durumda sızma testi işlemini zor bir süreç haline getirmektedir.

Bukalemun sızma testi işlemlerindeki yukarıda belirtilen probleme esnek bir çözüm sunmaktadır. Bukalemun'a kendi dallarında uzman kişilerin bir kere tanıttığı uygulamalar Bukalemun 'un basit ve kontrollü birer fonksiyonu haline gelir. Bu basit fakat etkin fonksiyonlar, platforma tanıtılan diğer fonksiyonlarla bir arada kullanılarak n tane farklı yapıda sızma testi senaryoları oluşturulmasına imkan sağlar. Bunun yanında, sızma testinin alanında uzman olmayan kişilerce kolay ve kontrollü bir şekilde gerçekleştirilmesi sağlanmaktadır.

XII. SONUÇ

Bilişim güvenliğinde yeni bir sızma test platformu olarak geliştirilen Bukalemun yeni bir yaklaşım getirilmektedir.. Sızma testleri, bilişim sistemlerinin güvenilirliğinin ve tutarlılığının düzenli olarak incelenmesini sağlayan bir süreç tesis eder. Güvenilir ve tutarlı sızma testleri yapmak için kullanılan uygulamaların test edilip kendini ispatlaması gerekmektedir. Bukalemun farklı platformlarda çok farklı uygulamalar için geliştirilmiş açık kaynak kodlu veya lisanlı uygulamaları tek bir çatı altında toplayarak kullanılmasına olanak sağlamaktadır. Farklı platformlarda farklı sistemler için geliştirilmiş uygulamaların bir arada kullanılması, bir kod parçacığından veya uygulamadan alınan çıktıyı bir başka kod parçacığı veya uygulamaya girdi olarak verebilme, her platformu kapsayan eşsiz bir üst uygulama oluşturulmasına olanak sağlamaktadır. Bukalemun'un bu avantajı sayesinde birçok farklı sızma test senaryosu oluşturulup kullanılabilir. Bukalemun'a kendini ispatlamış açık kaynak kodlu uygulamalar kolay bir şekilde entegre edilip kullanılabildiği için düşük maliyetli yüksek performanslı sızma testi yapılmasına olanak sağlar.

XIII. KAYNAKLAR

[1] Kenneth R. van Wyk. (2007) "Adapting Penetration Testing for Software Development Purposes" (2012) https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/penetration/655-BSI.html#dsy655-BSI_fyodor2006

[2] Securtyfocus.com (2012). Vulnerabilities [Online]. http://www.securityfocus.com/vulnerabilities

[3] Nitin A. Naik, Mr. Gajanan D. Kurundkar, Dr. Santosh D. Khamitkar, Namdeo V. Kalyankar
"Penetration Testing: A Roadmap to Network Security" (2012) http://arxiv.org/ftp/arxiv/papers/0912/0912.3970.pdf

[4] Bryan Burns, Dave Killion,Nicolas Beauchesne,Eric Moret,Julien Sobrier,Michael Lynn,Eric Markham,Chris Iezzoni,Philippe Biondi,Jennifer Stisa Granick,Steve Manzuik,Paul Guersch, (2007) "Security Power Tools", 4-27

[5] Abhinav Singh, (2012) "Metasploit Penetration Testing Cookbook" , 7-27

[6] Bryan Burns,Dave Killion,Nicolas Beauchesne,Eric Moret,Julien Sobrier,Michael Lynn,Eric Markham,Chris Iezzoni,Philippe Biondi,Jennifer Stisa Granick,Steve Manzuik,Paul Guersch, (2007)" Security Power Tools",132-137

[7] Bryan Burns,Dave Killion,Nicolas Beauchesne,Eric Moret,Julien Sobrier,Michael Lynn,Eric Markham,Chris Iezzoni,Philippe Biondi,Jennifer Stisa Granick,Steve Manzuik,Paul Guersch, (2007)" Security Power Tools",34-35

[8] Jay Beale,Haroon Meer,Charl van der Walt,Renaud Deraison, (2004)" Nessus Network Auditing: Jay Beale Open Source Security Series" 2-19

[9] Angela Orebaugh,Gilbert Ramirez,Jay Beale, (2007)" Wireshark & Ethereal Network Protocol Analyzer Toolkit (Jay Beale's Open Source Security)"52-65

[10] Bryan Burns,Dave Killion,Nicolas Beauchesne,Eric Moret,Julien Sobrier,Michael Lynn,Eric Markham,Chris Iezzoni,Philippe Biondi,Jennifer Stisa Granick,Steve Manzuik,Paul Guersch, (2007)"Security Power Tools" ,76-85

[11] Ec-Council,Course Technology, (2009)" Web Applications and Data Servers [With Access Code]" ,22-23

[12] Thomas Wilhelm, (2009)"Professional Penetration Testing"

[13] Mehdi Khosrow-Pour, (2006)" Emerging Trends And Challenges in Information Technology Management"

[14] Lee Allen, (2012) "Advanced Penetration Testing for Highly-Secured Environments: The Ultimate"

[15]https://buildsecurityin.us-cert.gov/bsi/articles/tools/penetration/657-BSI.html

Emrah Ayyüzlü: 2011 yılında İstanbul Fatih Üniversitesi Bilgisayar Mühendisliği'nden lisansını aldı. Haliç Üniversitesi Bilgisayar Mühendisliği alanında yüksek lisansına devam etmektedir. Kuveyt Türk Ar-Ge merkezinde Teknoloji Mimarı olarak görevine devam etmektedir.

Emre Özer: 2000 yılında İ.T.Ü Kontrol ve Bilgisayar Mühendisliği'nden lisansını aldı. 2010 yılında MIS alanında yüksek lisansını tamamladı.
12 yıldır profesyonel olarak yazılım sektöründe çeşitli kademelerde görev almıştır. IEEE üyeliği ve çeşitli akademik bildirileri mevcuttur. Kuveyt Türk Ar-Ge merkezin Kurumsal Mimari Müdürü olarak görevine devam etmektedir.