

**TÜBİTAK–2209-A ÜNİVERSİTE ÖĞRENCİLERİ ARAŞTIRMA PROJELERİ DESTEĞİ PROGRAMI**

**ARAŞTIRMA ÖNERİSİ FORMU**

2019

2. Dönem Başvurusu

**A. GENEL BİLGİLER**

|  |
| --- |
| **Başvuru Sahibinin Adı Soyadı: Ahmet Manga** |
| **Araştırma Önerisinin Başlığı: Zararlı Powershell Komutlarının Tespit edilmesi** |
| **Danışmanın Adı Soyadı: Arş. Gör. Seda Kul** |
| **Araştırmanın Yürütüleceği Kurum/Kuruluş: Kocaeli Üniversitesi** |

**ÖZET**

Türkçe özetin araştırma önerisinin (a) özgün değeri, (b) yöntemi, (c) yönetimi ve (d) yaygın etkisi hakkında bilgileri kapsaması beklenir. Her bir özet 450 kelime veya bir sayfa ile sınırlandırılmalıdır. Bu bölümün en son yazılması önerilir.

|  |
| --- |
| **Özet**  Microsoft PowerShell, Windows makinelerde varsayılan olarak yüklü gelen yeni nesil bir komut satırı uygulamasıdır. Programcıların, sistem yöneticilerinin ve yazılım geliştiricilerin .NET framework’üne ve Windows işletim sisteminin sağlamış olduğu özelliklere doğrudan erişip, bu özellikleri kullanabilmeleri için bir ortam sağlar. Sistem yöneticilerinin, rutin olarak yaptıkları dosya silme/oluşturma, çeşitli servisleri başlatıp/durdurma vb. işlemleri MS PowerShell komutları yardımıyla otomatik hale getirebilmeleri gibi birçok yönüyle Windows işletim sistemi ekosisteminde önemli bir yere sahiptir.  Tüm Windows makinelerde yüklü gelmesi, otomatize edilebilmesi, RAM üzerinden direkt çalıştırılabilmesi sayesinde performanslı olması ve çeşitli kod gizleme yöntemleri (obfuscation vb.) kullanılarak siber güvenlik ürünlerinin (antivirus, endpoint detection vb.) atlatılabilmesi gibi nedenlerle siber olaylar sonrası yapılan adli bilişim analizlerinde powershell komutları ile ilgili bir bilgi elde etmek zordur. Bu komutlar, çalıştırıldıktan sonra iz bırakmayacak şekilde rahatlıkla tasarlanabilmektedir.  Bu nedenlerden dolayı MS Powershell, siber tehdit aktörlerine, siber saldırı aşamalarının bir parçası olarak, çoğunlukla internetten zararlı dosya veya komut indirilmesi, bu dosya veya komutun siber güvenlik önlemlerini girişte belirtilen yöntemlerle aşarak gizli şekilde çalıştırması ve sonrasında hiçbir iz bırakmayacak şekilde kullanılması imkanı sağlamaktadır.  Ayrıca powershell kullanımının kolaylığı nedeniyle içerisinde hazır betiklerin (script) bulunduğu siber saldırı sonrası aşamasında saldırganların hedef sistem içerisinde yayılmalarını kolaylaştıracak olan Powersploit ve Powershell Empire gibi araçlar yazılmıştır. Bu araçlar, projenin örnek kötü amaçlı Powershell komutu veri setine dahil edilmiştir. Bu betiklerin tanınması ve altta belirtilen evrimsel sinir ağı (Convolutional Neural Networks - CNN) ve Doğal Dil İşleme (Natural Language Processing - NLP) yöntemleriyle analiz edilmesi amaçlanmıştır. Her bir kötü amaçlı Powershell betiğinden bir sonuç çıkarılarak, projenin ilerleyen safhalarında verilen bir powershell komutuna karşılık zararlı veya temiz olarak sonuç döndüren bir model tasarlanmıştır.  Kötü niyetli PowerShell kullanımının tespit edilmesi için çeşitli yöntemler kullanılmaktadır. Geleneksel-statik bir yöntem olan NLP ve CNN bu yöntemlere örnek olarak verilebilir. NLP yöntemi ile daha statik sonuçlar elde edilirken, CNN yöntemi ile daha çok davranış analizi odaklı çalışmalar yapılması mümkündür. Bu sayede, elde net bir zararlı komut örneği olmadan bir komutun analizinin yapılması mümkündür. NLP yöntemi ile de daha çok benzerlik kuralları odaklı çalışmalar yapılabilmektedir. Bu projede, her iki modelden de azami ölçüde fayda sağlanacaktır.  NLP ile yapılan çalışmalarda odak noktası olarak, tüm bu zararlı powershell komutlarında kullanılan ortak ifadelerin tespiti alınmıştır. Aynı zamanda benzer gizleme yöntemlerinin (obfuscation) tespit edilmesi de bu yöntemin odaklandığı konulardan biridir.  CNN yöntemi kullanılarak yapılan çalışmalarda ise, bir veri seti üzerinden gidilerek sistemin eğitilmesi ve yapılan girdiye karşılık zararlı/temiz gibi bir çıktı üretmesi amaçlanmıştır.  Proje sonucunda ortaya çıkacak olan program bu powershell komutlarını zararlı ve temiz olarak sınıflandıracak, bu sayede güvenlik ürünleri tarafından atlatılmış bir powershell komutu makine öğrenmesi yoluyla tespit edilmiş olacak. Ayrıca bu tespit saldırının en önemli aşamasında, network üzerinde hareket etmeden fark edilmiş olacaktır. |
| **Anahtar Kelimeler:** powershell, Microsoft, zararlı komut |

* **ÖZGÜN DEĞER**

**1.1. Konunun Önemi, Araştırma Önerisinin Özgün Değeri ve Araştırma Sorusu/Hipotezi**

Araştırma önerisinde ele alınan konunun kapsamı ve sınırları ile önemi literatürün eleştirel bir değerlendirmesinin yanı sıra nitel veya nicel verilerle açıklanır.

Özgün değer yazılırken araştırma önerisinin bilimsel değeri, farklılığı ve yeniliği, hangi eksikliği nasıl gidereceği veya hangi soruna nasıl bir çözüm geliştireceği ve/veya ilgili bilim veya teknoloji alan(lar)ına kavramsal, kuramsal ve/veya metodolojik olarak ne gibi özgün katkılarda bulunacağı literatüre atıf yapılarak açıklanır.

Önerilen çalışmanın araştırma sorusu ve varsa hipotezi veya ele aldığı problem(ler)i açık bir şekilde ortaya konulur.

|  |
| --- |
| Önerilen çalışmanın araştırma sorusu Windows istemci veya sunucularında kullanılan powershell komutlarının zararlı veya zararsız olarak analiz edilebilir mi?  Powershell, Microsoft tabanlı yazılımlar için script dili olarak görev yapacak ve yönetim işlemlerini otomatik hale getirebilecek bir ara yüz. Powershell içerisinde çok sayıda nesne mevcuttur. Bunlardan bazıları şöyledir;  Invoke-Command: Genel de uzak sunucuda komut çalıştırmak için kullanılır.  Get-Command: Çalıştırıldığında mevcut komutların listesini verir.  Aşağıda, tehdit aktörleri tarafından neden powershell dosyalarının kullanıldığı açıklanmıştır.  Siber suçluların, PowerShell kullanmalarının ilk 10 nedeni [1]   * Yeni windows sistemlerde varsayılan olarak yüklü gelmektedir. * Hafızadan(memory) direkt olarak çalışabilir. * Varsayılan olarak sistem üzerinde az iz üretir, bu yüzden adli bilişim(forensic) analizlerini zorlaştırır. * Uzaktan varsayılan olarak şifreli trafik ile erişim imkanı bulunmaktadır. * Betik (script) gibi çalışır, kolaylıkla karmaşık hale getirilebilir(obfuscate) ve geleneksel güvenlik araçlarıyla tespit edilmesi zordur. * Sistem yöneticileri, sistemlerinin güvenliklerini kontrol ederken powershell’i sıklıkla gözden kaçırır. * Powershell yapılandırmasına bağlı olarak uygulama-whitelist araçlarını atlatabilir. * Birçok sandbox (deneme ortamı) ortamı script tabanlı zararlı yazılımları iyi analiz edemez. * Kullanılabilir, hazır scriptlerle büyüyen bir topluluğu bulunmaktadır. * Birçok sistem yöneticisi, PowerShell kötü amaçlı yazılımının düzenli yönetim işleriyle karışmasına izin vererek, framework’e güvenir ve kullanır.   Yukarıda belirtilen nedenlerden dolayı Siber suçlular tarafından powershell tercih edilmektedir. Bu yüzden powershell kullanımının analizi önem arz etmektedir.  Genel olarak makine öğrenmede ve özellikle öğrenmede [7] son bilimsel başarılar, etkili siber savunma için yeni teknolojiler geliştirmek için birçok fırsat sunmaktadır. PowerShell betikleri metinsel veri içerdiğinden, Doğal Dil İşleme (NLP) topluluğu içinde geliştirilen çeşitli yöntemleri kullanarak analizlerini düşünmek doğaldır. Duygu analizi için geliştirilen NLP teknikleri [8], kötü niyetli PowerShell olmayan komut dosyalarını [5] tespit etmek için de uygulanabilinir. Bununla birlikte, NLP tekniklerini kötü niyetli komutları saptamak için uyarlamak kolay değildir, çünkü siber saldırganlar kasıtlı olarak komutları kaldırmaktan kaçınmak için komut komutlarını gizlice karıştırırlar [1]. NLP de kullanılmak üzere, karakter düzeyinde derin öğrenme metotlarının başarısı ile motive olarak, geliştirilecek bu projede, bu yaklaşım göz önüne alınarak gelecekte siber suçluların yapacakları girişimleri tespit etmek amaçlanmaktadır.  Zhang ve diğ. [13], metin sınıflandırmada kelime seviyesinden ziyade karakter seviyesinde girişin CNN verildiği bir yaklaşım öne sürmüştür. Derin öğrenme temelli sınıflandırıcılarını, kelime tabanlı geleneksel NLP yöntemleriyle (n-gram gibi) ve Recurrent Neural Networks (RNN LSTM bloklarını kullanarak) ile karşılaştırdılar. Ampirik değerlendirmeleri, duyarlılık analizi ve konu sınıflandırma veri setleri kullanılarak yapıldı. Elde ettikleri sonuçlar, geleneksel yöntemlerin küçük / orta veri kümelerinde daha iyi performans sunarken karakter tabanlı CNN'lerin daha büyük veri kümelerinde daha iyi performans gösterdiğini göstermektedir.  Prusa ve Khoshgoftaar [14], geniş bir tweets veri setinde uygulanan kısa metin duyarlılık analizi sınıflandırması için birkaç mimariyi karşılaştırmaktadır. İki göreceli olarak mimarinin (biri 3 evrimli katman ve 2 tamamen bağlı katman ve diğerinin bir tek LSTM katmanı ve ardından tek bir LSTM katmanı içeren) oluşturduğunu göstermektedir. Bizim çalışmamıza benzer şekilde, Saxe ve Berlin “cleartext” i analiz ederek kötü niyetli yazılım tespiti için derin öğrenme modelleri kullanmışlardır [15]. Daha spesifik olarak, bu modelleri (hem iyi hem de kötü niyetli) URL'lerden, dosya yollarından ve kayıt defteri anahtarlarından oluşan büyük bir veri kümesine uygulamışlardır. Wang ve diğ.[5] web sayfalarından toplanan JavaScript kodunu sınıflandırmak için derin öğrenme modelleri kullanmıştır. Modelleri 8 bitlik bir karakter gösterimi ile karakter düzeyinde kodlama kullanmaktadır. Sınıflandırıcılarını klasik özellik çıkarımına dayalı yöntemlerle karşılaştırmışlardır ve gizli katman sayısının ve boyutlarının algılama doğruluğu üzerindeki etkisini araştırmaktadırlar.  JavaScript [2, 3, 4, 5] gibi kötü amaçlı komut dosyalarını tespit etmek için bazı çalışmalar yapılmış olsada, PowerShell, siber savaştaki önemli statüsüne rağmen, akademik topluluk tarafından göreceli olarak ilgilenmemiştir. PowerShell'deki çalışmaların çoğu, Symantec [1] ve Palo Alto Networks [6] gibi şirketlerdeki güvenlik uzmanları tarafından yapılır. Bu yayınlar, kötü niyetli PowerShell etkinliklerini saptamak için yaklaşımlar geliştirmek ve değerlendirmek yerine, esas olarak PowerShell tehdidini araştırmaya odaklanır. Kötü niyetli PowerShell komutlarının otomatik olarak algılanması konusundaki araştırma eksikliği ile PowerShell tabanlı kötü amaçlı siber faaliyetlerin yaygınlığı arasındaki fark, bu tür saldırıları tespit etmek için etkili yöntemler geliştirme acil ihtiyacını vurgulamaktadır. |

* **Amaç ve Hedefler**

Araştırma önerisinin amacı ve hedefleri açık, ölçülebilir, gerçekçi ve araştırma süresince ulaşılabilir nitelikte olacak şekilde yazılır.

|  |
| --- |
| Araştırma önerisinin amacı herhangi bir Windows sistem üzerinde çalıştırılan powershell komutlarının analiz edilmesidir. Bir powershell komut’u çalıştırıldığında bu komutun tehdit aktörleri tarafından kullanılan bir komut mu aksi takdirde normal bir sistem yöneticisinin mi çalıştırdığı bir komut mu bunun belirlenmesidir.  Proje sonunda hedefimiz çalıştırılan bir powershell komutunun zararlı, veya zararsız olarak belirlenip, çıktı vermesidir. Bu aşama olayların incelenip sonuca bağlanması için çok önemlidir. Çünkü powershell komutu çalıştırıldığı bilgisayarda iz bırakmaz bu nedenle adli bilişim incelemelerinde bulunması zordur. Komut çalıştırıldığında program eğer zararlı bulup alarm verirse olası bir siber saldırı çok fazla yayılmadan tespit edilmiş olacak ve kısa sürede çözümlenecektir. Bu amaç doğrultusunda proje hedefleri:   * Kullanıma açık olarak yayınlanan, Powershell komutlarını içeren veri seti bulunacaktır. * Powershell komutlarının, NLP yöntemleri ile sınıflandırma algoritmalarına sunulabilecek düzeye getirilmesi sağlanacaktır. * Algoritmanın hassasiyet (precision), hatırlama (recall) ve doğruluk (accuracy) metrikleri ile kontrolü gerçekleştirilecektir. |

* **YÖNTEM**

Araştırma önerisinde uygulanacak yöntem ve araştırma teknikleri (veri toplama araçları ve analiz yöntemleri dahil) ilgili literatüre atıf yapılarak açıklanır. Yöntem ve tekniklerin çalışmada öngörülen amaç ve hedeflere ulaşmaya elverişli olduğu ortaya konulur.

Yöntem bölümünün araştırmanın tasarımını, bağımlı ve bağımsız değişkenleri ve istatistiksel yöntemleri kapsaması gerekir. Araştırma önerisinde herhangi bir ön çalışma veya fizibilite yapıldıysa bunların sunulması beklenir. Araştırma önerisinde sunulan yöntemlerin iş paketleri ile ilişkilendirilmesi gerekir.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Proje için yapılan literatür taramasında bu konu ile ilgili akademisyenlerin ve güvenlik firmalarının yaptığı çalışmalar görülmüştür. Bu çalışmalar genel olarak Doğal Dil İşleme ve CNN algoritması etrafında yoğunlaşmıştır. Büyük veri setlerinde CNN algoritması ile başarılı tespitlerin yapıldığı araştırmaların sonuçları içerisinde yer almaktadır [10]. Araştırma sonucunda yöntem belirlerken bu çalışmalar referans alınmıştır. Şekil 1’de çalışmanın akış diyagramı verilmiştir.    Şekil 1: Proje Mimarisi [11]  Program içerisindeki aşamalar şu şekilde olacaktır:  Projede ilk olarak NLP yöntemi uygulanacaktır daha sonra CNN algoritması büyük veri setlerinde daha iyi sonuç vermesi sebebiyle proje içerisinde komut sınıflandırmada kullanılacaktır.   * Powershell kodu içerisinde varsa encoded olanlar onlar decode edilir. * Named Entity Recognition (NER) modelini kullanarak kod içerisindeki URL gibi alanlar bulunur. * Kod içerisindeki özel karakterler, noktalama işaretleri (-.?!) kaldırılır. * (Powershellexe nop noni w hidden exec bypass enc iex newobject netwebclient downloadstring) * Daha sonraki aşamada kodlar kelime gibi birimlere ayırılır. * Stemmer algoritmaları ile kodlardaki morfolojik ekleri atarak kelime kökü bulunur. * Veriler vektör haline getirilir, yani kelimeler tam sayı olarak kodlanır. Bu özellik ile sayısal forma çevrilmiş oldukları için makine öğrenme algoritmaları verileri anlayabilir. * Daha sonra bu kelimeler bir liste haline getirilir ve herhangi bir zararlı komutta bulunan kodlar bu listedeyse kelime vektörüne 1 değilse 0 işlenir. * Machine learning algoritmaları kullanılarak powershell komutu analiz edilir. * Bunun sonucunda powershell komutu için zararlı veya temiz çıkarımları yapılır.   Projenin veri seti ihtiyacı ise ana kaynak olarak github üzerindeki bir veri setinden [13] karşılanacaktır. Bahse konu veri seti içerisinde gerekli etiketlemeler ve ayrıştırmalar yapılacak (zararlı/temiz) ve projeye girdi olarak verilen komutlara karşılık bir sonuç üretilmesi sağlanacaktır.  Yöntem olarak ilk etapta; domain ortamında sunucu ve bilgisayarlarda çalıştırılan powershell komutlarının gün içinde belirli aralıklarla programın çalıştığı sunucuya gönderilmesi gerekir. Daha sonra Doğal Dil İşleme yöntemleriyle metinler içerisinde geçen zararlı powershell komutlarının ortak kelimelerin bulunması gerekir.   |  | | --- | | Bilgisayarı kapatmak için yazılan powershell komutu. | | 1. if ( $HOME[0] -eq "/" ) { 2. # Linux 3. shutdown -P 4. } else { 5. # Windows 6. ( Get-WMIObject *-Query* "Select \* From Win32\_OperatingSystem Where Primary=True" ).ShutDown( ) 7. } |   Programın yanlış alarm (false positive) oranını azaltmamız ve zararlı kodları kaçırmamak için zararlı dosyalarda ortak olarak kullanılan kelimelerin tespit edilmesi önemlidir.  **Zararlı Komutların Kullanım Şekilleri ve Örnekleri**   |  |  | | --- | --- | | Açıklama | Örnek | | Şifrelenmiş komutların kullanılması | "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -Ex BYPasS -noP -W HIddEN -eC IAVwBtAHAAcgBpAHYAZQBsAGUAZwBlAC4AZQBgAZQAdI==” | | Güvenlik ürünlerinin atlatılabilmesi için büyük, küçük harfler veya - . gibi işaretler kullanılması daha sonra bu işaretlerin Replace komutu ile kaldırılması | ... $filename.Replace(’-’,’/’) ... $env:temp + ’:’ + $name + ’.exe ... | | İnternet üzerinden dosya indirilmesi | - Invoke-Expression (("New-Object Net.WebClient")).(’Downloadfile’) ... | | Powershell komutu çalıştırıldığında bir portu dinlemeye geçip bağlantı beklemesi | New-Object System.Net.HttpListener | | Hex olarak verilen powershell komutunun string olarak çevirilip çalıştırılması | powershell.exe -e $(sEt-ITem ‘VARIAble:OFS’) + [sTrInG]( ( 36, ,61 , 32,105 ,1115, 125) | foREAcH { ([Int] $\_-As[char])})+” $(set-Item ‘VaRIABLe:OFs’ ‘ ‘ )”|& ( $ENV:CoMSpec[4,26,25]-JoIN’’) | | Kısaltılmış komutların kullanılması | -nop -w hidden -e <removed> | | ASCII formatındaki zararlı komutların çalıştırılması | sySTem.TExT.ENcodInG]::asCii | | Registry üzerinde kritik değerlerin okunması, eklenmesi, silinmesi | Get-Item -path HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run |   **Şekil 2: Örnek PowerShell komutları** |

* **PROJE YÖNETİMİ**
* **İş- Zaman Çizelgesi**

Araştırma önerisinde yer alacak başlıca iş paketleri ve hedefleri, her bir iş paketinin hangi sürede gerçekleştirileceği, başarı ölçütü ve araştırmanın başarısına katkısı “İş-Zaman Çizelgesi” doldurularak verilir. Literatür taraması, gelişme ve sonuç raporu hazırlama aşamaları, araştırma sonuçlarının paylaşımı, makale yazımı ve malzeme alımı ayrı birer iş paketi olarak gösterilmemelidir.

Başarı ölçütü olarak her bir iş paketinin hangi kriterleri sağladığında başarılı sayılacağı açıklanır. Başarı ölçütü, ölçülebilir ve izlenebilir nitelikte olacak şekilde nicel veya nitel ölçütlerle (ifade, sayı, yüzde, vb.) belirtilir.

**İŞ-ZAMAN ÇİZELGESİ (\*)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **İP No** | **İş Paketlerinin Adı ve Hedefleri** | **Kim(ler) Tarafından Gerçekleştirileceği** | **Zaman Aralığı**  **(..-.. Ay)** | **Başarı Ölçütü ve Projenin Başarısına Katkısı** |
| 1 | Proje kapsamının belirlenmesi | Ahmet Manga  Zeki Esenalp  Rıdvan Kaplan | 2 Hafta | Projenin kapsamının doğru belirlenmesi, çıktılarının kalitesini doğrudan etkilemektedir. |
| 2 | Zararlı/şüpheli komutların tespit edilip kategorize edilmesi | Ahmet Manga  Rıdvan Kaplan | 1 Ay | Projenin en önemli ve sonucu etkileyecek aşaması, komutların sınıflandırılmasıdır. |
| 3 | Altyapı için kullanılacak sistemlerin tespiti | Zeki Esenalp | 2 Hafta | Projenin kullanıcı arayüzü, sunucu sistemleri gibi altyapı alanlarında kullanılacak teknolojilerin tespiti. |
| 4 | Istemci ve sunuculara yazılacak program | Ahmet Manga  Zeki Esenalp | 1.5 Ay | Rahat bir kullanıcı deneyimi oluşturmak için gereken tasarım ve UI komponentlerinin oluşturulması. |
| 5 | Sunucuda çalışacak programın ilk aşamasının yazılması; yazacağımız kuralların program tarafından tanınıp bunlara göre komut içinde arama yapılması | Ahmet Manga  Rıdvan Kaplan  Zeki Esenalp | 1 Ay | Kullanıcı tarafından yüklenen içeriklerin, belirlenen kurallara göre parse edilmesi ve sınıflandırma aşamasına geçilmesi. |
| 6 | Sunucuda çalışacak programın tamamlanması | Ahmet Manga  Rıdvan Kaplan  Zeki Esenalp | 1 Ay | Son kullanıcıya yönelik tüm hizmetlerin algoritma ve kodlama işlemlerinin tamamlanması |
| 7 | Programın testlerinin yapılması | Rıdvan Kaplan  Zeki Esenalp | 2 Hafta | Program testlerinin yapılması, uygun parametre kontrollerinin yapılması vb. |

(\*) Çizelgedeki satırlar ve sütunlar gerektiği kadar genişletilebilir ve çoğaltılabilir.

* **Risk Yönetimi**

Araştırmanın başarısını olumsuz yönde etkileyebilecek riskler ve bu risklerle karşılaşıldığında araştırmanın başarıyla yürütülmesini sağlamak için alınacak tedbirler (B Planı) ilgili iş paketleri belirtilerek ana hatlarıyla aşağıdaki Risk Yönetimi Tablosu’nda ifade edilir. B planlarının uygulanması araştırmanın temel hedeflerinden sapmaya yol açmamalıdır.

**RİSK YÖNETİMİ TABLOSU\***

|  |  |  |
| --- | --- | --- |
| **İP No** | **En Önemli Riskler** | **Risk Yönetimi (B Planı)** |
| 1 | Programın test aşamasında çok fazla komutu zararlı olarak tanımlaması. | Veri seti değiştirilerek daha doğru sonuç alınması amaçlanır. |
| 2 | Powershell komutlarının sunucuya atılamaması | Windows üzerinde başka programlar ile log yönlendirmeyi denenir. |

(\*) Tablodaki satırlar gerektiği kadar genişletilebilir ve çoğaltılabilir.

* **Araştırma Olanakları**

Bu bölümde projenin yürütüleceği kurum ve kuruluşlardavar olan ve projede kullanılacak olan altyapı/ekipman (laboratuvar, araç, makine-teçhizat, vb.)olanakları belirtilir.

**ARAŞTIRMA OLANAKLARI TABLOSU (\*)**

|  |  |
| --- | --- |
| **Kuruluşta Bulunan Altyapı/Ekipman Türü, Modeli**  (Laboratuvar, Araç, Makine-Teçhizat, vb.) | **Projede Kullanım Amacı** |
| Dizüstü bilgisayar | Proje ekibinin ortak çalışmalarında kullanılmaktadır. |
| Sunucu hizmeti | Bulut sunucu platformlarından alınan hizmetler. |
|  |  |

**(\*)** Tablodaki satırlar gerektiği kadar genişletilebilir ve çoğaltılabilir.

* **YAYGIN ETKİ**

Önerilen çalışma başarıyla gerçekleştirildiği takdirde araştırmadan elde edilmesi öngörülen ve beklenen yaygın etkilerin neler olabileceği, diğer bir ifadeyle yapılan araştırmadan ne gibi çıktı, sonuç ve etkilerin elde edileceği aşağıdaki tabloda verilir.

**ARAŞTIRMA ÖNERİSİNDEN BEKLENEN YAYGIN ETKİ TABLOSU**

|  |  |
| --- | --- |
| **Yaygın Etki Türleri** | **Önerilen Araştırmadan Beklenen Çıktı, Sonuç ve Etkiler** |
| **Bilimsel/Akademik**  (Makale, Bildiri, Kitap Bölümü, Kitap) | Powershell komutlarının, Derin Öğrenme Tabanlı metotları kullanılarak sınıflandırılması üzerine bildiri yayınlanması hedeflenmektedir. |
| **Ekonomik/Ticari/Sosyal**  (Ürün, Prototip, Patent, Faydalı Model, Üretim İzni, Çeşit Tescili, Spin-off/Start- up Şirket, Görsel/İşitsel Arşiv, Envanter/Veri Tabanı/Belgeleme Üretimi, Telife Konu Olan Eser, Medyada Yer Alma, Fuar, Proje Pazarı, Çalıştay, Eğitim vb. Bilimsel Etkinlik, Proje Sonuçlarını Kullanacak Kurum/Kuruluş, vb. diğer yaygın etkiler) | Projemiz sonucunda, sunucu/istemci mimarisi kullanılan bir güvenlik ürünü ortaya çıkarmayı planlıyoruz. Bu ürün ile, kullanıcıların sistemlerinde çalışan powershell dosyalarının merkezi bir sistem ile analizi yapılarak, sonuçlarının ilgili siber güvenlik birimlerine ulaştırılmasını hedeflenmektedir. |
| **Araştırmacı Yetiştirilmesi ve Yeni Proje(ler) Oluşturma**  (Yüksek Lisans/Doktora Tezi, Ulusal/Uluslararası Yeni Proje) | Powershell ve javascript komutlarının beraber sınıflandırılması üzerine yeni bir proje çalışması yürütülebilinir. |

* **BELİRTMEK İSTEDİĞİNİZ DİĞER KONULAR**

Sadece araştırma önerisinin değerlendirilmesine katkı sağlayabilecek bilgi/veri (grafik, tablo, vb.) eklenebilir.

|  |
| --- |
|  |

* **EKLER**

**EK-1: KAYNAKLAR**

1. Symantec. The increased use of Powershell in attacks. <https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/increased-use-of-powershell-in-attacks-16-en.pdf> , 2016.
2. Marco Cova, Christopher Kruegel, and Giovanni Vigna. Detection and analysis of drive-bydownload attacks and malicious javascript code. In Proceedings of the 19th international conference on World wide web, pages 281–290. ACM, 2010.
3. Charlie Curtsinger, Benjamin Livshits, Benjamin G Zorn, and Christian Seifert. Zozzle: Fast and precise in-browser javascript malware detection. In USENIX Security Symposium, pages 33–48. USENIX Association, 2011.
4. Peter Likarish, Eunjin Jung, and Insoon Jo. Obfuscated malicious javascript detection using classification techniques. In Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on, pages 47–54. IEEE, 2009.
5. Yao Wang, Wan-dong Cai, and Peng-cheng Wei. A deep learning approach for detecting malicious javascript code. Security and Communication Networks, 9(11):1520–1534, 2016.
6. PaloAlto. Pulling Back the Curtains on EncodedCommand PowerShell Attacks. https://researchcenter. paloaltonetworks.com/2017/03/ unit42-pulling-back-the-curtains-on-encodedcommand-powershell-attacks/, 2017.
7. Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. Deep Learning. Adaptive computation and machine learning. MIT Press, 2016.
8. Bing Liu and Lei Zhang. A survey of opinion mining and sentiment analysis. In Mining text data, pages 415–463. Springer, 2012.
9. <https://towardsdatascience.com/natural-language-processing-nlp-for-machine-learning-d44498845d5b> (28-10-2019)
10. <https://www.fireeye.com/blog/threat-research/2018/07/malicious-powershell-detection-via-machine-learning.html> (28-10-2019)
11. Danny Hendler, Shay Kels, and Amir Rubin. 2018. Detecting Malicious PowerShell Commands using Deep Neural Networks. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS '18). ACM, New York, NY, USA, 187-197. DOI: https://doi.org/10.1145/3196494.3196511.
12. <https://github.com/danielbohannon/Revoke-Obfuscation/blob/master/README.md#purpose> (28-10-2019)
13. Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In Advances in neural information processing systems, pages 649–657. NIPS, 2015.
14. Joseph D. Prusa and Taghi M. Khoshgoftaar. Deep neural network architecture for characterlevel learning on short text. In Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2017, Marco Island, Florida, USA, May 22-24, 2017., pages 353–358. AAAI Press, 2017.
15. Joshua Saxe and Konstantin Berlin. expose: A character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys. arXiv preprint arXiv:1702.08568, 2017.