شکار وب شل

ریاست جمهوری
مرکز مدیریت راهبردی افتا
طلیع جبهه فارس

آکادمی راوین

فهرست

۳ **بخش نخست:** < پیش‌گفتار >

۵ وب‌شل‌ها چگونه کار می‌کنند و استفاده می‌شوند؟

۸ **بخش دوم:** < شناسایی وب‌شل >

۹ مقایسه‌ی فایل‌ها با نسخه‌ی شناخته شده و معتبر

۱۰ اسکریپت PowerShell برای سرورهای ویندوز

۱۱ ابزار WinDiff

۱۲ دستورات قابل استفاده از سرورهای وب لینوکسی

۱۳ شناسایی وب‌شل‌ها با استفاده از ناهنجاری‌های ترافیک شبکه

۱۴ رول‌های جستجوی Splunk با هدف شناسایی URIهای غیرعادی در ترافیک وب

۱۸ نمونه اسکریپت‌های شناسایی وب‌شل با استفاده از تحلیل درخواست‌های غیر عادی

۲۵ کشف وب‌شل‌های شناخته شده

۲۶ مجموعه رول‌های Snort با هدف شناسایی وب‌شل‌های پرکاربرد

۲۸ ابزار LOKI

۲۹ سایر ابزارها

۳۰ شناسایی با استفاده از تحلیل‌های آماری

۳۱ جریان‌های شبکه‌ی (Network Flows) غیر منتظره

۳۲ شناسایی جریان غیرمنتظره‌ی شبکه

۳۳ شناسایی با استفاده از محصولات EDR

۳۴ شناسایی پروسس‌های غیر عادی در داده‌های Sysmon

۳۵ شناسایی پروسس‌های غیر معمول لینوکس با Auditd

۳۸ **بخش سوم:** < پاسخ به تهدید وب‌شل و بازیابی امنیت >

۳۹ **جمع‌بندی**

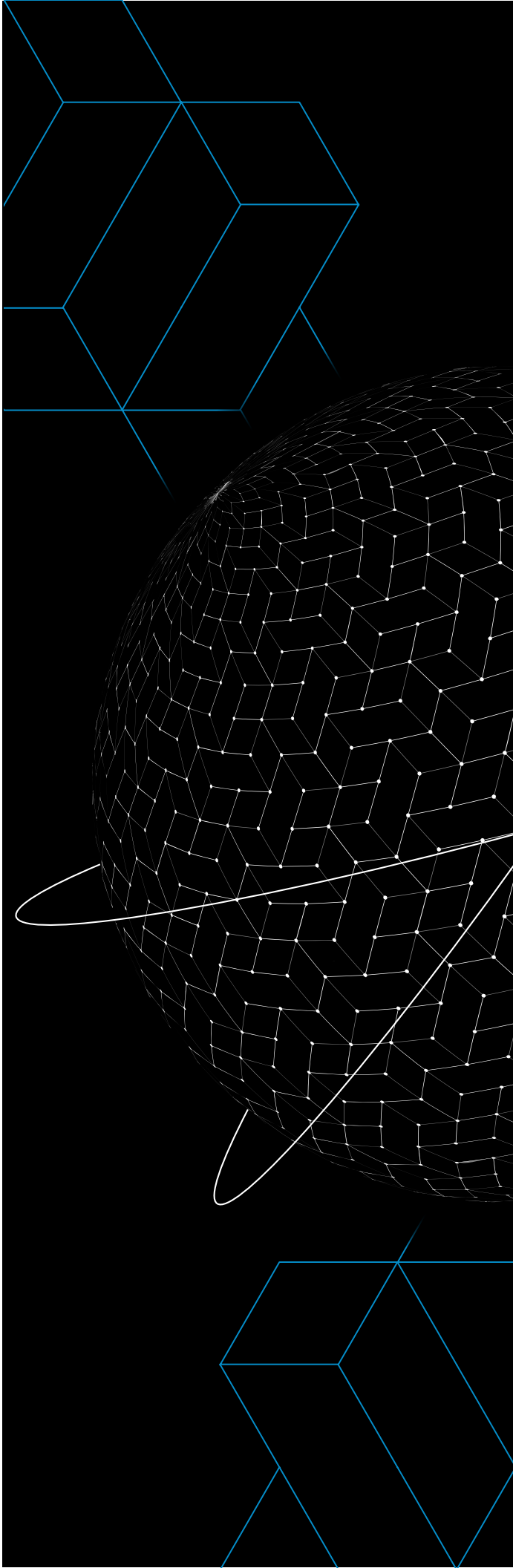
۴۰ **منابع**

۱ < پیش‌گفتار>

وب‌شل‌ها در حقیقت برنامه‌های مبتنی بر وب هستند که امکان تعامل مهاجمین با سامانه یا سیستم‌عامل سرور وب را ممکن می‌سازند. این تعامل شامل دسترسی به فایل‌ها، آپلود فایل، اجرای یک کد دل‌خواه بر روی سرور آسیب‌پذیر و بسیاری موارد دیگر می‌شود. این ابزارها بسته به نوع سرور وب آسیب‌پذیر، با زبان‌های متفاوتی مانند PHP، ASP، Java، Javascript و غیره نوشته می‌شوند که رایج‌ترین آن‌ها PHP هست چرا که بسیاری از سامانه‌های تحت وب دنیا مبتنی بر PHP هستند.

زمانی که یک وب‌شل در سامانه‌ی شما قرار می‌گیرد، مهاجم می‌تواند با استفاده از آن داده‌های شما را سرقت کند، به مهم‌ترین سرورهای داخل شبکه دسترسی پیدا کند و یا با هدف گسترش حمله، بدافزارهای گسترده‌تر و خطرناک‌تری را بر روی سامانه بارگذاری کند.

براساس فریم‌ورک MITRE ATT&CK، وب‌شل می‌تواند با هدف گسترش دسترسی یا پایدارسازی دسترسی بر روی سرورهای وب سازمان قربانی نصب شود. به عبارت دیگر ممکن است مهاجم با استفاده از آسیب‌پذیری موجود در سرور وب قربانی، وب‌شل خود را بر روی آن قرار دهد تا بتواند دستورات و کدهای مورد نظر را با استفاده از آن بر روی سرور اجرا کند. از طرف دیگر ممکن است مهاجمی که



موفق شده از روش دیگری به زیرساخت سازمان قربانی نفوذ کند برای پایدارتر کردن دسترسی خود، بر روی سرورهای وب سازمان وب‌شل را به عنوان Backdoor نصب کند. به این ترتیب مهاجم در صورت لو رفتن و از دست دادن دسترسی خود، می‌تواند مجدد از مسیر وب‌شل موجود در سرورهای وب که از اینترنت قابل دسترسی هستند به زیرساخت قربانی بازگردد.

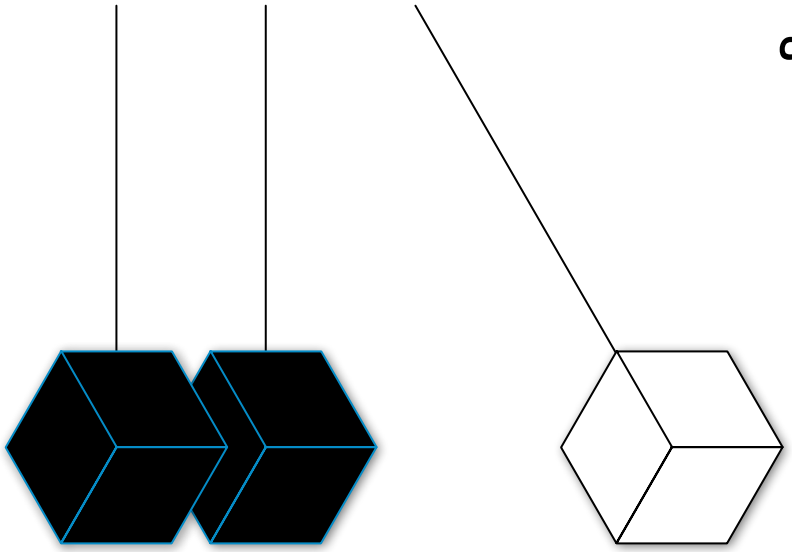
ممکن است این سوال برای شما مطرح شود که آیا ما هم نیاز داریم تا به مساله‌ی وجود وب‌شل در سازمان خود اهمیت بدهیم؟

همه‌ی سازمان‌ها و کسب‌وکارهایی که حتی یک سامانه‌ی تحت وب آنلاین دارند، در معرض تهدید وب‌شل هستند. آن‌ها اغلب بر روی وب‌سایت‌های متعلق به کسب‌وکارهای کوچک هم مشاهده می‌شوند، به خصوص سایت‌هایی که با استفاده از سامانه‌های عمومی مانند وردپرس ایجادشده‌اند. چرا که افزونه‌ها و تم‌های این سامانه‌ها به دلیل پیدا شدن آسیب‌پذیری‌های مداوم، از محبوبیت بالایی در میان نفوذگران برخوردار هستند.

البته وردپرس تنها یک مثال است و در حقیقت تمام برنامه‌های مبتنی بر وب گاهی با آسیب‌پذیری‌های جدید کشف شده روبرو می‌شوند که می‌تواند قابلیت نصب وب‌شل بر روی سامانه‌ی آسیب‌پذیر را به همراه داشته باشد. به عنوان مثال اگر شما وب‌سایتی دارید که اطلاعات کاربران را ذخیره می‌کند یا به آن‌ها سرویس‌های مختلفی را ارایه می‌دهد، زمان مناسبی است که راجع به وب‌شل‌ها بیشتر یاد بگیرید چرا که ممکن است در برابر آن‌ها آسیب‌پذیر باشید.

به این نکته هم دقت داشته باشید که وب‌شل‌ها تنها با استفاده از آسیب‌پذیری‌های موجود در سامانه بر روی سرور وب نصب نمی‌شوند، بلکه همان‌گونه که ذکر شد ممکن است مهاجمین از قبل به زیرساخت دسترسی داشته و وب‌شل را از داخل سازمان بر روی سرور وب نصب کنند. لذا آسیب‌پذیر نبودن سامانه به معنی عدم وجود وب‌شل بر روی آن نیست.

در این کتابچه تلاش شده است تا استفاده از روش‌های متنوع با هدف شناسایی انواع وب‌شل‌های شناخته شده و شناخته نشده به مخاطبین آموزش داده شود. هم‌چنین ابزارها، رول‌های شناسایی و اسکریپت‌های مختلفی برای این منظور در بخش‌های مختلف ارایه شده است.



۱.۱ <وب‌شل‌ها چگونه کار می‌کنند و استفاده می‌شوند؟>

اولین گام یک مهاجم برای کار با وب‌شل این است که به یکی از ده‌ها روش ممکن، آن را بر روی یک سرور آسیب‌پذیر بارگذاری کند به گونه‌ای که بتواند از اینترنت یا شبکه به آن دسترسی داشته باشد. این بارگذاری غیر مجاز وب‌شل به روش‌های متفاوتی ممکن است انجام شود اما رایج‌ترین تکنیک‌های مورد استفاده‌ی مهاجمین عبارتند از:

- اکسپلویت آسیب‌پذیری موجود در یکی از بخش‌های سامانه‌ی تحت وب یا سیستم‌عامل آن
- دسترسی به پرتال مدیریت
- سو استفاده از پیکربندی نامناسب یک میزبان (Host)
- دسترسی به سرور وب از داخل سازمان

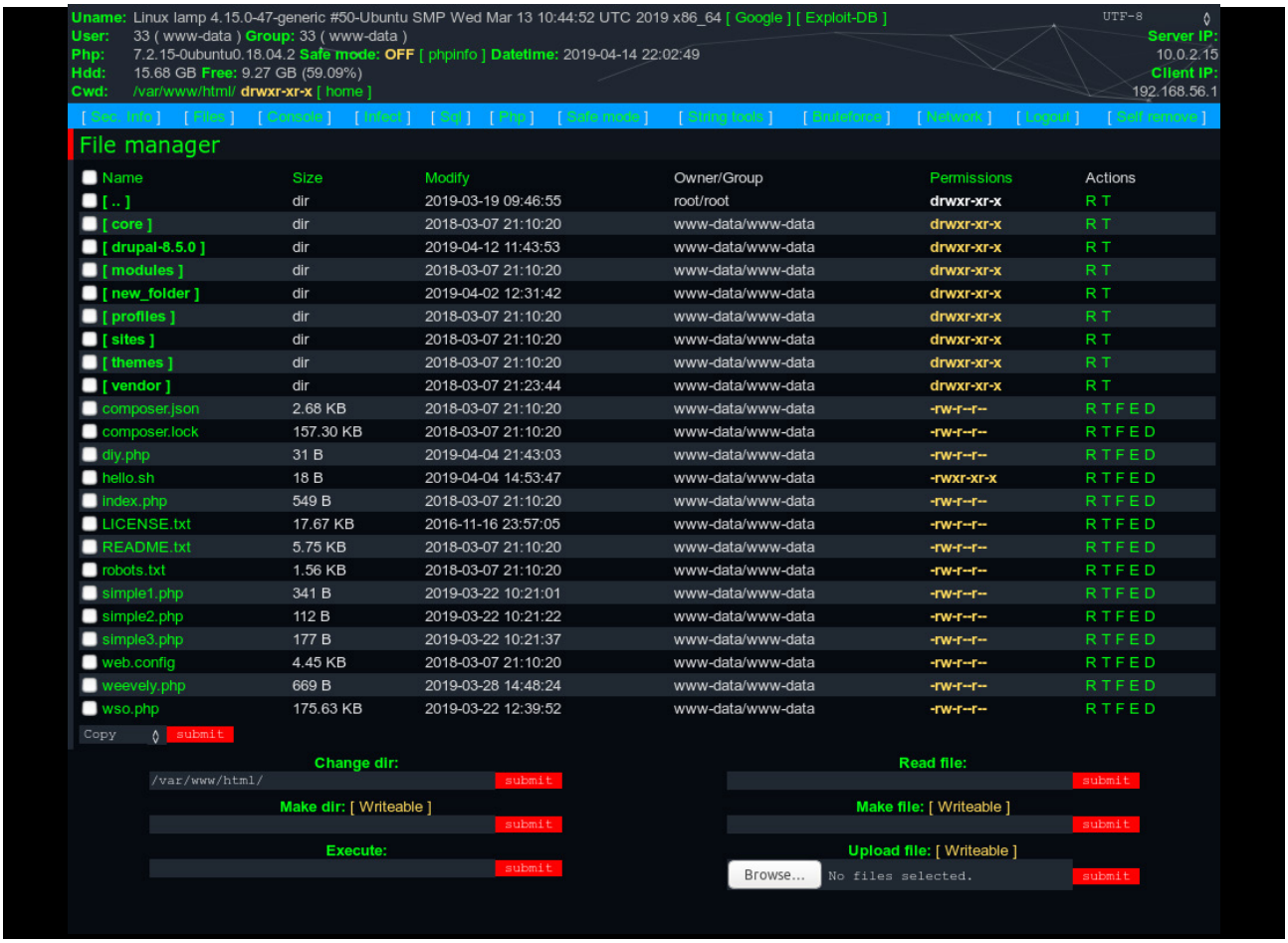
هنگامی که وب‌شل آپلود شد مهاجم می‌تواند از آن برای اکسپلویت سیستم‌عامل یا اهداف دیگر خود استفاده کند. توجه داشته باشید که این فرآیند از عامل[1] به عامل و هم‌چنین از یک وب‌شل به وب‌شل دیگر متفاوت است، چرا که انواع مختلفی از وب‌شل‌ها با قابلیت‌های گوناگون وجود دارند.

بعضی از آن‌ها بسیار ابتدایی هستند و به سادگی با دنیای‌بیرون ارتباط برقرار می‌کنند تا به مهاجم این اجازه را بدهند که کدهای خود را به صورت هدفمند اجرا کنند. از طرفی وب‌شل‌های پیچیده‌تری نیز وجود دارند که هم قابلیت‌های بسیار گسترده‌ای دارند و هم شناسایی آن‌ها دشوارتر است.

---

۱. Actor

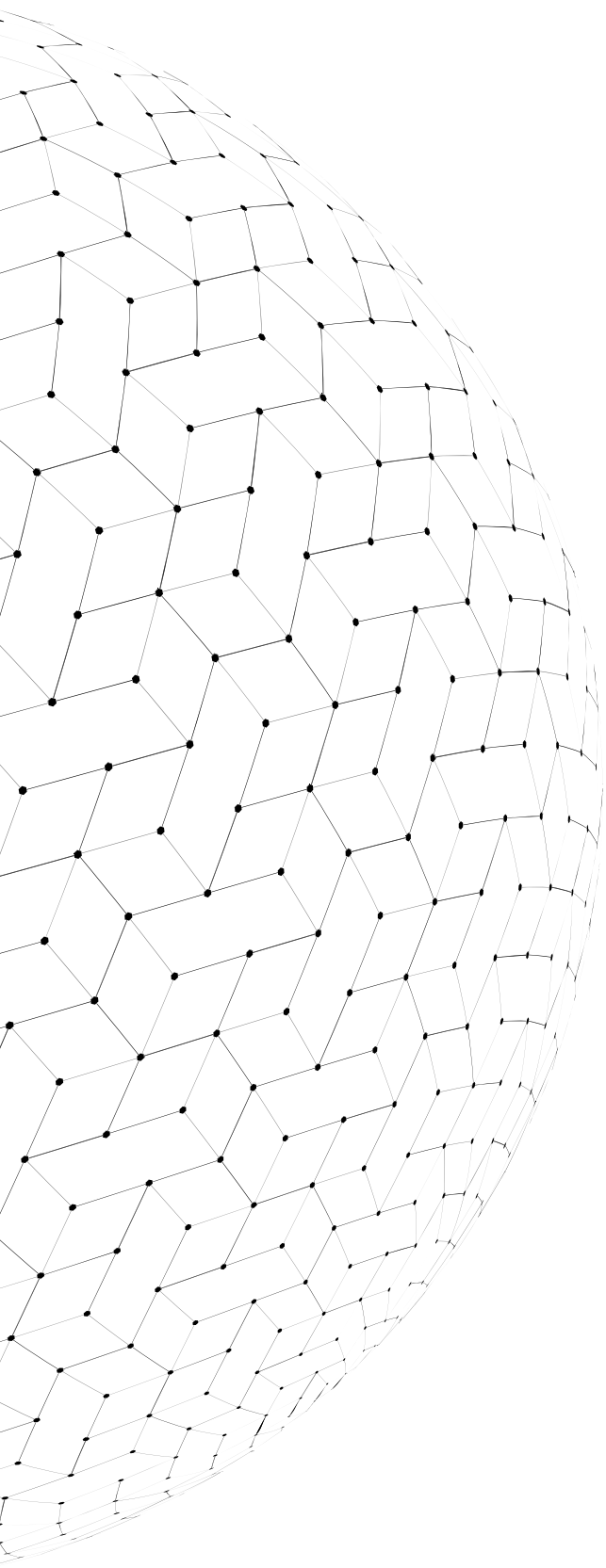
در شکل «۱» نمونه‌ای از رابط کاربری وب‌شل نمایش داده شده است.



شکل ۱: یک نمونه رابط کاربری وب‌شل با امکانات مختلف

البته تمام وب‌شل‌ها از یک رابط کاربری مشابه شکل «۱» استفاده نمی‌کنند. به عنوان مثال برخی از وب‌شل‌ها به گونه‌ای هستند که مهاجمین با استفاده از یک ابزار رابط در رایانه‌ی خود با آن‌ها ارتباط برقرار می‌کنند. برای نمونه می‌توان به وب‌شل China Chopper که توسط تیم‌های APT جدید استفاده می‌شود، اشاره کرد. فارغ از ساختار، وب‌شل‌ها می‌توانند بسیار خطرناک و در عین حال معمول و رایج باشند. مرکز US-CERT این وب‌شل‌ها را به عنوان ابزارهایی می‌شناسد که هم مجرمان سایبری خرده پا و هم تیم‌های APT از آن در حملات خود بهره می‌برند.

هم‌چنین وب‌شل‌ها می‌توانند قابلیت‌های مختلفی را به مهاجمین ارایه دهند. برخی از وب‌شل‌ها مانند شکل «۱» امکانات متنوعی شامل مدیریت فایل، ارتباط با پایگاه‌های داده، اجرای حملاتی



مانند Bruteforce، اجرای دستورات بر روی سیستم‌عامل و غیره را ارایه می‌دهند. برخی دیگر ممکن است با هدف مقابله با شناسایی شدن توسط تیم امنیت سازمان قربانی، وب‌شل خود را به صورت حداقلی و تنها با یک تابع برای اجرای دستور یا آپلود فایل طراحی کنند.

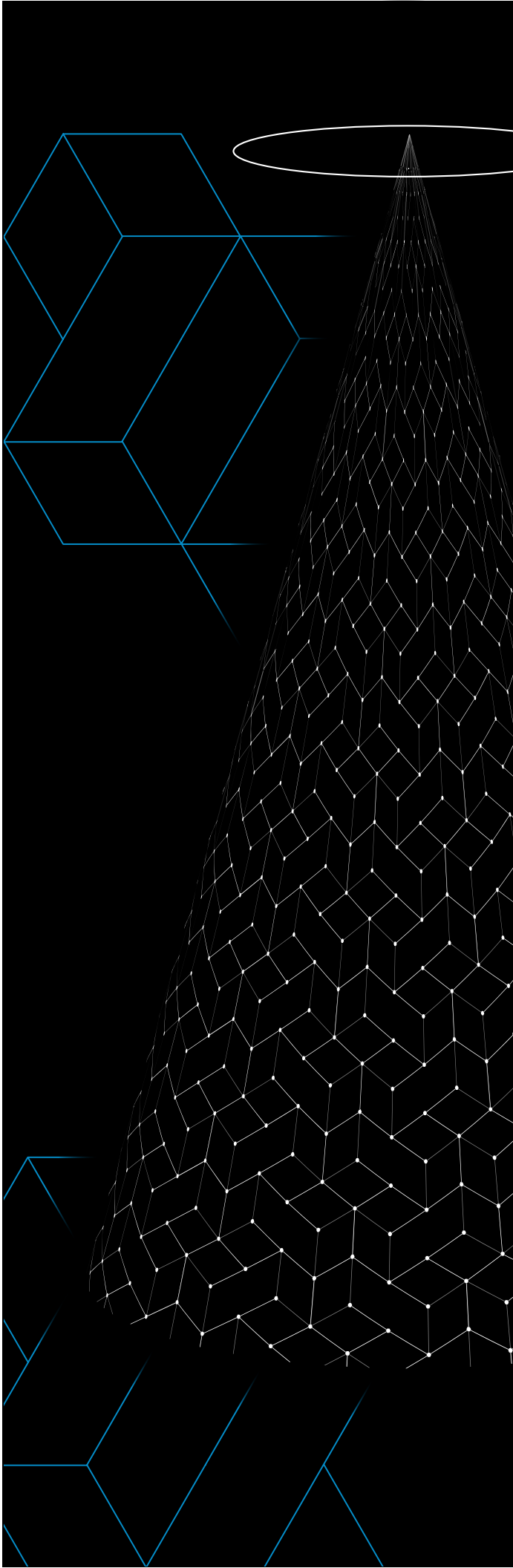
یک برداشت غلط در مورد وب‌شل‌ها این است که فقط سیستم‌هایی که با اینترنت در ارتباط هستند در معرض خطر این بدافزار قرار دارند. درحالی‌که بسیار متداول است که مهاجمین سایبری وب‌شل‌های خود را بر روی سرورهایی که به اینترنت متصل نیستند نیز نصب می‌کنند.

به عنوان مثال می‌توان به سامانه‌های مدیریت محتوای داخلی یا رابط کاربری تجهیزات مدیریت شبکه اشاره کرد.در حالت کلی سامانه‌های وب داخلی اغلب در برابر حملات سایبری آسیب‌پذیرتر هستند، زیرا کمتر مورد ارزیابی یا پایش امنیتی قرار می‌گیرند.

۲ <شناسایی وب‌شل>

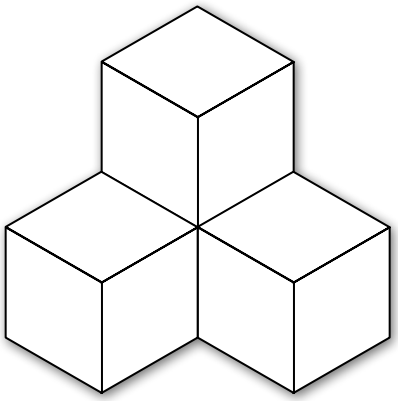
شناسایی وب‌شل‌ها می‌تواند کار دشواری باشد چراکه ویژگی‌ها و رفتار آن‌ها به سادگی توسط مهاجم تغییر داده می‌شود و اغلب با رمزنگاری، کدگذاری و مبهم‌سازی[۱] همراه هستند. اما با یک رویکرد عمیق دفاعی و با استفاده‌ی هم‌زمان از چندین تکنیک مختلف شناسایی، می‌توان بدافزارهای وب‌شل را با احتمال بالایی کشف و شکار کرد.

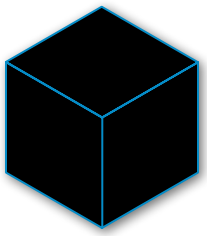
اما چالش بعدی این است که امکان دارد روش‌های شناسایی وب‌شل، به اشتباه فایل‌های بی خطر را نیز به عنوان بدافزار شناسایی کنند. بنابراین زمانی‌که یک وب‌شل شناسایی می‌شود مدیران شبکه یا امنیت باید این فایل‌ها را قبل از هر اقدامی، اعتبارسنجی کنند. مهم‌ترین تکنیک‌های شناسایی و شکار وب‌شل در بخش‌های بعد معرفی و بررسی شده است.



۱. Obfuscation

۲.۱ <مقایسه‌ی فایل‌ها با نسخه‌ی شناخته شده و معتبر>





وب‌شل‌ها در درجه‌ی اول، فایل‌ها و کدهای موجود در اپلیکیشن‌های وب را هدف قرار داده و بر اساس ایجاد یک فایل جدید یا تغییر فایل‌های موجود، نصب می‌شوند. بنابراین یکی از بهترین روش‌های کشف وب‌شل‌ها این است که به صورت دوره‌ای یا پس از مشاهده‌ی یک فعالیت مشکوک، فایل‌های موجود در دایرکتوری‌های سامانه‌ی تحت وب را با نسخه‌ی اولیه یا نسخه‌های معتبری که با همین هدف از آن‌ها یک نسخه‌ی کپی تهیه شده است، مقایسه کرد. خود فایل‌های معتبر را نیز باید حداکثر به صورت سالانه مورد ارزیابی قرار داد تا از امنیت و سالم بودن آن‌ها اطمینان حاصل شود.

زمانی‌که تفاوت فایل‌های موجود در دایرکتوری وب فعال را با تصویر نسخه‌های معتبر آن‌ها مقایسه می‌کنید، توصیه می‌شود به Timestamp فایل‌های موجود در سیستم‌های مشکوک اعتماد نکنید. چراکه برخی از مهاجمین با استفاده از تکنیکی به نام Timestomping، Timestamp مربوط به زمان «ساخته شدن» یا «آخرین تغییر» فایل‌ها را عوض می‌کنند تا بتوانند ظاهر موجه‌تری به فایل وب‌شل خود بدهند. به عنوان مثال پس از تغییر یک فایل معتبر، Timestamp آخرین تغییر آن را به چند ماه پیش تغییر می‌دهند تا شما را به اشتباه بیندازند.

کارشناسان سازمان نباید تصور کنند که Timestamp فایل‌ها حتما معتبر هستند و بر اساس آن تصمیم‌گیری کنند. اگرچه، به عنوان یک روش پیش‌فرض ممکن است کارشناسان سازمان، تایید فایل‌ها را با توجه به Timestamp های غیر معمولی اولویت‌بندی کنند.

اسکریپت‌ها و ابزارهایی که در ادامه معرفی شده است، می‌تواند برای مقایسه‌ی فایل‌های موجود در دایرکتوری یک وب‌سایت فعال با یک تصویر معتبر (Known-Good Image) از آن‌ها در دایرکتوری دیگر مورد استفاده قرار گیرد. این اسکریپت‌ها نیاز دارند تا به فایل‌های وب‌سایت فعال و تصویر تهیه شده از آن‌ها به طور هم‌زمان دسترسی داشته باشند، لذا اسکریپت باید بر روی سرور وبی که سایت را پشتیبانی می‌کند قرار بگیرد یا بر روی سیستمی که یک درایو map شده از سرور وب را دارد. هم‌چنین اسکریپت باید با سطح دسترسی کافی اجرا شود که بتواند مجوز خواندن بر روی هر دو دایرکتوری را داشته باشد.

۲.۱.۱ <اسکریپت PowerShell برای سرورهای ویندوز>

در شکل «۲» یک نمونه اسکریپت PowerShell برای مقایسه‌ی تفاوت‌های موجود بین فایل‌های موجود در دو دایرکتوری مختلف ارایه شده است. با استفاده از این اسکریپت می‌توانید فایل‌هایی که با نسخه‌ی ابتدایی یا شناخته شده‌ی خود تفاوت دارند را شناسایی و بررسی کنید.

```
.\dirChecker.ps1 -knownGood <known-good image path> -productionImage
<production image path>

<#
.DESCRIPTION
The script looks for files changes/additions between a production
directory (target) and a known-good directory.

.PARAMETER knownGood
Path of the known-good directory.

.PARAMETER productionImage
Path of the production directory (target).

-- Output --
File analysis started.
Any file listed below is a new or changed file.

C:\inetput\wwwroot\index2.aspx

File analysis completed.
#>
param (
```

```
[Parameter(Mandatory=$TRUE)][ValidateScript({Test-Path $_ -PathType
‹Container›})][String] $knownGood,
[Parameter(Mandatory=$TRUE)][ValidateScript({Test-Path $_ -PathType
‹Container›})][String] $productionImage
)

# Recursevely get all files in both directories, for each file calculate
hash.
$good = Get-ChildItem -Force -Recurse -Path $knownGood | ForEach-Object {
Get-FileHash -Path $_.FullName }
$prod = Get-ChildItem -Force -Recurse -Path $productionImage | ForEach-
Object { Get-FileHash -Path $_.FullName }

Write-Host «File analysis started.»
Write-Host «Any file listed below is a new or changed file.`n»


# Compare files hashes, select new or changed files, and print the
path+filename.
(Compare-Object $good $prod -Property hash -PassThru | Where-Object{$_.
SideIndicator -eq ‹=››}).Path

Write-Host «`nFile analysis completed.»
```

شکل ۲: اسکریپت PowerShellبرای مقایسه بین فایل‌های دو دایرکتوری مختلف

۲.۱.۲ <ابزار WinDiff>

مایکروسافت برای سیستم‌عامل‌های ویندوز ابزاری به نام WinDiff را توسعه داده که با استفاده از یک اینترفیس گرافیکی امکان مقایسه‌ی دایرکتوری‌ها را به شما می‌دهد. فایل این ابزار و راهنمای استفاده از آن در آدرس زیر قابل دسترس است:

https://support.microsoft.com/en-us/help/159214/how-to-use-the-windiff-exe-utility

۲.۱.۳ <دستورات قابل استفاده از سرورهای وب لینوکسی>

دستورات و اسکریپت‌های متعددی برای مقایسه‌ی دایرکتوری‌ها در لینوکس وجود دارد که با یک جستجوی ساده در اینترنت می‌توان به آن‌ها دسترسی داشت. برخی از دستورات پیشنهادی به شرح زیر است:

- دستوری برای پیدا کردن تفاوت‌های احتمالی موجود بین فایل‌های دو دایرکتوری:

```
diff -r -q <known-good image path> <production image path>
```

- دستوری برای پیدا کردن فایل‌هایی که طی یک روز اخیر (۲۴ ساعت) ایجاد شده‌اند:

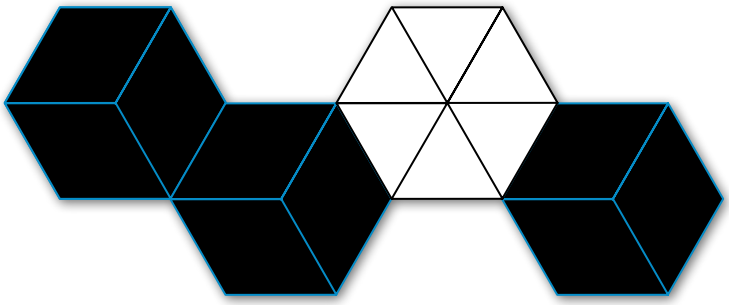
```
find . -type f -name '*.php' -mtime -1
```

- دستوری برای پیدا کردن تمام فایل‌های php که از تابع eval() در آن‌ها استفاده شده است:

```
find . -type f -name '*.php' | xargs grep -l "eval*("
```

و ده‌ها دستور و اسکریپت دیگر به که راحتی می‌توان با یک جستجوی ساده به آن‌ها دسترسی داشت.

۲.۲ <شناسایی وب‌شل‌ها با استفاده از ناهنجاری‌های ترافیک شبکه>



با این‌که اغلب مهاجمین وب‌شل‌های خود را به‌گونه‌ای طراحی می‌کنند تا ترافیک آن با ترافیک عادی وب در سازمان قربانی ترکیب شود، اما پنهان کردن برخی از ویژگی‌های آن دشوار است و نیاز به دانش بالایی در سمت مهاجمین دارد. این ویژگی‌ها شامل مواردی مانند User Agent String و آدرس IP کاربر می‌شود. پیش از حضور کامل در شبکه، بعید است مهاجمین از این‌که کدام User Agentها و آدرس‌های IP برای یک سرور وب عادی است آگاهی داشته باشند، بنابراین درخواست‌های وب‌شل ویژگی‌های غیر عادی خواهد داشت. URIهایی[1] هم که به صورت انحصاری توسط User Agentهای غیر عادی مورد دسترسی قرار می‌گیرند، به احتمال زیاد وب‌شل هستند.

برخی از مهاجمین نیز فراموش می‌کنند تا در درخواست‌های وب‌شل، هدر Referer را مشابه ترافیک عادی وب در شبکه استتار کنند. در نتیجه، درخواست‌هایی بدون هدر Referer یا با هدر غیر عادی نیز می‌تواند نشان‌دهنده‌ی وجود وب‌شل‌ها باشد.

با استفاده از قابلیت‌های یک سامانه‌ی تحلیل لاگ مرکزی مانند سامانه‌ی SIEM می‌توان این‌گونه تحلیل‌ها را به راحتی پیاده‌سازی کرد. اگر چنین امکاناتی در سازمان وجود نداشته باشد، مدیران شبکه اغلب می‌توانند با استفاده از اسکریپت‌نویسی برای پالایش و تحلیل وقایع سرورهای وب، URLهای احتمالی مربوط به وب‌شل‌ها را شناسایی کنند. چند نمونه از قواعد جستجوی Splunk و هم‌چنین اسکریپت‌هایی برای تحلیل لاگ در بخش‌های بعد آمده است.

۱. Uniform Resource Identifiers

۲.۲.۱ <رول‌های جستجوی Splunk با هدف شناسایی URIهای غیرعادی در ترافیک وب>

همان‌گونه که در بخش‌های قبل شرح داده شد، علیرغم تلاش مهاجمین برای پنهان‌سازی ترافیک وب‌شل خود در بین حجم بالای ترافیک وب سازمان، بسیاری از وب‌شل‌ها ویژگی‌های غیر عادی در ترافیک خود دارند که می‌توان از آن‌ها برای شناسایی بهره برد.

عباراتی که در ادامه ارایه شده، برای شناسایی URIهای مربوط به وب‌شل با استفاده از User Agent و آدرس‌های IP غیر عادی در Splunk کاربرد دارد. توصیه می‌شود که این نمونه Queryها را به جای اجرا در سراسر شبکه، تنها برای محیط‌های محدودتر مانند وب اپلیکیشن‌های خاص یا سرورهای وب اجرا نمایید. در برخی موارد ممکن است وب اپلیکیشن‌ها برای هر درخواست یک URI خاص تولید کنند که در این‌صورت از میزان اثر بخشی این Queryها کاسته خواهد شد.

۱. برخلاف URIهای عادی، URIهای مربوط به وب‌شل‌ها اغلب تعداد User Agentهای کمی دارند. در Query زیر از این نکته برای شناسایی URIهای غیرعادی استفاده شده است.

جدول ۱: رول‌های Splunk برای شناسایی URIهای غیر عادی

برای سرورهای وب Apache	sourcetype=»access_combined”
	| fillnull value=- ‘comment(“Fill all empty fields with -”)’
	| search status>=»200» status <»300» uri!=- clientip!=-
	`comment(«Only successful codes 299-200, eliminate blank URIs and client IPs»)`
	| stats min(_time) as start max(_time) as stop dc(useragent) as dc_user_agent values(useragent) as values_user_agent dc(clientip) as dc_src values(clientip) as values_src count by uri
	`comment(«Find first and last time the grouping was found, number of distinct User Agent strings and IP addresses used to access that URI»)`
	| convert ctime(start) ctime(stop) `comment(«Convert the times to a readable format»)`
	| search dc_src<=5 OR dc_user_agent<=5 `comment(«Only URIs with <=5 unique user agents or IP addresses»)`
	| table start stop uri dc_user_agent values_user_agent dc_src values_src

برای سرورهای وب IIS	sourcetype=»iis»
	| fillnull value=- ‘comment(“Fill all empty fields with -”)’
	| search sc_status>=»200» sc_status <»300» cs_uri_stem!=- c_ip!=-
	`comment(«Only successful codes 299-200, eliminate blank URIs and client IPs»)`
	| stats min(_time) as start max(_time) as stop dc(cs_User_Agent) as dc_user_agent values(cs_User_Agent) as values_user_agent dc(c_ip) as dc_src values(c_ip) as values_src count by cs_uri_stem
	`comment(«Find first and last time the grouping was found, number of distinct User Agent strings and IP addresses used to access that URI»)`
	| convert ctime(start) ctime(stop) `comment(«Convert the times to a readable format»)`
	| search dc_src<=5 OR dc_user_agent<=5 `comment(«Only URIs with <=5 unique user agents or IP addresses»)`
	| table start stop cs_uri_stem dc_user_agent values_user_agent dc_src values_src

۲. User Agentهای غیر عادی به‌خصوص در سامانه‌های وب داخلی، می‌تواند نشان دهنده‌ی فعالیت وب‌شل‌ها باشد. Query زیر با همین هدف ارایه شده است.

جدول ۲: رول‌های Splunk برای شناسایی User Agentهای غیر عادی

برای سرورهای وب Apache	sourcetype=»access_combined» | fillnull value=- ‘comment(“Fill all empty fields with -”)’ | search status>=»200» status <»300» ` comment(«Only successful codes 299-200”)` | stats count by useragent `comment(«Group User Agent strings to determine frequency»)` | sort + count `comment(«Sort count in ascending order»)` | head 10 `comment(«Limit results to top 10. This can be changed to see more or fewer results»)`
برای سرورهای وب IIS	sourcetype=»iis» sc_status>=»200» AND sc_status<»300» ` comment(«Only successful codes 299-200”)` | fillnull value=- ‘comment(“Fill all empty fields with -”)’ | search sc_status>=»200» sc_status <»300» `comment(«Only

```
successful codes 299-200»)`
| stats count by cs_User_Agent `comment(«Group User Agent strings
to determine frequency»)`
| sort + count `comment(«Sort count in ascending order»)`
| head 10 `comment(«Limit results to top 10. This can be changed
to see more or fewer results»)`
```

۳. مقادیر غیر عادی HTTP Referer می‌تواند نشان‌دهنده‌ی فعالیت وب‌شل‌ها باشد. با استفاده از Query زیر می‌توان این موارد را شناسایی کرد.

جدول ۳: رول‌های Splunk برای شناسایی مقادیر غیر عادی HTTP Referer

برای سرورهای وب Apache	`sourcetype=»access_combined»` `| fillnull value=- 'comment("Fill all empty fields with - (needed to make blank referer fields searchable)")'` `| search status>=»200» status <»300» `comment(«Only successful codes 299-200»)`` `| stats dc(uri) as dc_URIs values(uri) as All_URIs count by referer `comment(«Counts number of times each URI request is associated with a unique referer»)`` `| table referer, All_URIs, dc_URIs` `| sort + dc_URIs `comment(«Sort count in ascending order»)` `| head 10 `comment(«Limit results to top 10. This can be changed to see more or fewer results»)``
برای سرورهای وب IIS	`sourcetype=» iis»` `| fillnull value=- 'comment("Fill all empty fields with - (needed to make blank referer fields searchable)")'` `| search sc_status>=»200» sc_status<»300» `comment(«Only successful codes 299-200»)` `| stats dc(cs_uri_stem) as dc_URIs values(cs_uri_stem) as All_URIs count by cs_Referer `comment(«Counts number of times each URI request is associated with a unique referer»)` `| table cs_Referer, All_URIs, dc_URIs` `| sort + dc_URIs `comment(«Sort count in ascending order»)` `| head 10 `comment(«Limit results to top 10. This can be changed to see more or fewer results»)``

۴. مقادیر خالی برای HTTP Referer نیز می‌تواند نشان‌دهنده‌ی فعالیت وب‌شل‌ها باشد. از Query زیر می‌توان برای شناسایی این موارد بهره برد.

جدول ۴: رول‌های Splunk برای شناسایی بسته‌های با مقادیر خالی در فیلد HTTP Referer

برای سرورهای وب Apache	sourcetype=»access_combined» | fillnull value=- ‘comment(“Fill all empty fields with - (needed to make blank referer fields searchable)”)’ | search status>=”200” status<”300” referrer=- uri!=”/” `comment(«Only successful codes 299-200 and blank referrer not from root webpage») | stats count by referer, uri `comment(«Counts number of times each URI request is associated with a unique referer»)` | table uri, count | sort - count `comment(«Sort count in descending order»)` | head 10 `comment(«Limit results to top 10. This can be changed to add more or fewer results»)`
برای سرورهای وب IIS	sourcetype=»iis» | fillnull value=- ‘comment(“Fill all empty fields with - (needed to make blank referer fields searchable)”)’ | search sc_status>=»200» sc_status<»300» sc_Referer=- cs_uri_stem!=»/» `comment(«Only looking for successful status codes -200 299 and blank referer not from the root webpage»)` | stats count by cs_Referer, cs_uri_stem `comment(«Counts number of times each URI request is associated with a unique referer»)` | table cs_uri_stem, count | sort - count `comment(«Sort count in descending order»)` | head 10 `comment(«Limit results to top 10. This can be changed to add more or fewer results»)`

۲.۲.۲ <نمونه اسکریپت‌های شناسایی وب‌شل با استفاده از تحلیل درخواست‌های غیر عادی>

اسکریپت‌های PowerShell و Python زیر را می‌توان برای شناسایی URIهای درخواست شده توسط User Agentها یا آدرس‌های IP غیر عادی، به کار برد. مشابه Queryهای بخش قبل، در مواردی که وب اپلیکیشن‌ها برای هر درخواست یک URI خاص تولید کنند، از میزان اثر بخشی این اسکریپت‌ها کاسته خواهد شد.

۲.۲.۲.۱ <اسکریپت PowerShell برای تحلیل لاگ‌های IIS>

اسکریپت شکل «۳» با هدف تحلیل وقایع سرورهای IIS در ویندوز ارایه شده است. برای استفاده از آن کافیست تا دستور زیر را در محیط PowerShell اجرا کنید:

```
.\Script.ps1 -logDir <path to IIS log directory>
```

پس از پارامتر logDir باید مسیر لاگ‌های ذخیره شده‌ی سرور IIS را بنویسید.

```
#Specify Default parameters
Param (
    [ValidateScript({Test-Path $_ -PathType ‹Container›})][string]$logDir
= «C:\inetpub\logs\»,
    [ValidateRange(1,100)][int]$percentile = 5
)

If ($ExecutionContext.SessionState.LanguageMode -eq
«ConstrainedLanguage»)
    { Throw «Error: must use Full Language Mode (https://devblogs.
microsoft.com/powershell/powershell-constrained-language-mode/)» }

function analyzeLogs ( $field ) {
    $URIs = @{}
    $files = Get-ChildItem -Path $logDir -File -Recurse
    If ($files.Length -eq 0)  { «No log files at the given location
`n$($_)»; Exit }

    #Parse each file for relevant data. If data not present, continue to
next file
    $files | Foreach-Object {
        Try {
            $file = New-Object System.IO.StreamReader -Arg $_.FullName
```

```
            $Cols = @()
            While ($line = $file.ReadLine()) {
                If ($line -like «#F*») {
                    $Cols = getHeaders($line)
                } ElseIf ($Cols.Length -gt 0 -and $line -notlike «#*» ) {
                    $req = $line | ConvertFrom-Csv -Header $Cols
-Delimiter ‹ ‹
                    If ( IrrelevantRequest $req ) { Continue; }
                    #If target field seen for this URI, update our data;
otherwise create data object for this URI/field
                    If ($URIs.ContainsKey($req.uri) -and $URIs[ $req.uri
].ContainsKey($req.$field) )
                        { $URIs[ $req.uri ].Set_Item( $req.$field, $URIs[
$req.uri ][ $req.$field ] + 1 ) }
                    ElseIf ($URIs.ContainsKey($req.uri))
                        { $URIs[ $req.uri ].Add( $req.$field, 1 ) }
                    Else
                        { $URIs.Add($req.uri, @{ $($req.$field) = 1 }) }
                }
            }
            $file.close()
        } Catch {
            Echo «Unable to parse log file $($_.FullName)`n$($_)»
        }
    }

    Echo «These URIs are suspicious because they have the least number of
$($field)s requesting them:»
    $nth_index = [math]::ceiling( ($URIs.Count) * ([decimal]$percentile /
100))

    #Count the unique fields for each URI
    ForEach ($key in $($uris.keys)) { $uris.Set_Item( $key, $uris.$key.
Count) }
    $URIs.GetEnumerator() | sort Value | Foreach-Object {
        $uri_i = If (Get-Variable ‹uri_i› -Scope Local -ErrorAction
Ignore) { $uri_i + 1 } Else { 0 } #initialize/increment counter
        If($uri_i -le $nth_index) { Echo «   $($_.Name) is requested by
$($_.Value) $($field)(s)» }
   }
}


Function getHeaders ( $s ) {
    $s = (($s.TrimEnd()) -replace «#Fields: «, «» -replace «-»,»»
```

```
-replace «\(«,»» -replace «\)»,»»)
    $s = $s -replace «scstatus»,»status» -replace «csuristem»,»uri»
-replace «csUserAgent»,»agent» -replace «cip»,»ip»
    Return $s.Split(‹ ›)
}

Function IrrelevantRequest ( $req ) {
    #Skip requests missing required fields
    ForEach ($val in @(«status», «uri»,»agent»,»ip»))
        { If ($val -notin $req.PSobject.Properties.Name) { Return $True}
}
    #We only care about requests where the server returned success (codes
299-200)
    If ($req.status -lt 200 -or $req.scstatus -gt 299)
        { Return $True }
    Return $False
}

analyzeLogs «agent»
analyzeLogs «ip»
```

شکل ۳: اسکریپت PowerShell برای تحلیل لاگ‌های IIS

۲.۲.۲.۲ <اسکریپت Python برای تحلیل لاگ‌های APACHE>

با استفاده از اسکریپت شکل «۴» می‌توانید لاگ‌های سرور APACHE در محیط لینوکس را تحلیل کرده و URIهای مشکوک را کشف کنید. کافیست دستور زیر را در دایرکتوری که اسکریپت را در آن ذخیره کرده‌اید اجرا نمایید:

```
/Script.py <path to APACHE log file>
```

```
import sys
import os.path
import csv

# Script will generate a list of URL that from Apache web access log that
have least unique IP address or unique user-agents
# Written for Python 3

# Ideal Percentage of URL to display
#   Will display more base on matching count
```

```
urlpercentage = 0.05

# Hold the filename for Apache web access log
weblogfileName = None

# apache log fields
apachelogsfields = [‹ip›, ‹identd›, ‹frank›, ‹time_part0›, ‹time_part1›,
‹request›, ‹status›, ‹size›, ‹referer›, ‹user_agent›]

# function output the url based on lower counts unique ip address and
lower counts of unique user-agents
def analyze_weblog(filename):

    uniqueurlcount = 0                    # count of unique URL in web
log
    urls = []                             # list of unique URL, also
index into lists of lists of unique ip address and user-agents
    uniqueipcount = []                    # list of unique ip address
count for URL
    uniqueuseragentscount = []            # list of unique use agents for
URL
    iplist = []                           # list of list of ip address
per unique URL to keep track of unique URL
    useragentlist = []                    # list of list of user-agents
per unique URL to keep track of unique user-agents

    print(«The weblog file to analyze is %s» % filename)
    with open(filename, mode=›r›) as csv_file:                    # read in
web log as csv file
        csv_reader = csv.reader(csv_file, delimiter=› ‹)
        for row in csv_reader:

            # handles simple case where file has comments start with #
            if (row[0][0] != ‹#›):
                                                                  #
extract only fields of interest from the web log
                ipaddress = row[apachelogsfields.index(‹ip›)]        # ip
address
                request = row[apachelogsfields.index(‹request›)]     #
request (URL part of request)
                status = row[apachelogsfields.index(‹status›)]       #
user-agent
                user_agent = row[apachelogsfields.index(‹user_agent›)]
    #           print(‹ipaddress: %s request: %s status: %s user_agent:
%s› % (ipaddress, request, status, user_agent))
```

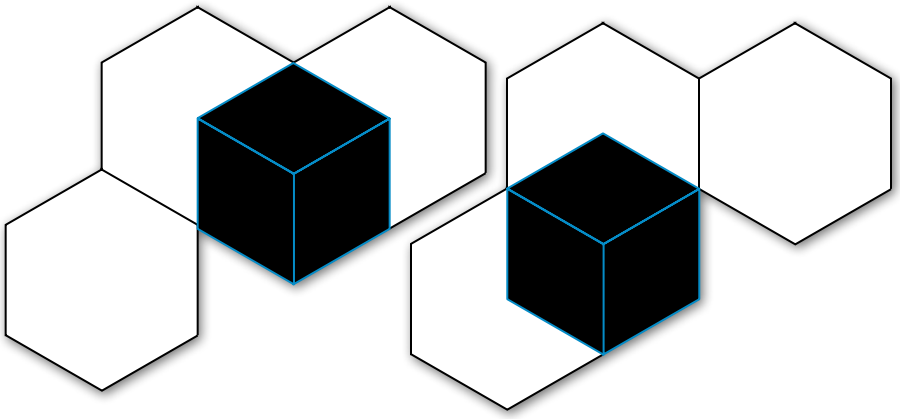
```
            url = (request.partition(‹ ›)[2]).partition(‹ ›)[0]  #
extract URL from request field
    #           print (‹url %s› % url)
            if (status >= ‹200› and status <= ‹299›):            #
only request with status of 299 - 200

                if (url not in urls):                           #
determine if URL is already been seen
                    uniqueurlcount += 1                         # if not
increment unique URL count
                    urls.append(url)                            # append
new URL to the unique URL list
                    uniqueipcount.append(0)                     # append
an element of zero for the unique ip count list
                    uniqueuseragentscount.append(0)             # append
an element of zero for the unique user-agents count list
                    newiplist = []                              # new
empty element list for ip address tracking per URL
                    iplist.append(newiplist)                    # append
empty list to list of list of ip per URL
                    newuseragentlist = []                       # new
empty element list for user-agents tracking per URL
                    useragentlist.append(newuseragentlist)      # append
empty list to list of list of user-agents per URL

                if (user_agent not in useragentlist[urls.index(url)]):
# determine if user-agents is in the particular URL list
                    useragentlist[urls.index(url)].append(user_agent)
# if not append user-agents to user-agents list for the particular URL
list
                    temp = uniqueuseragentscount[urls.index(url)] + 1
# also increment unique user-agents count for that URL
                    uniqueuseragentscount[urls.index(url)] = temp
                if (ipaddress not in iplist[urls.index(url)]):
# determine if ip address is in the particular URL list
                    iplist[urls.index(url)].append(ipaddress)
# if not append ip address to ip address list for the particular URL list
                    temp = uniqueipcount[urls.index(url)] + 1
# also increment unique ip address count for that URL
                    uniqueipcount[urls.index(url)] = temp


#       print(urls)
#       print(‹uniqueurlcount: %s› % uniqueurlcount)
#       print(uniqueuseragentscount)
```

```
#        print(uniqueipcount)
#        print(‹amount of useragentlist: %s› %
len(uniqueuseragentscount))
#        print(‹amount in the iplist: %s› % len(uniqueipcount))
        numberofurltodisplay = urlpercentage * uniqueurlcount       #
Determine line that represent percentage of URL wanted
        intnumberofurltodisplay = int(numberofurltodisplay)
        if (numberofurltodisplay > intnumberofurltodisplay):        #
Round up
            intnumberofurltodisplay += 1
        tempuniqueuseragentscount = uniqueuseragentscount.copy()    #
Create a temporary copy of list of unqiue user-agents count to sort
        tempuniqueuseragentscount.sort()
                                                                    #
Array start at 0 need to subtract 1- from index
        useragentcounttodisplay = tempuniqueuseragentscount[(intnumberofu
rltodisplay 1-)] # determine the count of unique user-agents to display
        tempuniqueipcount = uniqueipcount.copy()                    #
Create a temporary copy of list of unqiue ip address count to sort
        tempuniqueipcount.sort()
                                                                    #
Array start at 0 need to subtract 1- from index
        ipcounttodisplay = tempuniqueipcount[(intnumberofurltodisplay
1-)]               # determine the count of ip address to display

        print(‹URL with least user agents›)
        print(‹-------------------------›)

        for count in range (0, (useragentcounttodisplay + 1)):  #
Increament thru count to count of unique user-agents to display to order
url output based on count
            index = 0
            for elementuseragentcount in uniqueuseragentscount:         #
Increment thru unique user-agents count list
                if (elementuseragentcount == count):                    #
List URL where where user-agents is equal to count
                    print(urls[index])
                index += 1

        print(‹URL with least IP address›)
        print(‹-------------------------›)

        for count in range (0, (ipcounttodisplay + 1)):    # Increament
```

```
thru count to count of unique ip address to display to order url output
based on count
            index = 0
            for elementipcount in uniqueipcount:                          #
Increment thru unique ip address count list
                if (elementipcount == count):                             #
List URL where where user-agents is equal to count
                    print(urls[index])
                index += 1

if __name__ == ‹__main__›:
   try:
       if len(sys.argv) == 2:
# Simple check if an agrument is passed (assume weblog file
           weblogfileName=sys.argv[1]
           print («Web log file to read is %s» % weblogfileName)
           if(os.path.isfile(weblogfileName)):
                analyze_weblog(weblogfileName)
       else:
           print (‹Usage: python3 %s <weblogfile>› % sys.argv[0])
# Print usage statement
   except Exception as e:
        print («You must provide a valid filename (path) of a web logfile»)
        raise
```

شکل ۴: اسکریپت Python برای تحلیل لاگ‌های APACHE

۲.۳ <کشف وب‌شل‌های شناخته شده>



روش‌های شناسایی مبتنی بر امضا[1] نیز می‌تواند برای شناسایی وب‌شل‌های شناخته شده، مفید باشد. اما تکیه بر این روش در برخی موارد می‌تواند تا حد زیادی غیر قابل اطمینان باشد، چرا که وب‌شل‌ها ممکن است مبهم‌سازی شده یا ویژگی‌های آن‌ها به سادگی توسط مهاجمین تغییر داده شوند.

به هر حال برخی از مهاجمین سایبری کماکان از وب‌شل‌های رایج و شناخته‌شده‌ای مانند R57، B374K، C99، WSO، China Copper و غیره بدون ایجاد تغییر یا با اعمال تغییرات جزیی استفاده می‌کنند. در این موارد، می‌توان از روش‌های شناسایی مبتنی بر Finger Print و یا Expression-Based نتیجه‌ی مطلوبی گرفت.

از دیدگاه شبکه نیز شناسایی وب‌شل‌ها به صورت Signature Based چندان قابل اعتماد نیست چراکه ترافیک ارتباطات وب‌شل‌ها نیز اغلب رمزنگاری یا مبهم‌سازی شده است. از سوی دیگر، عبارات Hard Code شده مانند نام متغیرها نیز به راحتی توسط مهاجمین تغییر داده می‌شود تا راحت‌تر راهکارهای شناسایی را دور بزنند. با این‌که کشف وب‌شل‌های ناشناخته با این روش بسیار بعید است، اما روش شناسایی Signature Based می‌تواند برای کشف ترافیک فایل‌های شناخته شده در شبکه کاربرد داشته باشد.

در بخش بعد مجموعه‌ای از رول‌های Snort و ابزار LOKI با هدف کشف نمونه‌هایی از وب‌شل‌های پرکاربرد، ارایه شده است.

---

۱. Signature Based

۲.۳.۱ <مجموعه رول‌های Snort با هدف شناسایی وب‌شل‌های پرکاربرد>

همان‌گونه که در بخش قبل ذکر شد ترافیک وب‌شل‌ها اغلب مبهم‌سازی یا رمزنگاری شده است. اگر سازمان شما راهکارهایی از قبیل Reverse Proxy یا فایروال‌های وب اپلیکیشن (WAF) را جهت پایش ترافیک‌های رمزنگاری شده مانند نشست‌های TLS (Transport Layer Security) داشته باشد، با استفاده از رول‌های شکل «۵» می‌توانید ترافیک وب‌شل‌های معروف (که تغییرات زیادی در آن‌ها داده نشده باشد) را شناسایی کنید. این دسته از Signatureها می‌توانند به عنوان بخشی از استراتژی چند لایه‌ی دفاع در عمق استفاده شوند.

```
# These signatures are targeted at the China Chopper web shell
# Source: https://www.fireeye.com/blog/threat-research/08/2013/breaking-
down-the-china-chopper-web-shell-part-ii.html
alert tcp any any -> any any (msg: «China Chopper with first Command
Detected»; flow:to_server,established; content: «FromBase64String»;
content: «z1»; content:»POST»; nocase;http_method; reference:url,http://
www.fireeye.com/blog/technical/botnet-activities-research/08/2013/
breaking-down-the-china-chopper-web-shell-part-i.html; sid: 90000101;)

alert tcp any any -> any any (msg: «China Chopper with all Commands
Detected»; flow:to_server,established; content: «FromBase64String»;
content: «z»; pcre: «/Z\d{1,3}/i»; content:»POST»; nocase;http_method;
reference:url,http://www.fireeye.com/blog/technical/botnet-activities-
research/08/2013/breaking-down-the-china-chopper-web-shell-part-i.html;
sid: 90000102;)

# These signatures are targeted at the C99 web shell
# Source: https://github.com/jpalanco/alienvault-ossim/blob/master/snort-
rules-default-open/rules/2.9.2/emerging.rules/emerging-web_server.rules

alert tcp any any -> any any (msg:»ET WEB_SERVER c99 Shell Backdoor
Var Override URI»; flow:to_server,established; content:»c99shcook[«;
nocase; http_uri; fast_pattern:only; pcre:»/[&?]c99shcook\[/Ui»;
reference:url,thehackerblog.com/every-c-99php-shell-is-backdoored-
aka-free-shells/; sid:2018601; rev:1; metadata:created_at 24_06_2014,
updated_at 24_06_2014;)

alert tcp any any -> any any (msg:»ET WEB_SERVER c99 Shell Backdoor Var
```

```
Override Cookie»; flow:to_server,established; content:»c99shcook»; nocase;
fast_pattern:only; pcre:»/c99shcook/Ci»; reference:url,thehackerblog.com/
every-c-99php-shell-is-backdoored-aka-free-shells/; sid:2018602; rev:1;
metadata:created_at 24_06_2014, updated_at 24_06_2014;)

alert tcp any any -> any any (msg:»ET WEB_SERVER c99 Shell Backdoor Var
Override Client Body»; flow:to_server,established; content:»c99shcook[«;
nocase; fast_pattern:only; http_client_body; pcre:»/(?:^|&)c99shcook\[/
Pi»; reference:url,thehackerblog.com/every-c-99php-shell-is-backdoored-
aka-free-shells/; sid:2018603; rev:1; metadata:created_at 24_06_2014,
updated_at 24_06_2014;)

#These signatures are targeted at the R57 web shell
# Source: nsa.gov
alert tcp any any -> any any (msg: «R57 Web shell Detected»; content:
«<title>r57 Shell Version «; rev:1; sid: 90000201;)

#These signatures are targeted at the B374k web shell
# Source: nsa.gov
alert tcp any any -> any any (msg: «B374k Web shell Detected»; content:
«<title>b374k «; rev:1; sid: 90000301;)

#These signatures are targeted at the WSO web shell
# Source: nsa.gov
alert tcp any any -> any any (msg: «WSO Web shell Detected»; content: «on
click=\»g(‹SelfRemove›,null,››,››,››)\»>Self remove</a> ]»; rev:1; sid:
90000401;)
# Source: https://rules.emergingthreatspro.com/9598411999529178/
suricata2.0-/rules/web_server.rules

alert tcp any any -> any any (msg:»ET WEB_SERVER WSO Web
Shell Activity POST structure 2»; flow:established,to_server;
content:»POST»; http_method; content:» name=|22|c|22|»; http_client_
body; content:»name=|22|p22|1|»; http_client_body; fast_pattern;
pcre:»/name=(?P<q>[\x22\x27])a(?P=q)[^\r\n]*\r\n[\r\n\s]+(?:S(?:
e(?:lfRemove|cInfo)|tringTools|afeMode|ql)|(?:Bruteforc|Consol)
e|FilesMan|Network|Logout|Php)/Pi»; sid:2016354; rev:2; metadata:created_
at 05_02_2013, updated_at 05_02_2013;)
```

شکل ۵: نمونه رول‌های Snort با هدف شناسایی وب‌شل‌های پرکاربرد

۲.۳.۲ <ابزار LOKI>

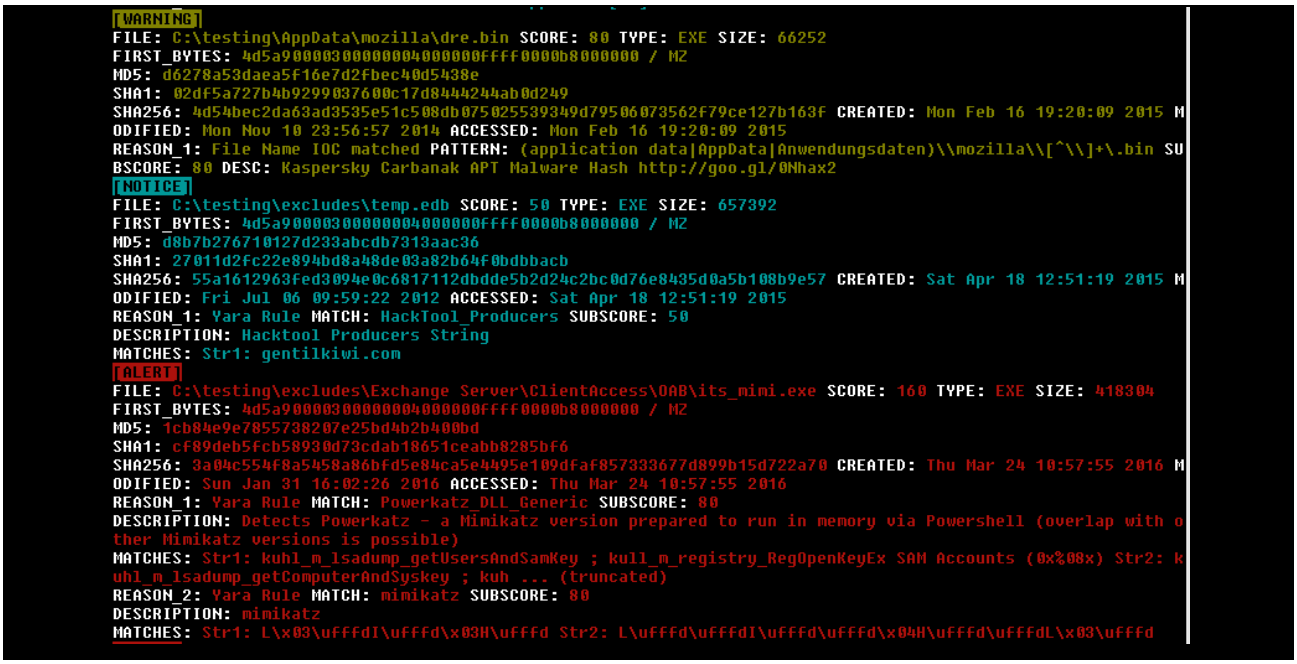
یک ابزار رایگان است که توانایی شناسایی بدافزارها با استفاده از IOC در ویندوز را دارد. در واقع ابزار LOKI توانایی شناسایی تعداد زیادی از وب‌شل‌های شناخته شده را دارد. این ابزار با استفاده از زبان Python توسعه داده شده و قابلیت شناسایی وب‌شل‌ها با استفاده از IOCهای مربوط به نام بدافزار، رول‌های YARA، بررسی Hash فایل‌ها و در نهایت مقایسه‌ی ارتباطات پروسس‌ها با IOC مربوط به ارتباط با C2های شناخته شده را دارد. از آدرس زیر می‌توانید به کد این ابزار دسترسی داشته باشید:

https://github.com/Neo23x0/Loki

فایل اجرایی کامپایل شده‌ی LOKI را نیز از آدرس زیر می‌توانید دانلود کنید:

https://github.com/Neo23x0/Loki/releases

در شکل «۶» یک نمونه از خروجی این ابزار نمایش داده شده است:



شکل ۶: نمونه‌ای از خروجی ابزار LOKI

همان‌گونه که مشاهده می‌کنید، به هر فایل با توجه به میزان مطابقت آن با IOCهای موجود یک امتیاز[۱] داده می‌شود و فایل‌هایی که از امتیاز مطابقت بالایی برخوردار هستند با رنگ قرمز نمایش داده شده است.

۱. Score

۲.۳.۳ <سایر ابزارها>

ده‌ها اسکریپت و ابزار آماده در سطح اینترنت وجود دارد که می‌توان از آن‌ها برای شناسایی انواع وب‌شل‌های شناخته شده بهره برد. در ادامه برخی از این ابزارها به عنوان نمونه شرح داده شده است.

۲.۳.۳.۱ <ابزار PHP Malware Finder>

این ابزار قابلیت شناسایی فایل‌های PHP مبهم‌سازی شده[1] و هم‌چنین فایل‌هایی که از توابع مشکوک (که در توسعه‌ی وب‌شل‌ها متداول است) استفاده کرده‌اند را دارد. دقت داشته باشید که فایل‌هایی که توسط این ابزار، مشکوک به بدافزار شناسایی می‌شوند را باید قبل از هر اقدامی به دقت بررسی کنید. ابزار PHP Malware Finder را از آدرس زیر می‌توانید دانلود کنید:

https://github.com/nbs-system/php-malware-finder

۲.۳.۳.۲ <ابزار Web Shell Detector>

این ابزار با دو زبان PHP و Python پیاده‌سازی شده و توانایی شناسایی انواع وب‌شل‌های PHP، ASP، ASPX و Perl را با استفاده از دیتابیس کاملی از Signatureها دارا می‌باشد. از آدرس زیر می‌توانید این ابزار را دریافت کنید:

http://www.shelldetector.com

۲.۳.۳.۳ <ابزار Linux Malware Detect (LMD)>

ابزار LMD توانایی شناسایی انواع بدافزار و از جمله وب‌شل‌ها در سیستم‌عامل لینوکس را دارد. این ابزار از آدرس زیر قابل دسترس است:

https://www.rfxn.com/projects/linux-malware-detect/

۲.۳.۳.۴ <اسکریپت Invoke-Ex-changeWebShellHunter>

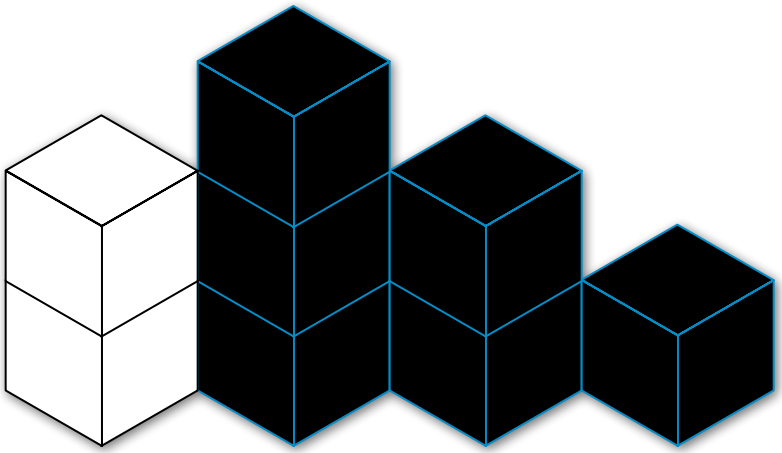
این اسکریپت به طور خاص برای شناسایی وب‌شل‌ها در سرور Microsoft Exchange و با استفاده از PowerShell توسعه داده شده است. این اسکریپت را از آدرس زیر می‌توانید دانلود کنید:

https://github.com/FixTheExchange/Invoke-ExchangeWebShellHunter

---

۱. Obfuscated

۲.۴ <شناسایی با استفاده از تحلیل‌های آماری>

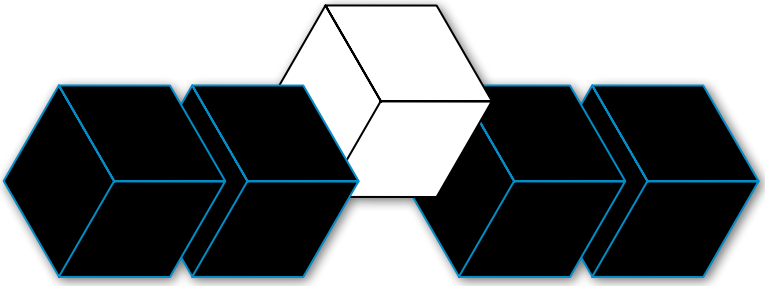


همان‌گونه که در بخش‌های قبل شرح داده شد، بسیاری از وب‌شل‌ها فایل‌های خود را به صورت رمزنگاری شده یا مبهم‌سازی شده ذخیره می‌کنند. مهم‌ترین هدف آن‌ها از این کار را نیز می‌توان دشوارتر کردن شناسایی و تحلیل این بدافزارها دانست. ما می‌توانیم از این ویژگی نیز برای شناسایی فایل‌های مشکوک در سرور وب سازمان استفاده کنیم.

به عنوان مثال ابزار NeoPI که با استفاده از زبان Python توسعه داده شده است، با استفاده از روش‌های مختلف آماری، اسکریپت‌ها و فایل‌های Text رمزنگاری یا مبهم‌سازی شده را با دقت خوبی شناسایی می‌کند. در واقع با استفاده از این ابزار و ابزارهای مشابه دیگر، می‌توان وب‌شل‌های پنهان شده در فایل‌های سامانه‌ی وب را نیز شناسایی کرد. NeoPI فایل‌های سیستم‌عامل را بررسی کرده و به آن‌ها با توجه به قواعد خود، یک نمره اختصاص می‌دهد که بر اساس آن می‌توان فایل‌های مشکوک را شناسایی و بررسی کرد. این ابزار را از آدرس زیر می‌توانید دانلود کنید:

https://github.com/CiscoCXSecurity/NeoPI

۲.۵ <جریان‌های شبکه‌ی (Network Flows) غیر منتظره>



در برخی موارد ، مهاجمین از وب‌شل در سامانه‌هایی غیر از سرورهای وب (به عنوان مثال ایستگاه‌های کاری) استفاده می‌کنند. این وب‌شل‌ها بر روی سرویس‌های وبی که خود مهاجمین بر روی رایانه‌ی قربانی نصب کرده‌اند کار می‌کند و می‌تواند با اجرا شدن در حافظه‌ی RAM (یعنی اجرای بدون فایل یا همان Fileless) راهکارهای شناسایی مبتنی بر فایل را دور بزنند.

گرچه این روش از نظر عملکرد مشابه یک تروجان یا RAT سنتی است، اما به مهاجمین این امکان را می‌دهد تا با استفاده از ترافیک وب با بدافزارهای خود در ارتباط باشند. این نوع وب‌شل‌ها را می‌توان در شبکه‌های به خوبی مدیریت شده، به راحتی شناسایی کرد زیرا آن‌ها از پورت‌هایی که در گذشته استفاده نمی‌شده است برای دریافت و ارسال ترافیک بهره می‌برند. علاوه بر آن، اگر مهاجم از یک سرور وب مرزی برای تانل کردن ترافیک به شبکه استفاده کند، Connectionهایی از یک سرور مرزی شبکه به یک رایانه‌ی داخلی برقرار می‌شود. اگر مدیران شبکه بدانند کدام گره‌های شبکه‌ی آن‌ها به عنوان سرور وب عمل می‌کند، آن‌گاه یک تحلیل ساده‌ی شبکه می‌تواند این جریان‌های غیر منتظره را شناسایی کند.

با استفاده از ابزارهای مختلف، از جمله اسکنرهای آسیب‌پذیری (به عنوان مثال Nes–sus و OpenVAS)، سامانه‌های IDS (مانند

Snort و سایر برندهای تجاری) و ابزارهای مانیتورینگ امنیتی شبکه (مانند Zeek) می‌توان وجود سرورهای وب غیر مجاز در شبکه را کشف کرد. در واقع با ایجاد و نمایش تصویر فعالیت‌های مورد انتظار و عادی در شبکه‌ی سازمان، می‌توان بسیاری از انواع حملات سایبری را شناسایی و با آن‌ها مقابله کرد. رول‌های Snort که در بخش بعد به عنوان نمونه ارایه شده است، می‌تواند برخی از جریان‌های غیر منتظره شبکه را شناسایی کند.

۲.۵.۱ <شناسایی جریان غیرمنتظره‌ی شبکه>

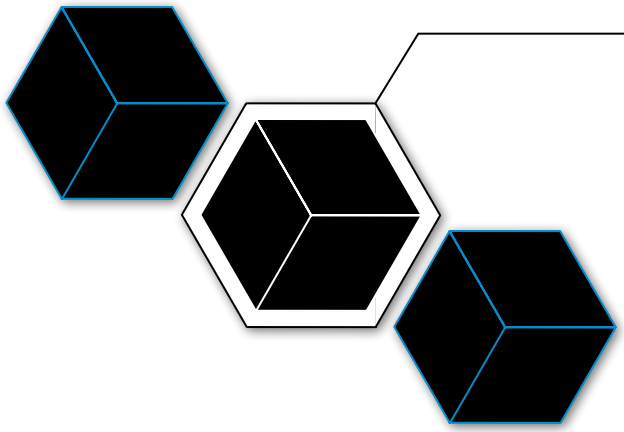
رول Snort که به عنوان نمونه در شکل «۷» ارایه شده است می‌تواند به مدیران شبکه در شناسایی Net-Flowهای غیر منتظره کمک کند. برای شناسایی Flowهای غیر معمول در شبکه، نیاز است تا مدیران شبکه درک عمیق و دقیقی از معماری و جریان‌های موجود در شبکه‌ی سازمان خود داشته باشند. در غیر این‌صورت رول‌هایی مانند این بی اثر یا کم اثر خواهد بود.

```
alert tcp XXX.XXX.XXX.XXX/XX [443,80] -> any any (msg: “potential unex-
pected web server”; sid:4000921)
```

شکل ۷: نمونه رول Snort با هدف شناسایی جریان ترافیکی غیر منتظره

به جای عبارت «XXX.XXX.XXX.XXX/XX» رنج آدرس زیرشبکه‌ی مورد نظر خود (مانند 172.16.0.0/24) را جایگزین کرده و سپس آن را در Snort اضافه کنید.

۲.۶ <شناسایی با استفاده از محصولات EDR و تحلیل لاگ‌های سیستمی>



برخی از EDR[1]ها و راهکارهای Logging پیشرفته می‌توانند وب‌شل‌ها را براساس روش‌هایی مانند تحلیل فراخوانی‌های سیستمی (Syscall) یا ناهنجاری‌های موجود در رفتار پروسس‌ها شناسایی کنند. این محصولات امنیتی تمام پروسس‌ها را در ایستگاه‌های کاری کنترل و پایش می‌کنند. از طرفی وب‌شل‌ها اغلب باعث می‌شوند تا سرور وب رفتارهای غیر عادی از خود بروز دهد. به عنوان مثال، برای یک سرور وب این رفتار که ابزار ipconfig خود را فراخوانی کنند غیر عادی است و این یک روش معمول جمع‌آوری اطلاعات است که توسط وب‌شل‌ها اجرا می‌شود.

EDRها قابلیت‌های خودکار مختلف و هم‌چنین رابط‌های کاربری برای نوشتن Queryدارند، بنابراین سازمان‌ها تشویق می‌شوند که اسناد مرتبط با تشخیص وب‌شل‌ها را بررسی و آن‌ها را با مشاوره‌ی تامین‌کنندگان خود پیاده‌سازی کنند. در بخش‌های بعد شرح داده شده است که چگونه می‌توان از فرایند پیشرفته‌ی لاگ‌گیری ابزار Sysmon، برای شناسایی پروسس‌های غیر عادی در محیط‌های ویندوزی استفاده کرد. هم‌چنین، در بخش دیگری شرح داده‌ایم که چگونه می‌توان از Auditd برای شناسایی فرایندهای غیر عادی در محیط‌های لینوکسی بهره برد.

علاوه بر موارد ذکر شده، بسته به رفتار مهاجمین، شاخص‌های شناسایی دیگری را نیز می‌توان برای شناسایی رفتارهای غیر عادی در ترافیک شبکه در نظر گرفت. به عنوان مثال، بسته‌های Response که به صورت غیر عادی

۱. Endpoint Detection and Response

بزرگ هستند (امکان Exfiltration داده‌ها)، دسترسی به منابع شبکه در ساعات غیر کاری (احتمال فعالیت غیر مجاز در غیر از ساعات کار محلی) و در نهایت افزایش بسته‌های Request از نقاط متفاوت جغرافیایی می‌تواند نشان دهنده‌ی URI مربوط به وب‌شل‌های بالقوه باشد.

البته باید توجه داشت که این نتیجه‌گیری‌ها تا حد زیادی به ویژگی‌های محیط و شرایط هر سازمان بستگی دارد و در نتیجه استفاده از آن‌ها بدون در نظر گرفتن شرایط، به احتمال زیاد بسیاری از URIهای معمول سازمان را نیز به عنوان URI غیر عادی شناسایی خواهد کرد.

۲.۶.۱ <شناسایی پروسس‌های غیر عادی در داده‌های Sysmon>

Sysmon یک ابزار پیشرفته‌ی لاگ‌گیری از ویندوز می‌باشد که توسط شرکت مایکروسافت ارایه شده است. در بین تمام موارد، Sys–mon از اطلاعات مربوط به ایجاد هر پروسس در ویندوز نیز لاگ می‌گیرد. این اطلاعات برای شناسایی رفتارهای ناهنجار مانند وب‌شل‌های مخرب می‌تواند بسیار مفید باشد. فایل Sys–mon را می‌توانید از آدرس زیر دریافت و روی سامانه‌های خود نصب کنید:

https://docs.microsoft.com/en–us/sysinternals/downloads/

بهتر است که لاگ‌های تولید شده توسط Sysmon و سایر لاگ‌های سیستم‌عامل ویندوز به یک سامانه‌ی SIEM مرکزی ارسال شود تا در آن‌جا به صورت یک‌پارچه بتوان از آن در تحلیل‌های امنیتی استفاده کرد. به عنوان مثال عبارت Query زیر به‌سادگی نشان می‌دهد که چه فایل‌های اجرایی توسط سرور وب IIS اجرا شده‌اند. کافیست اسکریپت زیر را در محیط PowerShell با سطح دسترسی Administrator اجرا کنید.

```
Get-WinEvent -FilterHashtable @{logname=”Microsoft-Windows-Sysmon/Opera-
tional”;id=1;} |

Where {$_.message -like “*ParentImage: C:\Windows\System32\inetsrv\w3wp.
exe*”} |

%{ $_.properties[4]} | Sort-Object -Property value -Unique
```

شکل ۸: اسکریپت PowerShell با هدف شناسایی فایل‌های اجرا شده توسط IIS

در بسیاری از موارد، یک وب اپلیکیشن می‌تواند سبب شود تا سرور IIS یک پروسس بی‌خطر را اجرا کند. اما برخی از پروسس‌ها کمتر توسط سرورهای IIS یا وب اپلیکیشن‌ها فراخوانی می‌شود، بلکه اغلب برای مهاجمین در راستای جمع‌آوری اطلاعات کاربرد دارند. برخی از مهم‌ترین این پروسس‌ها در جدول «۵» نمایش داده شده است. مدیران شبکه می‌توانند با استفاده از اسکریپت بالا بررسی کنند که سرور IIS مورد نظر آن‌ها چه پروسس‌هایی را اجرا کرده و آن را با جدول «۵» مقایسه کنند. این تحلیل ممکن است توسط یک سامانه‌ی EDR یا به صورت دستی و با استفاده از اسکریپت‌های مشابه شکل «۸» انجام شود.

جدول ۵: فایل‌ها و دستورات مشکوک و مورد علاقه‌ی مهاجمین در سرورهای ویندوزی

schtasks.exe	ntdutil.exe	hostname.exe	arp.exe
systeminfo.exe	pathping.exe	ipconfig.exe	at.exe
tasklist.exe	ping.exe	nbtstat.exe	bitsadmin.exe
tracert.exe	powershell.exe	net.exe	certutil.exe
ver.exe	qprocess.exe	net1.exe	cmd.exe
vssadmin.exe	query.exe	netdom.exe	dsget.exe
wevtutil.exe	qwinsta.exe	netsh.exe	dsquery.exe
whoami.exe	reg.exe	netstat.exe	find.exe
wmic.exe	rundll32.exe	nltest.exe	findstr.exe
wusa.exe	sc.exe	nslookup.exe	fsutil.exe

۲.۶.۲ <شناسایی پروسس‌های غیر معمول لینوکس با Auditd>

Auditd یکی از اجزای Auditing System برای فضای کاربری لینوکس است. Auditd می‌تواند به کاربران لینوکس امکان بررسی لاگ ایجاد پروسس‌ها را بدهد. مشابه بخش قبل، این اطلاعات برای شناسایی ناهنجاری‌هایی مانند وب‌شل‌های مخرب می‌تواند مفید باشد. Auditd اغلب در سرورهای لینوکس به صورت پیش‌فرض نصب است و تنها کافیست پیکربندی اولیه بر روی آن انجام شود، تا از اطلاعات مربوط به پروسس‌های سرور وب مربوطه لاگ بگیرد.

توصیه می‌شود این لاگ‌ها نیز به همراه سایر لاگ‌های تولید شده در سیستم‌عامل‌های لینوکس به یک سامانه‌ی مرکزی SIEM ارسال شود. با استفاده از دستورالعمل زیر می‌توانید گزارش بگیرید که چه برنامه‌هایی توسط سرور وب APACHE اجرا شده‌اند.

گام اول: پیکربندی Auditd

۱. مقدار uid سرور وب را پس از نصب Auditd با استفاده از دستور زیر مشخص کنید:

```
apachectl -S
```

خروجی این دستور مشابه مثال زیر خواهد بود:

```
User: name=”www-data” id=33
```

در این مثال مقدار uid برابر است با ۳۳.

البته uid سرور آپاچی را با استفاده از دستورات زیر نیز می‌توان به دست آورد:

```
ps aux | egrep ‘([a|A]pache|[h|H]ttpd)’ | awk ‘{ print $1}’ | uniq | tail -1
```

```
id -u <username>
```

۲. رول‌های زیر را به فایل /etc/audit/rules.d/audit.rules اضافه و مقدار XX را با مقدار uid که از دستور بالا به دست آورده‌اید جایگزین کنید.

```
-a always,exit -F arch=b32 -F uid=XX -S execve -k apacheexecve
```

```
-a always,exit -F arch=b64 -F uid=XX -S execve -k apacheexecve
```

۳. با استفاده از دستور زیر، سرویس Auditd را ریست کنید:

```
service auditd restart
```

گام دوم: بازبینی لاگ‌های تولید شده توسط Auditd

۱. برنامه‌های اجرا شده توسط APACHE را می‌توان با استفاده از دستور زیر دریافت کرد:

```
cat /var/log/auditd/audit.* | grep “apacheexecve”
```

این دستور مسیر فایل‌های اجرا شده را نمایش می‌دهد، مانند مسیر پررنگ شده در مثال زیر:

```
ype=SYSCALL msg=audit(1581519503.841:47): arch=c000003e syscall=59
success=yes exit=0 a0=563e412cbbd8 a1=563e412cbb60 a2=563e412cbb78
a3=7f065d5e5810 items=2 ppid=15483 pid=15484 auid=4294967295 uid=33
gid=33 euid=33 suid=33 fsuid=33 egid=33 sgid=33 fsgid=33 tty=(none)
ses=4294967295 comm=”cat” exe=”/bin/cat” key=”apacheexecve
```

۲. نتایج به دست آمده می‌تواند با جدولی مشابه جدول «۶» مقایسه شود تا ببینیم آیا برنامه یا دستور غیر عادی شناسایی می‌شود یا خیر.

۳. در صورت نیاز اطلاعات جزیی‌تری از برنامه‌ها مانند آرگومان‌های فراخوانی را نیز می‌توان با دستور زیر به دست آورد:

```
cat /var/log/auditd/audit.* | grep “msg=audit(1581519503.841:47)”
```

مقدار «msg=audit» در دستور بالا را با مقدار به دست آمده از دستور شماره‌ی «۱» جایگزین کنید.

در این‌جا نیز مشابه بخش قبل، یک وب اپلیکیشن می‌تواند سبب شود تا سرور APACHE یک برنامه‌ی بی‌خطر را اجرا کند. اما برخی از پروسس‌ها کمتر توسط سرورهای APACHE یا وب اپلیکیشن‌ها فراخوانی می‌شوند، بلکه اغلب برای مهاجمین در راستای جمع‌آوری اطلاعات کاربرد دارند. برخی از مهم‌ترین این دستورات در جدول «۶» نمایش داده شده است که می‌تواند برای مقایسه با نتایج حاصل از اجرای دستورات بالا کاربرد داشته باشد.

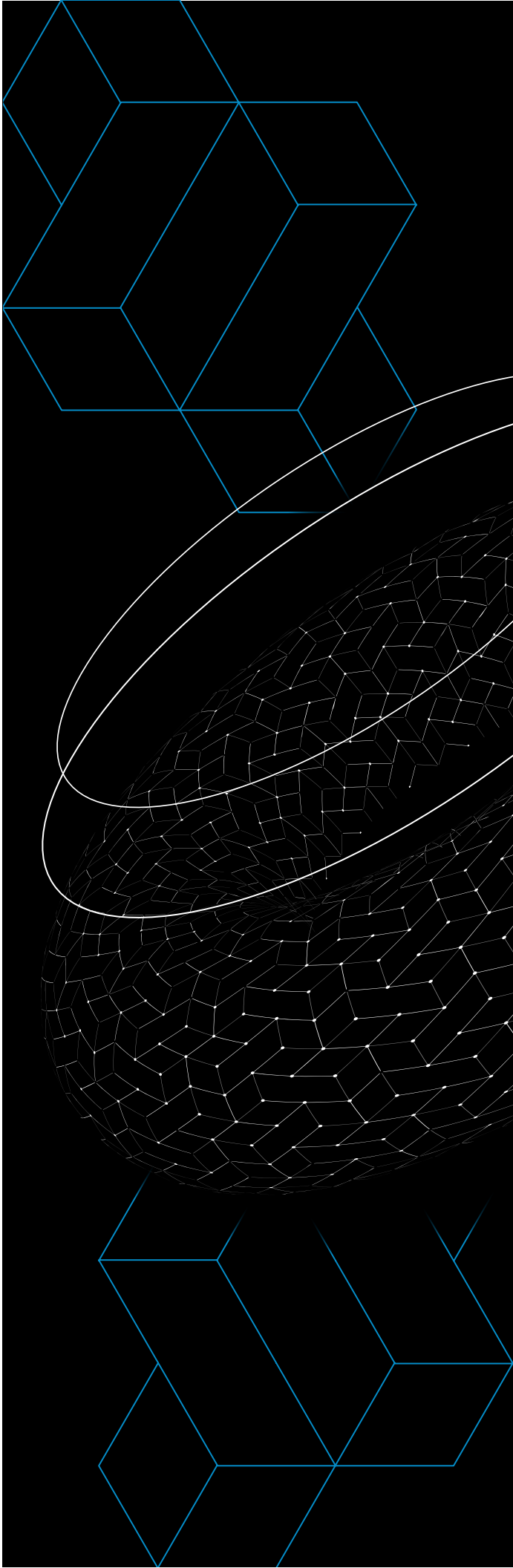
جدول ۶: فایل‌ها و دستورات مشکوک و مورد علاقه‌ی مهاجمین در سرورهای لینوکس

route	ls	ifconfig	cat
uname	netstat	ip	crontab
whoami	pwd	iptables	hostname

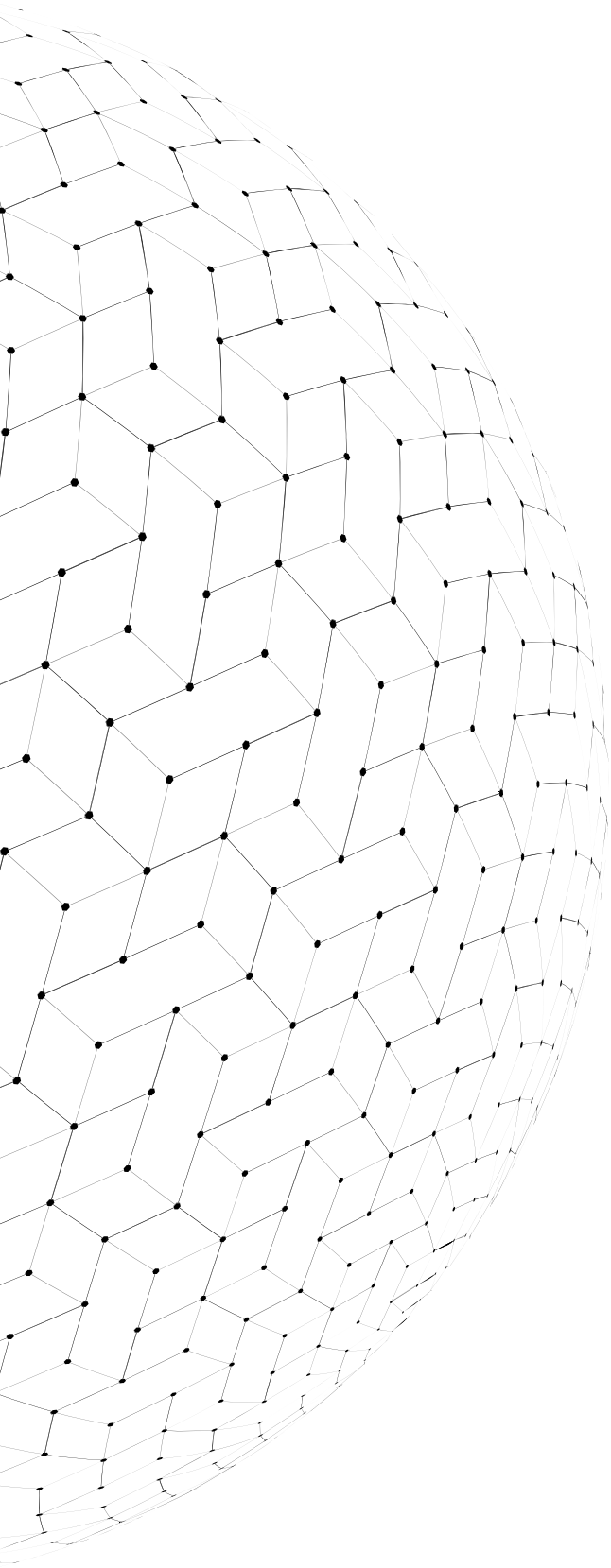
۳ <پاسخ به تهدید وب‌شل و بازیابی امنیت>

علیرغم این‌که بعضی از وب‌شل‌ها ماندگاری ندارند و تنها برای مدت محدود در حافظه‌ی RAM اجرا می‌شوند و برخی دیگر نیز به صورت فایل باینری یا فایل اسکریپت ساده ذخیره می‌شوند، اما بسیاری از آن‌ها با استفاده از مکانیزم‌های پیچیده و با هدف ماندگاری (Persistence) بر روی سرور قربانی نصب می‌شوند. هم‌چنین بسیاری از وب‌شل‌ها می‌توانند بخشی از یک حمله‌ی گسترده‌تر باشند که عمق زیرساخت سازمان را هدف گرفته است.

نکته‌ی حیاتی در مورد وب‌شل‌ها این است که به محض کشف شدن باید بررسی‌های اولیه در جهت این باشد که مهاجم تا کجا در داخل شبکه نفوذ کرده است. روش‌هایی مانند تحلیل ترافیک و لاگ می‌تواند در شناسایی این‌که وب‌شل‌ها چگونه برای نفوذ به زیرساخت سازمان و این‌که به چه سامانه‌های داخلی نفوذ شده است کمک کند. اگر مدرکی مبنی بر نفوذ بیشتر مهاجمین پیدا نشد و وب‌شل را از بین بردید باید آمادگی داشته باشید که مهاجمین بلافاصله یا در زمان دیگری برای گرفتن دسترسی مجدد به زیرساخت شما از کانال‌های دیگر اقدام کنند. بنابراین لازم است تا در کوتاه‌ترین زمان ممکن راهکارهای شناسایی و شکار تهدیدات سایبری در سازمان خود را پیاده‌سازی کرده یا بهبود دهید.



۴ <جمع‌بندی>



در این کتابچه روش‌های مختلفی برای شناسایی و شکار وب‌شل‌های موجود در زیرساخت سازمان ارایه شد. اغلب این روش‌ها مبتنی بر بررسی ویژگی‌های ذاتی وب‌شل‌ها برای شکار آن‌ها و هم‌چنین IOCهای مربوط به وب‌شل‌های شناخته شده هستند.

سعی بر آن بوده تا جای ممکن، روش‌ها و رول‌های پرکاربرد در راستای شناسایی طیف گسترده‌ای از وب‌شل‌ها به صورت یکپارچه ارایه شود. اما در حالت کلی وب‌شل‌ها نیز نوعی بدافزار هستند که از روش‌های مختلفی برای پنهان کردن خود و فعالیتشان استفاده می‌کنند.

در طول زمان مهاجمین روش‌های خود را در صورت شناسایی شدن، تغییر می‌دهند به‌گونه‌ای که این نبرد همواره بین تحلیل‌گران امنیت سایبری و مهاجمین برقرار خواهد بود. بنابراین نیاز است تا علاوه بر موارد ذکر شده در این کتاب‌چه، با توجه به زیرساخت و دارایی‌های سازمان، همواره دانش خود را به روز نگه دارید تا از تکنیک‌های جدید مورد استفاده‌ی مهاجمین و روش‌های شکار و مقابله با آن‌ها آگاه باشید.

۵ <منابع>

https://medium.com/swlh/web-shell-hunting-meet-the-web-shell-analyzer-f062686b443b

https://www.rsa.com/content/dam/en/solution-brief/asoc-threat-solution-series-webshells.pdf

https://github.com/nsacyber/Mitigating-Web-Shells

https://blog.rapid7.com/2016/12/14/webshells-101

https://www.tenable.com/blog/hunting-for-web-shells

https://github.com/Neo23x0/Loki

https://github.com/nbs-system/phpmalware-finder

https://www.rfxn.com/projects/linuxmalware-detect

https://github.com/FixTheExchange/Invoke-ExchangeWebShellHunter

http://www.shelldetector.com

https://github.com/CiscoCXSecurity/NeoPI

https://docs.microsoft.com/en-us/sysinternals/downloads

https://github.com/DarkenCode/yara-rules/blob/master/malware/Web-shell-shell.yar

آکادمی راوین

www.ravinacademy.com

www.afta.gov.ir