SYNful Knock (بدافزاری مختص مسیریاب‌ها)

۱. مقدمه

دسته‌ای از حملات که به طور گسترده توسط نفوذگرها به منظور ایجاد وقفه در عملکرد دستگاه‌ها و تجهیزات درون شبکه صورت می‌گیرد، حملات منع‌سرویس می‌باشند. این حملات از متداول‌ترین حملاتی هستند که تجهیزات یک شبکه را تهدید می‌کنند و هر روز حملات پیشرفته‌تری از این نوع به منظور شنود یا ایجاد راهی برای تسخیر تجهیزات شبکه و در دست گرفتن کنترل آن‌ها دیده می‌شود.

اخیراً تیم پاسخگویی به بحران‌های امنیتی تجهیزات سیسکو PSIRT[1] دسته جدیدی از حملات را گزارش داده است که پلتفرم‌های نرم‌افزاری IOS سیسکو را مورد هدف قرار داده‌اند. این گزارش که منبع اصلی آن شرکت Mandiant/FireEye است، دسته‌ای از IOS های روترهای دستکاری شده سیسکو را معرفی می‌کند که ۱۴ نمونه از آن در چهار کشور مکزیک، هند، فیلیپین و اوکراین مشاهده شده است. این نمونه‌های دستکاری شده شامل نسخه‌ای از سیستم‌عامل‌های سیسکو است که به فرد حمله‌گر این قابلیت را می‌دهد تا از طریق دسترسی نامحدود به روترها، بتواند ماژول‌های دیگر را بر روی روتر تسخیر شده بارگذاری کند. این حمله که تحت عنوان SYNful Knock شناخته شده است، سریِ ۱۸۴۱، ۲۸۱۱ و ۳۸۲۵ از روترهای سیسکو را مورد حمله قرار داده است ولی قابلیت تاثیرگذاری بر روی سایر مدل روترهای سیسکو را دارد.

۲. SYNful Knock چیست؟

SYNful Knock یک درب پشتی کار گذاشته شده در نسخه دستکاری شده نرم‌افزار IOS سیسکو است که دسترسی دائم به شبکه هدف را میسر می‌سازد. این بدافزار ماهیتی ماژولار داشته و درنتیجه امکان سفارشی‌سازی عملکرد آن وجود دارد. برخی از نکات مهم و تهدیدات این بدافزار به شرح زیر است:

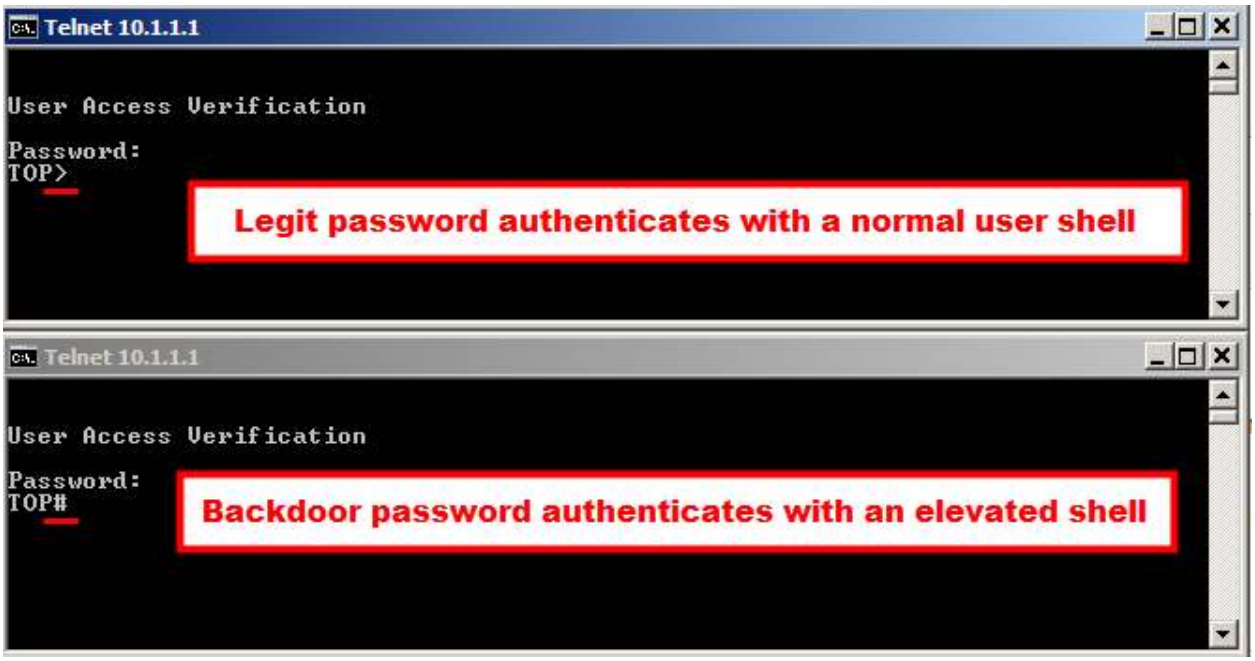
۱. به حمله‌گر این اجازه را می‌دهد تا ماژول‌های عملیاتی مختلفی را از طریق اینترنت بر روی سیستم‌عامل نصب کند.

[1] Cisco Product Security Incident Response Team

۲. با ایجاد درب‌پشتی (backdoor) بر روی دستگاه، دسترسی نامحدود فراهم می‌نماید.
۳. ماژول‌ها را از طریق HTTP نه HTTPS دریافت می‌کند.
۴. حمله‌گر بسته‌های TCP را دستکاری می‌کند تا دارای شماره Sequence غیراستانداردی باشند.
۵. با ایجاد یک درب پشتی، می‌تواند از طریق کنسول و یا Telnet دوباره به سیستم عامل دسترسی پیدا کند.

این بدافزار پس از استقرار در سیستم‌عامل روتر، یک درب پشتی[۲] ایجاد می‌کند و برای ارتباط مجدد از طریق این درب پشتی از یک کلمه عبور محرمانه استفاده می‌کند.

نسخه دستکاری شده سیستم‌عامل به‌وسیله بدافزار، در ابتدای کار بررسی می‌کند که آیا ورودی کاربر، پسورد در نظر گرفته‌شده برای backdoor است یا خیر. اگر این طور بود، دسترسی ویژه در اختیار کابر قرار می‌دهد و در غیر این صورت اطلاعات ورودی کاربر را به سیستم‌عامل اصلی می‌دهد تا به طور عادی کاربر احراز هویت شود. دو شکل زیر این تفاوت عملکرد را نشان می‌دهند. در حالت اول یک کاربر عادی با وارد نمودن پسورد خود به محیط دستوری وارد می‌شود ولی در حالت دوم فرد حمله‌گر با وارد نمودن پسوردی که برای ارتباط از طریق درب‌پشتی در نظر گرفته است، وارد شده و به شِل[۳] دسترسی پیدا کرده است.



[2] Backdoor
[3] Shell

با استفاده از این تکنیک ادمین‌ها و مسئولین شبکه کم‌ترین میزان تردید در وجود این بدافزار را خواهند داشت. سه نمونه که این موضوع را تصدیق می‌کنند، در شکل زیر نشان داده شده‌اند.

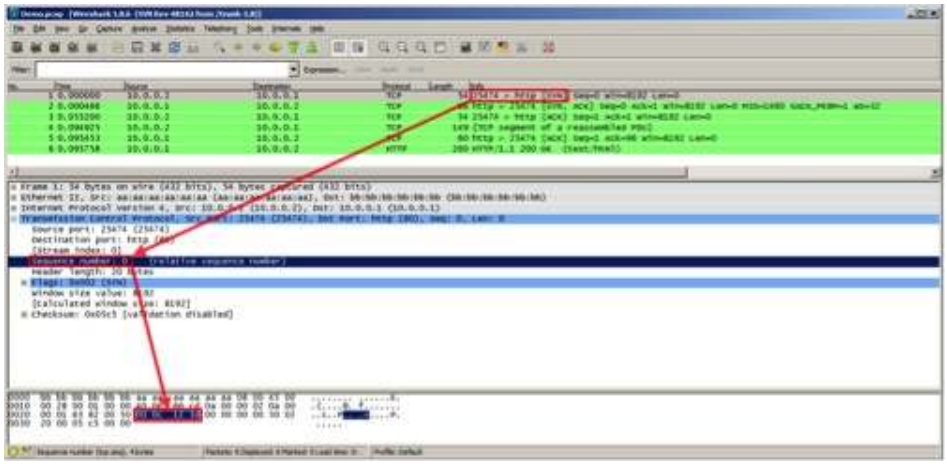
Method	Prompt	Results
Console	"User Access Verification"	Access and elevated session
Telnet	Username is the backdoor password	Access and elevated session
Elevation (enable)	enable password	Elevated session

یک نکته مثبت در مورد حملاتی که تاکنون گزارش شده‌اند، این است که سیستم‌عامل‌های هک‌شده نمی‌توانند از طریق پروتکل‌های HTTPS و SSH با فردحمله‌گر ارتباط برقرار نمایند. پس ادمین‌های شبکه می‌توانند با بلاک‌نمودن دسترسی‌ها بر روی پروتکل‌های رمزنشده مثل telnet از دسترسی‌های بعدی حمله‌گر به سیستم‌عامل جلوگیری کنند. البته باید توجه نمود که به این دلیل که سیستم‌هایی که تاکنون مشاهده شده است، نمی‌توانند بر روی پروتکل‌های رمزشده ارتباطات خود را برقرار کنند، نمی‌توان به این نتیجه رسید که بستن دسترسی بر روی این پروتکل‌ها بهترین راهکار است. چرا که مطمئناً در آینده و نسخه‌های بعدی این سیستم‌عامل دستکاری‌شده، حتماً این قابلیت‌ها نیز به سناریوی حمله افزوده خواهد شد.

۳. ارتباط با سرور C&C از طریق شبکه

ارتباط با سرور C&C به صورت مخفیانه صورت می‌گیرد چرا که بدافزار به دنبال یک دنباله خاص از بسته‌های TCP می‌گردد و هنگامی که مقادیر خاصی را در سرآیند بسته‌ها مشاهده می‌کند، پروسه خود را آغاز می‌کند. حتی اگر فیلترها بر روی روتر فعال شده باشند، باز هم بدافزار بسته‌های TCP که به منظور شروع عملیات ارتباط با C&C ارسال می‌شوند را پردازش می‌کند. بدافزار، بسته‌ها برای ۳ آدرس را بررسی نموده و به آن‌ها پاسخ می‌دهد. این آدرس‌ها شامل اینترفیسِ روتر، آدرس IP همه‌پخشی شبکه و آدرس شبکه (اولین آدرس در زیرشبکه) هستند.

۱. برای شروع پروسس، یک بسته TCP SYN دست‌کاری شده برای پورت ۸۰ روتر آلوده ارسال می‌گردد. نکته قابل‌توجه این است که اختلاف بین عدد acknowledgment و sequence در این بسته‌ها باید برابر با 0xC123D باشد. هم‌چنین ACK number نباید صفر باشد.



۲. مشابه با تمامی پروتکل‌های موجود در TCP که هنگام شروع ارتباط یک 3way handshaking برقرار می‌کنند، این روتر که توسط SYNful Knock دستکاری شده است، به بسته SYN دریافتی اولیه، یک بسته TCP SYN-ACK پاسخ می‌دهد، ولی با شرایط زیر:

- اختلاف بین عددهای acknowledgement و sequence برابر با 0xC123E است.
- در فیلد Option این مقادیر هگز قرار داده شده است: "02 04 05 b4 01 01 04 02 01 03 03 05"
- Urgent Pointer بر روی مقدار 0x0001 تنظیم شده ولی urgent flag برابر صفر است.
- هم‌چنین بدافزار عدد Sequence این بسته را برابر با عدد acknowledgment مربوط به بسته SYN قبلی قرار می‌دهد. در حالت عادی سرور یک عدد تصادفی ایجاد می‌کند و در این فیلد قرار می‌دهد.

که تمامی این‌ها حاکی از این است که این یک 3-way handshaking ساده نیست و با استفاده از این نشانه‌ها شرکت mandiant ابزارها و قوانین شناسایی این بدافزار در شبکه را توسعه داده‌اند.

۳. پس از پایان یافتن 3-way handshaking از طریق دریافت بسته Ack، کنترل‌کننده دنباله پیام TCP با شناسه زیر را ارسال می‌کند:

- پرچم‌های PUSH و ACK با مقدار ۱
- در شروع سرآیند بسته TCP، در آفست 0x62 رشته ’text‘ نوشته شده است.
- هم‌چنین در آفست 0x67 از سرآیند بسته، دستور زیر تنظیم شده است:

```
[4 byte Command Length][CMD Data][4 byte checksum]
```

پاسخ بدافزار در فرمت HTTP/HTML زیر بسته‌بندی شده که کاملاً واضح است که هیچ مکانیزم رمزنگاری‌ای در آن در نظر گرفته نشده است و به راحتی قابل مشاهده است:

```
HTTP/1.1 200 OK
Server: Apache/2.2.17 (Ubuntu)
X-Powered-By: PHP/5.3.5-1ubuntu7.7
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
<html><body><div>[Response]</div></body></html>
```

۳-۱. دستورات قابل پشتیبانی

این بدافزار ماژولار بوده و با استفاده از ۵ دستور، روتری که بدافزار SYNful Knock بر روی آن مستقر شده است، ماژول‌ها و قابلیت‌های بیشتری را بر روی خود بارگذاری می‌نماید. این بدافزار می‌تواند در حدود ۱۰۰ ماژول اضافی بر روی روتر بارگذاری کند ولی این ماژول‌ها همگی بر روی حافظه ذخیره شده و پس از ریستارت نمودن و یا بارگذاری مجدد دیگر قابل دسترس نیستند. تمامی پیام‌های دستوری با ۸ بایت زیر شروع می‌شوند که ۴ بایت اول همگی صفر بوده و ۴ بایت دوم نوع پیام را معین می‌کنند. این متد در جدول زیر نشان داده شده است.

ID	توضیحات
0	تمامی ماژول‌ها و وضعیت فعلی آن‌ها را لیست می‌کند. پاسخ دریافتی در این حالت شامل چهار بایت بوده که نشان‌دهنده شماره id است و چهار بایت دیگر در ادامه، نشان‌دهنده وضعیت هرکدام از ماژول‌های بارگذاری شده است. وضعیت‌ها به صورت زیر هستند: ○ حافظه اختصاص داده شده است. ○ ماژول درون حافظه بارگذاری شده است. ○ ماژول فعال است. به طور مثال اگر بدافزار به صورت 02 00 00 03 00 00 00 به این پیام پاسخ دهد، نشان‌دهنده این است

که ماژول ۳ در وضعیت فعال قرار دارد.	
حافظه بیش‌تری برای اختصاص دادن به سایر ماژول‌ها در نظر می‌گیرد. این دستور ۲ حافظه بافر را برای ماژول‌ها اختصاص می‌دهد و آدرس‌های این بافرها را در پاسخ ارسال می‌کند. اولین بافر برای کد اجرایی است و احتمالاً بافر دوم نیز برای ذخیره‌سازی و تنظیمات است. قالب دستورات به فرم زیر است: [WORD ID][WORD first buffer length][WORD second buffer length] به طور مثال یک نمونه که بدافزار 0x0c بایت برای بافر اول و 0x90 بایت را برای بافر دوم با ماژول ID برابر با 0x02 اختصاص می‌دهد، به صورت زیر است: 00 00 00 02 00 00 00 0C 00 00 00 90 یک نمونه پاسخ سرور که نشان می‌دهد بافر اول در آدرس حافظه 0x6012c4c و دومی در 0x650dcd20 است به صورت زیر است: 66 01 2C 4C 65 0D CD 20 پس از اجرای این دستور، وضعیت ماژول در حالت صفر قرار می‌گیرد.	1
این دستور برای مستقرسازی ماژول درون حافظه‌های اختصاص داده شده استفاده می‌گردد. با استفاده از این دستور کد اجرایی و داده‌های پیکربندی در حافظه قرار می‌گیرند: [0x80 Bytes hook data][WORD first buffer length][WORD second buffer length] [First buffer...][Second buffer...] با استفاده از این دستور فایل‌های مخرب بیشتری درون سیستم‌عامل قرار داده می‌شوند. پس از اجرای این دستور ماژول در وضعیت ۱ قرار می‌گیرد.	2
یک ماژول بارگذاری‌شده را فعال می‌کند و تنها به عنوان ورودی یک word (که معادل چهار بایت است) را	3

به عنوان ورودی برای شناسه ماژول دریافت می‌کند. پس از اجرای این دستور، ماژول در وضعیت ۲ قرار خواهد گرفت.	
حذف کردن یک ماژول (با اجرای این دستور، حافظه‌ای که برای ماژول اختصاص داده شده است، آزاد شده و وضعیت به حالت ۰ تغییر خواهد نمود و ماژول دیگر به صورت فعال نخواهد بود).ی ک	4

اگر چهار بایت اول پیام صفر نباشند، کد اجرا خواهد شد با این تفاوت که دیگر جزء دستورات مربوط به بدافزار نبوده و دستورات خود سیستم‌عامل اجرا می‌شوند (نه دستورات سیستم‌عامل دست‌کاری‌شده).

۴. روش‌های شناسایی

به‌طور کلی راه‌کارهای موجود برای شناسایی این بدافزار را می‌توان به دو دسته کلی زیر تقسیم‌بندی نمود:

- تشخیص مبتنی بر میزبان: این روش برای سازمان‌هایی مناسب است که می‌توانند دستورات را اجرا نموده و پاسخ آن‌ها را مشاهده کنند. این روش تنها برای روترهایی که در یک شبکه کوچک و در دسترس قرار دارند، امکان‌پذیر است.
- تشخیص مبتنی بر شبکه: برای سازمان‌هایی مناسب است که پراکنده بوده و به راحتی نمی‌توانند دسترسی به روترها داشته و دستورات محلی را اجرا کنند.

به طور کلی ترکیبی از این دو روش می‌تواند بهترین راه‌کار برای شناسایی روترهایی باشد که به بدافزار SYNful Knock آلوده شده‌اند.

۴-۱. تشخیص مبتنی بر میزبان

اگر امکان دسترسی به محیط Command-line روتر مورد نظر امکان‌پذیر باشد، با استفاده از دستور show می‌توان اطلاعات مفیدی به‌دست آورد:

“show platform | include RO, Valid”

معمولاً در روترهای تسخیرشده، با اجرای این دستور هیچ نتیجه‌ای مشاهده نمی‌شود.

یک نکته قابل توجه دیگر این است که در نسخه‌های تسخیرشده سیستم عامل، اندازه فایل باینری سیستم‌عامل برابر با همان اندازه سیستم عامل اصلی است. پس از این طریق نمی‌توان تفاوتی بین آنها قائل شد. ولی با گرفتن مقدار درهم (Hash) از هرکدام و مقایسه مقدار درهم محاسبه شده برای سیستم‌عاملی که مشکوک به SYNful Knock است و سیستم‌عامل اصلی سیسکو می‌توان به راحتی سیستم‌عامل دستکاری شده را شناسایی نمود چرا که مقادیر درهم آن‌ها با یکدیگر متفاوت است. ولی این راه‌کار تنها برای زمانی کاربرد دارد که فایل سیستم‌عامل در دسترس باشد و برای حالتی که بر روی دستگاه بارگذاری شده و در حال اجرا است، کاربردی ندارد.

۴-۲. تشخیص مبتنی بر شبکه

در این متد هر دو روش فعال و غیرفعال (اکتیو یا پسیو) را می‌توان برای تشخیص و جلوگیری از SYNful Knock استفاده نمود که در بخش زیر توضیح داده شده است:

۴-۲-۱. روش پسیو (غیرفعال)

در گزارش ارائه شده از سوی شرکت FireEye، چهار روش برای شناسایی پسیو رفتار درون شبکه‌ای بدافزار و ارتباط آن با سرور C&C[4] ارائه شده است که این ۴ متد به شرح زیر است:

۱. SYN: اولین نشانه‌ای که باید بررسی گردد، signature مربوط به اختلاف عدد acknowledgment و sequence است که در قسمت‌های قبل توضیح داده شد. چرا که هنگامی که روتر قربانی درخواست شروع برقراری ارتباط با سرور فرماندهی و کنترل را دارد، یک بسته syn با نشانه خاص ارسال می‌کند که باید این بسته‌ها شناسایی شوند. کد مربوط به این متد تشخیص در پیوست ۱ آورده شده است. هم‌چنین

[4] Command and Control

برای اینکه خطا تا حد ممکن کاهش پیدا کند، کد به نحوی در نظر گرفته شده تا فقط حالاتی را در نظر بگیرد که عدد acknowledgement بسته‌ها برابر با صفر نباشد.

۲. SYN/ACK: دومین نشانه (signature) که سورس کد آن در پیوست ۲ ارائه شده است، به دنبال بسته‌های SYN/ACK با نشانه‌های مخصوص این بدافزار می‌گردد. در این حالت دیگر بررسی نمی‌کند که آیا acknowledgement number برابر با صفر است یا نه.

۳. signature نشان داده شدهِ زیر پاسخ‌های دریافتی از سرور HTTP را هنگامی که یک دستور از طرف بدافزار ارسال می‌گردد، بررسی می‌کند. مزایت اصلی این signature این است که یک signature استاندارد Snort است. ولی به هر حال قابلیت بررسی اختلاف بین Sequence و Acknowledgment را ندارد.

```
alert tcp any any -> any any (\
msg: "SYNful Knock Cisco Implant HTTP Header";\
flow: from_server;\
content: "HTTP/1.1 200 OK|0d 0a|Server: Apache/2.2.17 (Ubuntu)|0d 0a|X-Powered-By:
PHP/5.3.5-1ubuntu7.7|0d 0a|Keep-Alive: timeout=15, max=100|0d 0a|Connection: Keep-
Alive|0d 0a|Content-Type: text/html|0d 0a 0d 0a|<html><body><div>"; offset:0;\
flags:PA;\
sid:201504232;\
```

۴. دستورات کنترلی: signature زیر دستوری که از طرف مرکز کنترلی بدافزار ارسال شده است را شناسایی می‌کند.

```
alert tcp any any -> any any (\
msg: "SYNful Knock Cisco Implant HTTP Request";\
flow: to_server;\
content: "text"; offset:78; depth:4;\
content: "|00 00 00|"; offset: 83; depth: 3;\
content: "|45 25 6d|"; offset: 87; depth: 3;\
```

```
sid:201504233;\
)
```

۴-۲-۲. روش اکتیو (فعال)

۵. روش اکتیو به این صورت است که یک سری بسته‌های خاص برای روتر ارسال می‌شوند و از طریق پاسخ‌هایی که گرفته می‌شود تشخیص داده شود که آیا روتر دستکاری‌شده است یا خیر. شرکت FireEye ابزارهای زیادی برای این منظور طراحی نموده است. به طور مثال می‌توان به اسکریپت نوشته شده برای موتور Nmap با نام NSE[5] اشاره نمود که با زبان LUA نوشته شده است. این اسکریپت می‌تواند نشان دهد که آیا بدافزار بر روی روتر مستقر شده است یا خیر.

۶. یک راه‌کار دیگر استفاده از اسکریپت نوشته‌شده به زبان پایتون است که این اسکریپت و دستورات مورد نیاز آن از طریق آدرس زیر قابل دسترسی است:

https://github.com/fireeye/synfulknock

این اسکریپت شامل ابزارهایی است که می‌توان نسخه‌های دستکاری‌شده از روترهای سیسکو به بدافزار SYNful Knock را شناسایی کند.

توجه: یک راه ساده برای شناسایی این روترها، استفاده از نرم‌افزار Snort است که یک نرم‌افزار Open-Source می‌باشد. به همین منظور می‌توان از rule زیر (که برای این دسته از حملات که سیستم‌عامل روتر سیسکو را مورد حمله قرار داده و تحت کنترل می‌گیرند) استفاده نمود. کافی است rule زیر را در Snort اضافه نمود:

Snort Rule SID:36054
https://snort.org/advisories/talos-rules-2015-09-15 link:

این rule شامل یک قانون برای شناسایی ارتباطات C&C بدافزار با فردحمله‌گر است که با GID=1 و SID=36054 شناخته شده است.

پیوست۱:

[5] Nmap Scripting Engine

Signature کامپایل‌شده برای نرم‌افزار Snort (شناسایی از طریق بسته‌های TCP SYN)

```
#include "sf_snort_plugin_api.h"
#include "sf_snort_packet.h"
/* declare detection functions */
int rule201504230eval(void *p);
/* declare rule data structures */
/* flow:to_server; */
static FlowFlags rule201504230flow0 =
{
  FLOW_TO_SERVER
};
static RuleOption rule201504230option0 =
{
  OPTION_TYPE_FLOWFLAGS,
  {
    &rule201504230flow0
  }
};
/* references for sid 201504230 */
static RuleReference *rule201504230refs[] =
{
  NULL
};
/* metadata for sid 201504230 */
/* metadata:; */
static RuleMetaData *rule201504230metadata[] =
{
  NULL
};
RuleOption *rule201504230options[] =
{
  &rule201504230option0,
  NULL
};
Rule rule201504230 = {
  /* rule header, akin to => tcp any any -> any any */
  {
    IPPROTO_TCP, /* proto */
    "any", /* SRCIP     */
```

```
    "any", /* SRCPORT   */
    0, /* DIRECTION */
    "any", /* DSTIP     */
    "any", /* DSTPORT   */
  },
  /* metadata */
  {
    3,  /* genid */
    201504230, /* sigid */
    1, /* revision */
    "misc-activity", /* classification */
    0,  /* hardcoded priority */
    "TCP Trigger SEQ SYN",     /* message */
    rule201504230refs, /* ptr to references */
    rule201504230metadata /* ptr to metadata */
  },
  rule201504230options, /* ptr to rule options */
  &rule201504230eval, /* use custom detection function */
  0 /* am I initialized yet? */
};
/* detection functions */
int rule201504230eval(void *p) {
  const u_int8_t *cursor_normal = 0;
  SFSnortPacket *sp = (SFSnortPacket *) p;
  uint32_t seq = 0;
  uint32_t ack = 0;
  if(sp == NULL)
    return RULE_NOMATCH;
  if(sp->payload == NULL)
    return RULE_NOMATCH;
  ack = ntohl(sp->tcp_header->acknowledgement);
  seq= ntohl(sp->tcp_header->sequence);

  if (ack == 0){
    return RULE_NOMATCH;
  }

  //Test for SYN packets
  if((sp->tcp_header->flags & TCPHEADER_SYN)&& !(sp->tcp_header->flags &
TCPHEADER_ACK)){
    if ((ack > seq) && (ack - seq == 0xC123D)){ return RULE_MATCH; }
```

```
    else if ((seq > ack) && (seq - ack == 0xC123D)){ return RULE_MATCH;}
  }
  return RULE_NOMATCH;
}
```

پیوست ۲:

Snortافزار شده برای نرم کامپایلSignature

TCP SYN/ACK(شناسایی با استفاده از بسته‌های)

```
#include "sf_snort_plugin_api.h"
#include "sf_snort_packet.h"
/* declare detection functions */
int rule201504231eval(void *p);

/* declare rule data structures */
/* flow:to_server; */
static FlowFlags rule201504231flow0 =
{
  FLOW_TO_SERVER
};
static RuleOption rule201504231option0 =
{
  OPTION_TYPE_FLOWFLAGS,
  {
    &rule201504231flow0
  }
};
/* references for sid 201504230 */
static RuleReference *rule201504231refs[] =
{
  NULL
};
/* metadata for sid 201504230 */
/* metadata:; */
static RuleMetaData *rule201504231metadata[] =
{
  NULL
};
RuleOption *rule201504231options[] =
{
  &rule201504231option0,
```

```
  NULL
};
Rule rule201504231 = {
  /* rule header, akin to => tcp any any -> any any */
  {
    IPPROTO_TCP, /* proto */
    "any", /* SRCIP     */
    "any", /* SRCPORT   */
    0, /* DIRECTION */
    "any", /* DSTIP     */
    "any", /* DSTPORT   */
  },
  /* metadata */
  {
    3,  /* genid */
    201504231, /* sigid */
    1, /* revision */
    "trojan-activity", /* classification */
    0,  /* hardcoded priority */
    "TCP Trigger SEQ SYN/ACK",     /* message */
    rule201504231refs, /* ptr to references */
    rule201504231metadata /* ptr to metadata */
  },
  rule201504231options, /* ptr to rule options */
  &rule201504231eval, /* use custom detection function */
  0 /* am I initialized yet? */
};
/* detection functions */
int rule201504231eval(void *p) {
  const u_int8_t *cursor_normal = 0;
  SFSnortPacket *sp = (SFSnortPacket *) p;
  uint32_t seq = 0;
  uint32_t ack = 0;
  if(sp == NULL)
    return RULE_NOMATCH;
  if(sp->payload == NULL)
    return RULE_NOMATCH;
  ack = ntohl(sp->tcp_header->acknowledgement);
  seq= ntohl(sp->tcp_header->sequence);
  //Test for SYN/ACK packets
  if((sp->tcp_header->flags & TCPHEADER_SYN) && (sp->tcp_header->flags & TCPHEADER_ACK))
    if ((ack > seq) &&  (ack - seq != 0xC123E)){ return RULE_NOMATCH; }
    else if ((seq > ack) && (seq - ack != 0xC123E)){ return RULE_NOMATCH; }
```

```
    //Hardcoded SYN/ACK has 6 options
    if (sp->num_tcp_options != 6) return RULE_NOMATCH;
    //MSS
    if(sp->tcp_options[0].option_code != 2) return RULE_NOMATCH;
    //NOP
    if(sp->tcp_options[1].option_code != 1) return RULE_NOMATCH;
    //NOP
    if(sp->tcp_options[2].option_code != 1) return RULE_NOMATCH;
    //SACK
    if(sp->tcp_options[3].option_code != 4) return RULE_NOMATCH;
    //NOP
    if(sp->tcp_options[4].option_code != 1) return RULE_NOMATCH;
    //Window Scale
    if(sp->tcp_options[5].option_code != 3) return RULE_NOMATCH;
    //compare the entire TCP options section
    if (memcmp(sp->tcp_options[0].option_data - 2,
      "\x02\x04\x05\xb4\x01\x01\x04\x02\x01\x03\x03\x05", 12) != 0)
      return RULE_NOMATCH;

    //All conditions satisfied
    return RULE_MATCH;
  }
  return RULE_NOMATCH;
}
/*
Rule *rules[] = {
   &rule201504231,
   NULL
};
*/
```