Projektnummer/Project no
E72779
Kund/Customer
Försvarsmakten
FoT-område
Operationer i cyberdomänen

Handläggare/Our reference
Hannes Holm, Teodor Sommestad

Datum/Date
2019-12-09
Memo nummer/number
FOI Memo 6941

ÖvExCND: Omvärldsbevakning 2019

Sändlista/Distribution:

PROD RPE MTRL (avsett för Produktion)

LEDS INRI (avsett för Lua)

LEDS CIO (avsett för FoT-ordförande David Olgart)

ITF (avsett för Mathias Bjärme)

PROD LEDUND (avsett för Gunnar Marcusson)

MUST

FMV (avsett för SPL Stab), FMV, 115 88 Stockholm

FHS (avsett för fo-samordnare FUS), FHS, Box 27805, 115 93 Stockholm

FOI (avsett för forskningsområdesföreträdare för Operationer i Cyberdomänen)

1. Introduktion

Cybersäkerhetsövningar är ett viktigt medel för att öka logganalytikers förmåga att detektera och hantera cyberangrepp. Det är dock i regel mycket kostsamt att utföra en cybersäkerhetsövning, mycket på grund av alla tekniska inspel de kräver. I synnerhet kräver exekvering av relevanta cyberangrepp mycket arbetstid och ställer också stora krav på expertkunskap. Det är även ont om personal med denna typ av kunskap, vilket skapar problem, särskilt för stora cybersäkerhetsövningar som innefattar många angrepp. Ett sådant exempel är cybersäkerhetsövningen Locked Shields[1], som involverar hundratals cybersäkerhetsspecialister.

Projektet Övning och Experiment för ökad förmåga gällande Cyber Network Defense (ÖvExCND) ämnar förenkla utförandet av cybersäkerhetsövningar genom att automatisera de tekniska inspel som sådana övningar ställer krav på. Under 2018 genomfördes en studie av artiklar publicerade inom akademiska konferenser och tidskrifter för att identifiera andra forskares arbete rörande denna uppgift. Resultatet från studien visade att forskningen nästan uteslutande var av teoretisk natur utan empiriska grunder. Vidare beskrev endast en artikel ett tekniskt verktyg som kunde nyttjas för automatiskt skapa test- och träningsscenarion. Detta verktyg var begränsat till sidoförflyttningsaktioner i Windows-domäner och krävde installation av en speciell fjärrstyrningsmjukvara på en maskin i den angripna domänen för att fungera.

Detta memo utgör en mer tillämpad fortsättning på omvärldsbevakningsdelen utförd under 2018 och beskriver en studie av mjukvaruverktyg för automatisering av angreppsaktiviteter. De övergripande forskningsfrågorna studien besvarar är:

1. *Vilka angreppsverktyg finns för automatisering av cyberangrepp?*
2. *Hur effektiva är de publika angreppsverktyg som automatiserar cyberangrepp?*

Ett angreppsverktyg definieras i denna rapport som ett mjukvaruverktyg som har förmågan att utföra angreppsaktiviteter utan någon typ av användarinteraktion utöver specificering av starttillstånd. Vidare skall verktyget kunna utnyttja på förhand kända sårbarheter för att erhålla privilegier i datornätverk. Det innebär rent praktiskt att verktyg som endast identifierar sårbarheter eller enbart gör teoretiska beräkningar om vilka angrepp som borde fungera inte studeras. Inkluderade verktyg är även tvungna att kunna hantera fler än fem typer av angrepp mot fler än fem olika tekniska plattformar, dock inte samtidigt. Avgränsningarna utesluter därmed verktyg som rör automatiska angrepp mot specifika webbapplikationer (såsom Wordpress).

Angreppsverktyget måste också kunna fungera utan att målet behöver förberedas innan angrepp, det vill säga verktygen ska till exempel fungera utan att en agent installeras på målet innan något angrepp kan ske. Det gör att fokus läggs på angreppsprocessen snarare än vilka skalkommandon som används.

[1] https://ccdcoe.org/news/2018.html

2. Metod

Detta kapitel är strukturerat enligt de två forskningsfrågor som presenterades i kapitel 1. Först beskrivs metoden som använts för att identifiera angreppsverktyg och därefter hur verktygen testats.

2.1 Identifiering av angreppsverktyg

Det gjordes två sökningar efter angreppsverktyg, en efter verktyg skapade i öppen källkod och en efter kommersiella verktyg. Sökningarna utfördes av två erfarna FOI-forskare. De verktyg som slutligen bedömdes som relevanta kartlades mot kategoriseringsmodellen som finns beskriven i bilaga A.

2.1.1 Identifiering av angreppsverktyg skrivna i öppen källkod

För att kunna identifiera angreppsverktyg skrivna i öppen källkod gjordes sökningar i databasen Github med utgångspunkt i en söksträng med 12 kombinerade nyckelord[2]. Resultatet gallrades först manuellt baserat på projekttitel och -sammanfattning. Därefter studerades hela dokumentationen för de projekt som bedömts som lovande. Mognadsgraden hos kvarvarande verktyg bedömdes slutligen utifrån en kodgranskning och kategorisering enligt kategoriseringsmodellen i bilaga A. Mognadsgraden bestämdes utifrån hur omfattande förändringar av koden som krävdes för att den skulle kunna exekvera på ett korrekt sätt. Enbart verktyg som krävde mindre sådana kodförändringar togs med.

2.1.2 Identifiering av kommersiella angreppsverktyg

Utgångspunkten för sökandet efter kommersiella verktyg var de fyra verktygsleverantörer inom *Breach and Attack Simulation*, som analysföretaget Gartner listar i en rapport från 2018[3]. Utifrån dessa fyra gjordes sedan sökningar, främst med hjälp av Google, för att identifiera längre listor på leverantörer med samma inriktning. Sökningar gjordes också på begrepp som ”threat simulation”, ”attack simulation” och ”automate penetration test”.

När en uppsättning kommersiella angreppsverktyg erhållits bedömdes de enligt de gemensamma kriterier som beskrivs i bilaga A.

2.2 Tester av nätverksbaserade angreppsverktyg

När en uppsättning angreppsverktyg identifierats vidtog arbetet med att testa deras effektivitet. För detta krävdes väl preparerade målmaskiner som möjliggjorde mätning av effektiviteten, samt ett pålitligt fundament för att ge de testade verktygen samma förutsättningar.

---

[2] (metasploit AND nmap) OR (deep AND exploit) OR (deep AND metasploit) OR (svm AND metasploit) OR (machine learning AND metasploit) OR (script AND metasploit) OR (pawn AND metasploit) OR (pwn AND metasploit) OR (auto AND attack) OR (auto AND apt) OR (auto AND exploit) OR (auto AND metasploit)

[3] [1] G. Young, “Hype Cycle for Threat-Facing Technologies ,” Gartner, 2017.

2.2.1 Konfiguration av målmaskiner

Alla tester utfördes i datorklustret CRATE hos FOI i Linköping. CRATE är en övnings-, tränings- och forskningsanläggning för IT-säkerhet som FOI driver sedan tio år tillbaka. Tekniskt består CRATE av cirka 1000 fysiska noder och en mängd mjukvaror som möjliggör realisering av stora virtuella nätverk med sammankopplade virtuella maskiner, ungefär som ett simulerat ”mini-internet”.

För testerna användes 556 virtuella målmaskiner tagna från två källor. Den första källan var de 453 grundinstallationer av virtuella maskiner som fanns i CRATE under januari 2019. Dessa grundinstallationer har skapats för allehanda ändamål sedan CRATE bildades, men huvudsakligen för IT-säkerhetsövningar, undervisning och laborationer. Totalt innefattades 453 sådana virtuella maskiner. Den andra källan var de 103 virtuella maskiner som fanns på *www.vulnhub.com* i januari 2019 och som kunde installeras (eller monteras om det rörde sig om en live-image) i CRATE utan större arbetsinsatser[4]. Denna datakälla användes även av bland annat utvecklarna av verktyget DeepExploit.

2.2.2 Reliabel exekvering av angreppsverktyg

För att kunna exekvera verktygen i en kontrollerad miljö installerades de tillsammans med eventuella tillägg på en särskild virtuell maskin som preparerats för studien. I flera fall krävdes revidering av koden i verktygen. Exempelvis har verktyg som byggts för Metasploit-versioner tidigare än version 5.0 reviderats för att fungera även för denna version. I Metasploit version 5 har ett sökfilter som flera verktyg nyttjade tagits bort. Dessutom har parametern RHOST ändrats till RHOSTS för alla serverangrepp. Den virtuella maskin som angreppsverktygen installerats på utrustades med en egenutvecklad programvara som hade följande funktioner:

1. Ta emot instruktioner gällande exekvering av ett test med ett visst angreppsverktyg mot ett givet mål.
2. Avgöra om testet redan utförts.
3. Lägga till testet i exekveringskön.
4. Utföra tester samt rapportera eventuella fel:
    a. Ställa om sitt VxLAN och sin IP-adress så att angriparmaskinen kan nå målet.
    b. Starta om målmaskinen via VirtualBox API.
    c. Säkerställa att målmaskinen kan nås (genom ICMP echo).
    d. Sätta igång alla beroenden som ett angriparverktyg krävde (t.ex. Metasploit).
    e. Starta TCPdump för att logga all nätverkstrafik som genereras under exekveringen av ett verktyg mot ett givet mål.
    f. Exekvera angreppsverktyget mot målet.

[4] Exempelvis var vissa maskiner i vulnhub skapade för andra virtualiseringsplattformar än VirtualBox. Dessa skulle naturligtvis kunna implementeras i CRATE efter visst arbete. Andra maskiner hade inga publicerade inloggningsuppgifter eller aktiverade nätverksinterface och var därmed sannolikt tänkta för mer forensiska/fysiska tillämpningar.

g. Sammanställa all relevant output från angreppsverktyget som produceras under dess exekvering.
h. Avgöra när angreppsverktyget har arbetat klart. På grund av den stora mängden tester som utförs används en övre gräns för exekveringstiden; om ett test tar mer än 3 timmar avslutas det.
i. Städa upp efter testet (till exempel avsluta TCPdump-processen och andra processer som startats av angreppsverktyget, men sedan inte stängts ned).

5. Sätta igång nästa test i kön.

Programvaran hade ett REST-API för användarinteraktion och alla data som producerades under testerna gjordes tillgängliga via en FTP-server. Den virtuella maskinen med programvaran klonades sedan tio gånger i CRATE för att öka testets hastighet (vilken i detta fall skalar relativt linjärt med antalet angriparmaskiner).

En annan programvara utvecklades för att möjliggöra tillförlitliga lastbalanserade tester mot mål via dessa tio angriparmaskiner. Denna programvara tog en lista med tester mot virtuella målmaskiner som indata och slumpade sedan dessa tester över de tio angriparmaskinerna. Varje angriparmaskin hade exklusiv tillgång till sina tilldelade målmaskiner för att ta bort risken för *race conditions* på grund av samtidiga angrepp.

För att motverka risken för bias över tid, till exempel om något okänt problem i nätverksstacken i VirtualBox eller CRATE skulle påverka testernas utfall, användes randomiserade försöksplaner. Två försöksplaner användes, en för verktyget DeepExploit och en annan för de resterande åtta verktygen. Denna partitionering gjordes eftersom DeepExploit krävde upplärning genom angrepp inför faktisk tillämpning. På grund av tidsbrist utgick de strukturerade testerna av DeepExploit under 2019. Däremot gjordes en kvalitativ utvärdering baserat på ostrukturerade tester och kodgranskning. Analysen av DeepExploit beskrivs i avsnitt 4.2.

2.2.3 Variabler för att mäta effektivitet

Tre variabler användes som utfallsmått i testerna:

- antal komprometterade målmaskiner
- typer av lyckade angrepp
- antal Snort-larm av hög prioritet.

De två första variablerna rör angreppens framgång och den tredje variabeln ger ett mått på angreppens kostnad i form av potentiell upptäckt. Snort utgör en de facto standard vid nätverksbaserad intrångsdetektion och tillämpar väl definierade regler för att upptäcka angrepp. Eftersom det kan skilja mycket mellan olika Snort-regelverk för operativa system användes fyra regelverk för testerna: Emerging Threats från 2011 och 2019, samt Vulnerability Research Team från 2014 och 2019. Endast standardregler var aktiverade.

3. Resultat

Resultatet av studien redovisas i två avsnitt uppdelat efter de forskningsfrågor som studien besvarar.

3.1 Identifierade verktyg

För att ge en heltäckande bild av vilka verktyg som finns tillgängliga eftersöktes både verktyg med öppen källkod och kommersiella verktyg. De två kategorierna presenteras var för sig i separata avsnitt.

3.1.1 Angreppsverktyg baserade på öppen källkod

Utifrån de ursprungliga 2290 verktyg som baserades på öppen källkod identifierades 13 verktyg, som bedömdes tillräckligt mogna för att kunna testas empiriskt. Fyra av dessa föll sedan bort på grund av att de krävde att en agent installerades på målet innan något angrepp kunde ske. De fyra verktyg som föll bort är kursiverade i följande lista, dock beskrivs alla 13 verktyg i den efterföljande texten:

- APT2[5]
- AutoExploit[6]
- AutoNSE[7]
- Autoscan[8]
- AutoSploit1[9]
- AutoSploit2[10]
- DeepExploit[11]
- Metasploit-autopwn[12]
- Msf-autopwn[13]
- *Caldera*[14]
- *DeathStar*[15]
- *AutoDANE*[16]
- *Auto-Root-Exploit*[17]

[5] https://github.com/MooseDojo/apt2

[6] https://github.com/shi359/autoExploit

[7] https://github.com/m4ll0k/AutoNSE

[8] https://github.com/darksh3llgr/autoscan

[9] https://github.com/NullArray/AutoSploit

[10] https://github.com/RootUp/AutoSploit

[11] https://github.com/13o-bbr-bbq/machine_learning_security/tree/master/DeepExploit

[12] https://github.com/hahwul/metasploit-autopwn

[13] https://github.com/DanMcInerney/msf-autopwn

[14] https://github.com/mitre/caldera

[15] https://github.com/byt3bl33d3r/DeathStar

[16] https://github.com/sensepost/autoDANE

[17] https://github.com/nilotpalbiswas/Auto-Root-Exploit

APT2 utför serverangrepp via Metasploit (76 moduler stöds) och lösenordsgissning via Hydra. Även verktyget Responder.py nyttjas. Alla angreppsmoduler, utom Responder.py, väljs ut med hjälp av en heuristisk algoritm, som använder resultatet från 28 av Nmap:s och Metasploit:s moduler för kartläggning. Responder.py körs alltid. Konfigurationen för en utvald angreppsmodul anpassas utifrån resultatet av kartläggningsarbetet, där parametrarna RHOST, TARGET, SRVPORT, SRVHOST, LPORT, SMBUser, SMBPass, HTTPUsername och HTTPPassword används. Om en meterpreter-session (en särskild typ av bakdörr i Metasploit:s ramverk) erhålls till en Windows-maskin körs ett kommando för att hämta maskinens lösenordshashar. APT2 kräver inga i förväg höjda behörigheter för att kunna exekveras. Verktyget använder inte någon förhandsinformation kring målsystemet och hanterar enbart IPv4-adresser. APT2 är skrivet i Python och omfattar 4020 rader kod. Github-projektet startades under 2016 och uppdaterades senast i mars 2018.

AutoExploit kan exekvera serverangrepp i Metasploit mot en eller flera förvalda IPv4-adress. Adresserna väljs ut genom en heuristisk algoritm baserat på resultatet från kartläggningsarbete med hjälp av Nmap. Varje angrepp konfigureras via parametrarna RHOST och RPORT. Alla målkonfigurationer och verkansdelar (initiala instruktioner som exekveras vid ett lyckat angrepp, generellt för att sätta upp en bakdörr) testas för varje angrepp, och om det finns många tillämpbara angrepp testas dessa i ordning enligt deras Metasploit-angivna tillförlitlighet. Inga specifika styrkommandon körs i målet om ett angrepp lyckas. AutoExploit består av 852 rader Python-kod. Github-projektet startades under juli 2018 och uppdaterades senast i augusti 2018.

AutoNSE involverar exekvering av 19 olika angreppskoder i Nmap. Koderna väljs ut baserat på en portskanning av 22 portar med Nmap mot en given IPv4-adress. Inga instruktioner körs om angrepp lyckas. AutoNSE är ett shellskript på 526 rader kod. Github-projektet startades under april 2018 och uppdaterades senast i november 2018.

Autoscan innefattar angrepp i form av resursskript i Nmap, ett fåtal moduler i Metasploit och verktyget Brutex[18], vilket huvudsakligen involverar lösenordsgissning med Hydra. Angreppsmoduler i Metasploit väljs och konfigureras (RPORT, RHOST) baserat på resultatet från ett fåtal utvalda Metasploit-moduler, resursskript i Nmap och portskans via Nmap. Sårbarheter i webbtillämpningar analyseras även via Yasuo, sqlmap, Nikto och arachni. Ett verktyg kallat theHarvester[19] körs för att extrahera publik information om målet. Inga automatiska beslut tas dock baserat på resultatet från dessa verktyg. Lösenordshashar hämtas från mål om något tillämpat Metasploit-angrepp lyckas mot ett Windows-baserat mål. Enbart IPv4-adresser stöds som mål. Autoscan består av 1296 rader kod skrivet i Bash. Github-projektet startades under september 2018 och uppdaterades senast i januari 2019.

AutoSploit1 involverar exekvering av 263 olika serverangrepp i Metasploit mot en given IPv4-adress. Verktyget har funktionalitet för att automatiskt samla in data från Shodan för urval av angrepp. Om inte måladressen finns i Shodan exekveras alla server-angrepp mot målmaskinen. Inga kommandon körs på framgångsrikt angripna maskiner. AutoSploit1

[18] https://github.com/1N3/BruteX

[19] https://github.com/laramies/theHarvester

består av 1566 rader kod skrivet i Python. Github-projektet startades under januari 2018 och uppdaterades senast i juli 2019.

AutoSploit2 innefattar exekvering av ett fåtal serverangrapp i Metasploit mot en given IPv4-adress. Dessa angrepp och deras konfigurationer (RHOST, LPORT, payload, SMBUser, SMBPass, Domain, Password, User, User_SID) väljs ut baserat på resultatet från portskanning med Nmap och ett fåtal webbapptester som utvecklarna själva hittat på. Inga kommandon körs på framgångsrikt angripna maskiner. AutoSploit2 innefattar 412 rader kod skrivet i Ruby och Bash. Github-projektet startades under juni 2017 och uppdaterades senast i februari 2018.

DeepExploit innefattar serverangrepp med hjälp av Metasploit. Angreppen mot en given IPv4-adress väljs ut med hjälp av portskanning med Nmap, samt ett fåtal webbapptester som utvecklarna själva hittat på. Till skillnad från de övriga angreppsverktygen involverar inte DeepExploit enbart heuristiska algoritmer för urvalsprocessen, utan också maskininlärningsfunktioner för prioritering av angreppsmoduler och angreppskonfigurationer. Rent praktiskt betyder detta att DeepExploit har två lägen; ett träningsläge och ett testläge. I träningsläget körs verktyget mot maskiner för att träna en maskininlärningsmodell, i testläget tillämpas denna modell. Maskininlärningsalgoritmerna som tillämpas är djupa neurala nätverk som tränas upp genom reinforcement learning (via algoritmen A3C[20]) för urval av angrepp och Naive Bayes i form av bag-of-words för profilering av tre olika webbtillämpningar. Om en maskin angrips framgångsrikt under testläget körs fyra kommandon; uppgradering av terminalkommandoskal till Meterpreter, behörighetseskalering via den angripna maskinen till interna nätverk, kartläggningar med hjälp av proxychains och Nmap mot dessa nätverk och slutligen angrepp mot identifierade maskiner. DeepExploit består av 4181 rader kod skrivna i Python. Github-projektet startades under maj 2017 och uppdaterades senast i februari 2019.

Metasploit-autopwn använder Metasploit för serverangrepp mot portar och IPv4-adresser som finns lagrade i Metasploit:s databas. För testerna beskrivna i denna rapport kördes modulen *auxiliary/scanner/portscan/tcp* mot målmaskinen för att uppdatera Metasploit:s databas med målets exponerade TCP-portar. Metasploit-autopwn är skrivet som ett resursskript till Metasploit och består av 400 rader kod skrivna i Ruby. Github-projektet är en klon av en modul som ursprungligen fanns i Metasploit, men som togs bort för ett antal år sedan. Projektet startades under augusti 2017 och uppdaterades senast i augusti 2018.

Msf-autopwn utför serverangrepp via Metasploit. Angreppen väljs ut baserat på kartläggningar gjorde med Nmap eller Nessus, samt i ett fåtal fall även med hjälp av kartläggningsmoduler i Metasploit (Windows SMB-tester, till exempel för MS17_010 och SMB named pipes). Om kartläggningsarbetet görs med Nessus väljs de Metasploit-angrepp ut som rekommenderas av Nessus. Vid användning av Nmap tillämpas enbart ett fåtal Windows-baserade angrepp. Msf-autopwn består av 643 rader kod skrivna i Python. Github-projektet startades under mars 2018 och uppdaterades senast i juni 2018.

[20] Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." *International conference on machine learning*. 2016.

Caldera skapades av MITRE som bygger på MITRE:s ATT&CK-ramverk. För närvarande innefattar Caldera enbart angrepp mot Windows Active Directory (AD)-miljöer, i huvudsak horisontella förflyttningar via till exempel psexec eller wmic. Caldera kräver dock att en agent installeras på målsystemet för att fungera, vilket diskvalificerar Caldera från vidare testning i studien. Utöver webbfrontend-kod (cirka 1.2 miljoner rader HTML/CSS/Javascript) består Caldera huvudsakligen av 2440 rader Python, 2221 rader Powershell och 936 rader Go. Github-projektet startades under november 2017 och uppdaterades senast i oktober 2019.

DeathStar bygger på automatisering av de Windows-AD-angrepp som finns i verktyget PowerShell Empire och fungerar därmed på liknande sätt som Caldera, fast med mer hårdkodad logik. DeathStar består av 580 rader Python-kod. Github-projektet startades under maj 2017 och uppdaterades senast i september 2019, vilket dock blev den sista eftersom utvecklingen av PowerShell Empire lades ner under samma månad.

AutoDANE står för Auto Domain Admin and Network Exploitation och är relativt likt DeathStar, fast med Metasploit som medium för angrepp istället för Powershell Empire. AutoDane består huvudsakligen av Python (9051 rader), Qt (4588 rader) och SQL (1276 rader). Github-projektet startades under oktober 2015 och uppdaterades senast i augusti 2016.

Auto-Root-Exploit innefattar tester av olika lokala angreppskoder mot Linux, FreeBSD och MacOS-baserade system. Angreppskoderna används för att eskalera en existerande behörighet till root. Verktyget består av 630 rader skrivna i bash. Github-projektet startades under maj 2016 och uppdaterades senast i januari 2019.

3.1.2 Kommersiella angreppsverktyg

Sökningarna efter kommersiella angreppsverktyg gav träff på ett stort antal företag och produkter. Det var dock få av dessa som matchade de typöverskridande kriterier som satts upp. Produkten Contrast Assess var exempelvis en lösning begränsad till webb-tillämpningar. CYBOT å sin sida tolkar vi som ett verktyg för att identifiera angreppsscenarion baserat på sårbarhetsskanningar, men utan att utföra angreppen. Produkten Cymulate genomför som vi förstår det endast förvalda angrepp på ett strukturerat sätt (motsvarande CRATE-verktyget SVED[21]). Totalt 13 kommersiella produkter som först identifierades som relevanta kunde avskrivas som irrelevanta efter noggrannare granskning.

Fem kommersiella produkter beskrevs inte på ett sätt som gjorde det möjligt att avgöra om de hade funktionalitet som var relevant för denna studie. Det var till exempel svårt att bedöma om den produkt företaget XM Cyber marknadsför utför några faktiska angrepp eller om simuleringarna enbart innefattar beräkningar. De övriga företagen som var svåra att bedöma var: Pentom, Shield-Cybot och PenTeraTM. Den produkt som närmast matchade det som eftersöktes var PCYSYS. Det framgår dock inte från det publika

[21] Holm, Hannes, and Teodor Sommestad. "SVED: Scanning, vulnerabilities, exploits and detection." *MILCOM 2016-2016 IEEE Military Communications Conference*. IEEE, 2016.

materialet om PCYSYS faktiskt fattar komplexare beslut på egen hand eller om det likt Cymulate och ett antal andra verktyg exekverar förbestämda kommandon på ett strukturerat sätt.

3.2 Tester av angreppsverktyg

I Tabell 1 visas antalet unika maskiner som de testade verktygen kompromtterade. Som synes var spridningen mellan verktygen mycket stor. Det bästa verktyget (autoscan) lyckades kompromettera nästan dubbelt så många maskiner som det näst bästa verktyget (metasploit-autopwn) och de sämsta verktygen (AutoSploit2 och APT2) lyckades inte kompromettera en enda maskin.

Tabell 1. Kompromtterade maskiner.

Verktyg	Kompromtterade maskiner
autoscan	42
metasploit-autopwn	22
msf-autopwn	21
AutoExploit	7
AutoSploit1	3
AutoNSE	1
AutoSploit2	0
APT2	0

Totalt åtta olika unika angreppskoder lyckades. Sju av dessa rörde serverangrepp i Metasploit mot Windows-plattformen och det åttonde rörde Nmap och lösenordsgissning mot SSH i en Ubuntu Linux. Dessa resultat ligger i linje med tidigare studier av FOI[22] gällande automatisering av angrepp.

I tabell 2 syns det tydligt att antalet lyckade angrepp skiljer sig mycket mellan de olika verktygen trots att angreppen testades mot samma maskiner under samma förutsättningar och att alla (utom AutoNSE) innefattar Metasploit-angrepp. Notera särskilt skillnaden i resultat mellan metasploit-autopwn och autoscan för modulerna ms03_026_dcom och ms08_067_netapi.

[22] Holm, Hannes, and Teodor Sommestad. "So long, and thanks for only using readily available scripts." *Information & Computer Security* 25.1 (2017): 47-61. Holm, Hannes, and Ioana Rodhe. "A Model for Predicting the Likelihood of Successful Exploitation." *Hawaii International Conference on Systems Schiences (HICSS)*, 2020.

Tabell 2. Lyckade angreppsförsök.

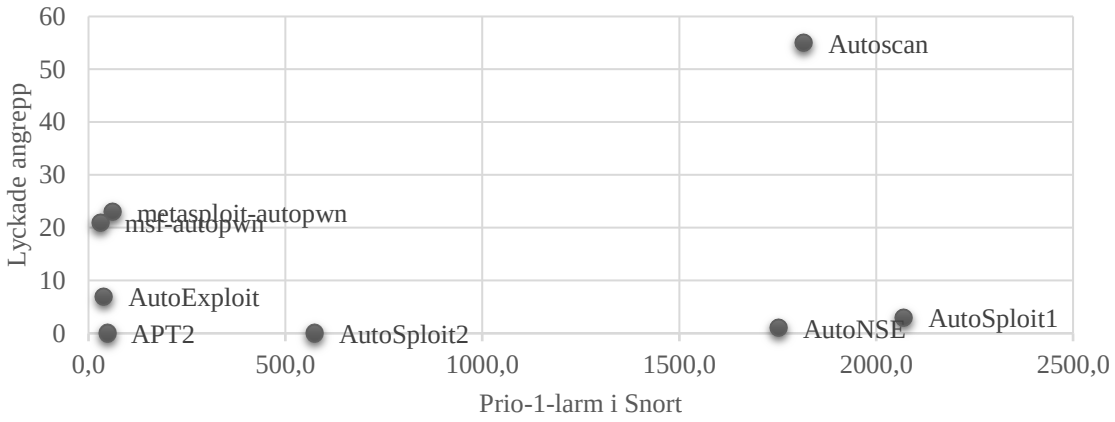
Angreppskod	Antal lyckade försök för olika verktyg							
	auto scan	metasploit-autopwn	msf-autopwn	Auto Exploit	Auto Sploit1	Auto NSE	Auto Sploit2	APT2
ms03_026_dcom	42	8	0	7	0	0	0	0
ms01_023_printer	2	0	0	0	2	0	0	0
ms03_022_nsiislog_post	1	0	0	0	1	0	0	0
ms06_040_netapi	0	2	0	0	0	0	0	0
ms08_067_netapi	10	13	18	0	0	0	0	0
ms17_010_eternalblue	0	0	3	0	0	0	0	0
ssh account brute-force	0	0	0	0	0	1	0	0

De fyra Snort-regelverken som beskrivs i avsnitt 2.2.3 testades mot totalt 200GB med nätverkstrafik. Medelantalet larm av prioritet ett (det vill säga hög prioritet) som genererades av testerna beskrivs i Tabell 3. Spridningen är som synes mycket stor, från 2071 larm i snitt för AutoSploit1 till 30 larm i snitt för msf-autopwn. Antalet larm beror på urvalet av moduler som exekveras av verktygen och hur dessa moduler konfigureras. Verktygens beteenden kan förklara varför AutoSploit1, autoscan och AutoNSE genererade många larm. AutoSploit1 kartlägger inte, utan utför direkt 263 angreppsmoduler oavsett målets konfiguration. Autoscan och AutoNSE använder istället Nmap-moduler med mycket aggressivt beteende. AutoSploit2 däremot använder ett lägre antal och mindre aggressiva Nmap-moduler för kartläggning än vad Autoscan och AutoNSE gör. Metasploit-autopwn å sin sida utför kartläggning via modulen *auxililiary/scanner/portscan/tcp*, vilken enbart letar efter öppna portar utan att identifiera tjänsterna bakom. Även APT2 och msf-autopwn har ett mindre aggressivt beteende genom att enbart tillämpa ett fåtal utvalda kartläggningsmoduler och angrepp. Resultatet för AutoExploit kan tyckas lite märkligt eftersom verktyget använder en Nmap-profil som är snarlik AutoSploit2:s, fast med fler angreppsförsök. Djupare analyser visade att en bugg i dess kod medförde att många av angreppstesterna inte utfördes.

Tabell 3. Snort prioritet 1 larm.

Verktyg	Medel	Lyckade angrepp
AutoSploit1	2071,1	3
autoscan	1816,5	55
AutoNSE	1752,0	1
AutoSploit2	575,0	0
metasploit-autopwn	60,6	23
APT2	47,7	0
AutoExploit	39,7	7
msf-autopwn	30,0	21

Som Figur 1 och Tabell 3 visar finns det ingen tydlig koppling mellan larm och lyckade angrepp.



Figur 1. Lyckade angrepp och Snort prioritet 1 larm.

4. Slutsatser och diskussion

Detta kapitel presenterar först validitet och reliabilitet för de utförda testerna och ger sedan en kritisk analys av de studerade angreppsverktygen. Sist ges en beskrivning av framtida arbete kopplat till tester av existerande verktyg.

4.1 Validitet och reliabilitet för utförda tester

Det finns en del potentiella reliabilitets- och validitetsproblem som rör testerna i kapitel 3. Först och främst inkluderar analysen av testerna enbart uppnådda fjärrstyrningsmöjligheter av maskiner. Information som indirekt kan ge utökade rättigheter analyseras inte. Exempelvis utvärderar AutoNSE om anonym access av en ftp-server tillåts men går inte vidare och inhämtar information från ftp-servern. Hade den gjort det hade det eventuellt varit möjligt att utifrån informationen med enkla medel få utökad behörighet på en maskin. Resultaten bör tolkas med detta i åtanke. Det ska även noteras att vissa av verktygen inkluderade moduler som är bättre lämpade för driftsatta nät med flera datorer. Exempelvis tillämpar APT2 angreppsverktyget Responder.py och autoscan utför lösenordsgissningar mot diverse protokoll. Om angreppen görs mot solitära maskiner utnyttjas inte dessa moduler. Dessutom innebär tidsgränsen på tre timmar per test problem för verktyg som utför tidskrävande tester, såsom lösenordsgissning, då dessa sällan hinner bli klara.

4.2 Kritisk analys av testade angreppsverktyg

De testade verktygens funktionalitet är på många sätt begränsad. Till att börja med stödjer de endast IPv4-adresser för målangivelse. Dessutom hanterar de bara indata i form av nätverkskartläggningar utförda med Nmap eller Nessus. Inget av de testade verktygen tar ställning till andra parametrar än om angrepp lyckas. Det finns inte heller några beskrivningar i dokumentation eller kod som förklarar utvecklarnas val av verktygsbeteende eller ordningen på olika aktiviteter. Det är även oklart om hänsyn tas till kostnaden för ett angreppsförsök eller det faktiska värdet av lyckade angrepp. Alla testade verktyg utför sina angreppsförsök i maximal hastighet och tycks inte ta hänsyn till sin egen eller målens robusthet. Detta visade sig under testerna, bland annat genom att

- Metasploit kraschade eller slutade ta emot instruktioner på grund av överbelastning
- angrepp som skulle fungera misslyckades eftersom tidigare (inte fungerande) angrepp kraschat målservern (ofta SMB i Windows).

Den stora variationen på resultaten mellan verktygen, trots liknande förutsättningar, förvånar. Konfigurationerna för de testade angreppskoderna är förmodligen inte en signifikant faktor eftersom de är snarlika för de olika verktygen. En troligare anledning är att ordningen på utförandet av olika tester skiljer sig mellan verktygen i kombination med att ett misslyckat angrepp kan medföra att en server stängs ned eller hamnar i ett korrupt tillstånd. Denna hypotes styrks av resultaten (se tabell 2), där autoscan lyckas över 500 % bättre än metasploit-autopwn vid användning av modulen ms03_026_dcom, men nästan 25 % sämre för modulen ms08_067_netapi.

Endast fyra av nio verktyg involverar särskilt utvalda angrepp mot konfigurations-sårbarheter, övriga verktyg testar enbart serverangrepp i Metasploit. Dessutom ställer de allra flesta verktyg enbart in RHOST och RPORT-parametrarna i Metasploit. Undantagen är APT2 och AutoSploit2 som involverar nio olika parametrar. Dessa verktyg har å andra sidan endast stöd för en liten delmängd av alla serverangrepp i Metasploit.

Endast tre verktyg involverar styrkommandon på framgångsrikt angripna maskiner. Dessa kommandon är alla del i Metasploit-moduler. Ett modul ger ökad behörighet i systemet och de andra två hämtar lösenordshashar från Windows-maskiner.

DeepExploit genomgick enbart strukturerade tester i mindre format och kodgranskning. Testerna visade att verktyget till skillnad från övriga verktyg genomförde värderingar av olika angreppsalternativ. De verktyg som testades under 2019 använde endast enkla heuristiska algoritmer såsom ”om port X ses – testa alla angrepp som rör port X”. DeepExploit använder istället djupa neurala nätverk och reinforcement learning. Det har dock ett antal buggar, problem och begränsningar, till exempel

- stängs DeepExploit ned om inga portar är öppna på en maskin som angrips, vilket skapar problem om det finns maskiner att angripa kvar i kön.
- har DeepExploit en bugg gällande race conditions i träningsläget då alla trådar i applikationen delar på samma lista med tillämpbara angrepp. I praktiken kan detta leda till att flera angrepp samtidigt körs mot samma mål.
- kör DeepExploit enbart kommandon som rör ökad behörighet på kompromette rade maskiner. Kommandon rörande persistens eller lokal datainsamling körs inte alls.
- DeepExploit antar att det finns linjära samband mellan olika egenskaper som inte har tydliga sådana samband (till exempel att windows =1 och linux=2).
- Responsvariabeln för maskininlärningsfunktionen pekar på vilken verkansdel (eng. payload) som skall väljas. Detta är udda eftersom valet av verkansdel är relativt oviktigt. Vissa verkansdelar är bättre (robustare eller har mer funktionalitet) än andra och bör därför snarare hårdkodas för varje angreppskonfiguration. Under testerna misslyckades DeepExploit med tillämpbara angrepp eftersom den valde verkansdelar som inte var giltiga för angreppskonfigurationen, eller inte erbjöd fjärrstyrningsfunktionalitet (till exempel *payload/generic/debug_trap*).

4.3 Framtida arbete

Projektet planerar att fortsätta med ytterligare tester av publikt tillgängliga verktyg under 2020. Testerna som gjordes under 2019 omfattade enbart angreppsverktyg som inte kräver särskilda behörigheter i målnätverket. Under arbetets gång identifierades ett fåtal andra typer av automatiska angreppsverktyg som av denna anledning exkluderades. Dessa var

- Caldera, DeathStar och AutoDANE, vilka utför horisontella förflyttningar i Windows active directory domäner
- Auto-Root-Exploit, vilket utför lokala angrepp.

Vi planerar att testa de tidigare exkluderade verktygen under 2020. Systematiska tester av DeepExploit är också planerade för 2020.

Appendix A. Kategoriseringsmodell

Kategoriseringsmodellen skapades baserat på resultat från ÖvExCND 2018 och förädlades sedan baserat på de studerade verktygen. Kategorierna i modellen är

- Angrepp som stöds
    - Fjärrangrepp
        - Implementation
            - Server
            - Klient
        - Konfiguration
    - Lokala angrepp
        - Implementation
        - Konfiguration
- Kartläggningsfunktioner som stöds
    - Nätverk
        - Passiv
        - Aktiv
    - Lokala
    - Sekundära källor
- Missbruk och styrkommandon
    - Behörighetseskalering i system (pivotering)
    - Datainsamling
    - Persistens
- Prioritering av angrepp
    - Algoritmer
        - Heuristik
        - Maskininlärningsmodeller
    - Variabler som tas ställning till
- Licensmodell
- Förkonfigurationskrav (Krav på ursprunglig behörighet)
- Förkonfigurationsmöjligheter
- Källkod
    - Språk
    - Rader kod
- Val av mål
- Ursprunglig utgåva av verktyg
- Senaste ändring av verktyget
- Övrigt