# Fallout from Log4Shell-related Vietnamese Cryptocurrency Exchange Attack: KYC Data for Sale on Dark Web

cadosecurity.com/fallout-from-log4shell-related-vietnamese-cryptocurrency-exchange-attack-kyc-data-for-sale-on-dark-web

January 20, 2022


Blog

January 20, 2022

**Introduction**

Since its discovery at the end of 2021, Log4Shell – a zero-day vulnerability affecting Apache's Log4j logging library – has been actively exploited to infect vulnerable hosts with malware. We previously reported on a novel strain of ransomware, named Khonsari, that targeted Windows Servers vulnerable to Log4Shell.

In this report we'll look at a published account of the compromise of a Vietnamese cryptocurrency exchange, named Onus, where the attackers exploited the Log4Shell vulnerability to steal customer data and advertise it for sale on a popular hacking forum.

Specifically, we discovered a user offering a 9 TB database of information, including extremely valuable Know Your Customer (KYC) images and video, along with names, phone numbers and email addresses.

**Know Your Customer (KYC)**

Know Your Customer (KYC) refers to a set of guidelines for financial institutions and decentralized finance (DeFi) services alike to combat money laundering and other fraudulent activities. Generally, KYC involves identifying a customer through a formal identification procedure. Copies of official state-issued identification, such as passports or driving licences, are typically requested by the financial institution when a customer applies for a financial product. Examples of such products include personal loans and credit cards.

Naturally, this data is extremely valuable to hackers, as it allows them to impersonate another individual when making an application for a financial product. The funds raised from this could then be used to finance other criminal endeavours. Since this information is intended to reliably identify an individual, it could also be used in more targeted attacks. An attacker could suspect that an individual is likely to use Onus, find their KYC data in the leaked database and leverage that to conduct spear-phishing and other social engineering attacks.

**Attack Overview**

We first learned about the attack on Onus via a report published by CyStack – Onus' security partner – in late December 2021.

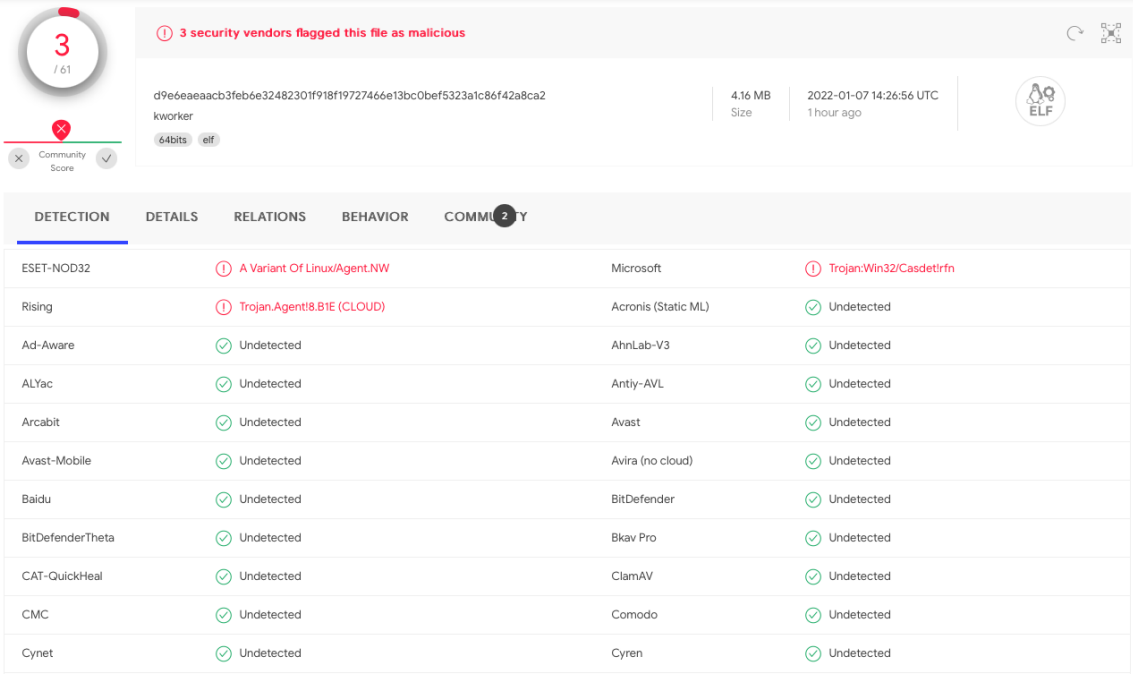
According to the report, Onus had deployed payment software developed by Cyclos, which contained a version of Log4j vulnerable to Log4Shell. Despite following advice to patch the vulnerability shortly after it had been disclosed, attackers still managed to exploit the Log4Shell vulnerability before a patch was issued. They then used shell commands to retrieve AWS credentials which, in turn, allowed them to compromise and subsequently delete the contents of S3 buckets containing customer data.This resulted in a leak of personal data from some 2 million Onus customers, including eKnow Your Customer information (eKYC) and password hashes.

Due to the nature of cryptocurrency exchanges, owners often have to collect highly-sensitive information from their customers in order to comply with regulatory requirements. This information typically includes passport scans and even photographs of the customers themselves which, as we'll demonstrate, is highly sought after in underground marketplaces.

**Analysis of Malicious Payload**

As Cystack reported, a file named *kworker*(masquerading as a legitimate Linux process) was installed on the vulnerable host. This payload served as a backdoor to the host, allowing the attackers to establish Command and Control (C2) communication and exfiltrate the customer

data. The file was uploaded to VirusTotal on December 26, 2021 by a user in Vietnam and, at the time of writing, has few detections.




We downloaded and analysed this payload and our findings are largely the same as CyStack's. The capabilities of this malware are covered in their original report but to summarise:

- After execution, the malware creates an SSH connection to the host 45[.]147[.]230[.]219 over port 81. A username of "peter" and a password of "kim" is used for authentication.
- A SOCKS connection is also created as a backup in the event that SSH is unavailable. The same credentials are used.
- Stdout and stderr of the compromised machine is relayed back to the C2 server and commands are received by the compromised host.

```
int main.main(int arg0, int arg1, int arg2, int arg3, int arg4, int arg5) {
    while (rsp <= *(r14 + 0x10)) {
            runtime.morestack_noctxt.abi0();
    }
    rsp = rsp - 0x20;
    stack[24] = rbp;
    rax = runtime.newobject(rdi, rsi, rdx, rcx, r8, r9, stack[0]);
    stack[16] = rax;
    *(rax + 0x10) = 0x0;
    *(int128_t *)rax = intrinsic_movups(*(int128_t *)rax, xmm15);
    *(rax + 0x18) = 0x0;
    rax = runtime.newobject(rdi, rsi, rdx, rcx, r8, r9, stack[0]);
    *(rax + 0x18) = *qword_6dffb8;
    if (*(int32_t *)runtime.writeBarrier == 0x0) {
            *(rax + 0x10) = *main.SSHHost;
    }
    else {
            rdi = rax + 0x10;
            rax = runtime.gcWriteBarrierCX(rdi, rsi, *qword_6dffb8, *main.SSHHost, r8, r9);
    }
    *(rax + 0x28) = *qword_6dffc8;
    if (*(int32_t *)runtime.writeBarrier == 0x0) {
            *(rax + 0x20) = *main.SSHPort;
    }
    else {
            rdi = rax + 0x20;
            rax = runtime.gcWriteBarrierCX(rdi, rsi, *qword_6dffc8, *main.SSHPort, r8, r9);
    }
    *(rax + 0x38) = *qword_6dffe8;
    if (*(int32_t *)runtime.writeBarrier == 0x0) {
            *(rax + 0x30) = *main.SSHUser;
    }
    else {
            rdi = rax + 0x30;
            rax = runtime.gcWriteBarrierCX(rdi, rsi, *qword_6dffe8, *main.SSHUser, r8, r9);
    }
    *(rax + 0x48) = *qword_6dffd8;
    if (*(int32_t *)runtime.writeBarrier == 0x0) {
            *(rax + 0x40) = *main.SSHPwd;
    }
    else {
            rdi = rax + 0x40;
            rax = runtime.gcWriteBarrierCX(rdi, rsi, *qword_6dffd8, *main.SSHPwd, r8, r9);
    }
    *(rax + 0x58) = *qword_6e0008;
    if (*(int32_t *)runtime.writeBarrier == 0x0) {
            *(rax + 0x50) = *main.SocksUser;
    }
    else {
            rdi = rax + 0x50;
            rax = runtime.gcWriteBarrierCX(rdi, rsi, *qword_6e0008, *main.SocksUser, r8, r9);
    }
    *(rax + 0x68) = *qword_6dfff8;
    if (*(int32_t *)runtime.writeBarrier == 0x0) {
            *(rax + 0x60) = *main.SocksPwd;
    }
```

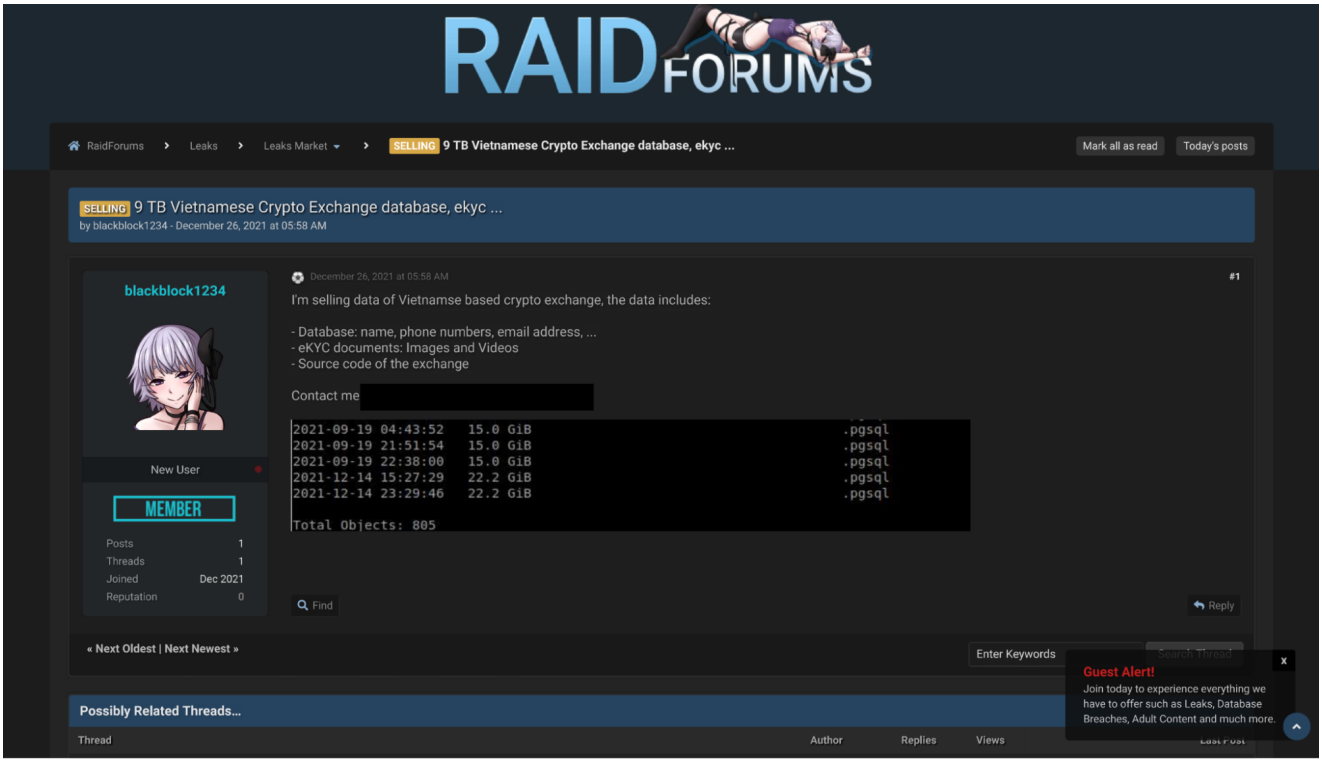
*C Pseudocode of the malware's main() function*

**Stolen Data from AWS S3 Buckets for Sale**

After hearing about the attack on Onus, researchers from Cado Security searched through popular hacking forums to see if evidence of data stolen in this attack could be found.

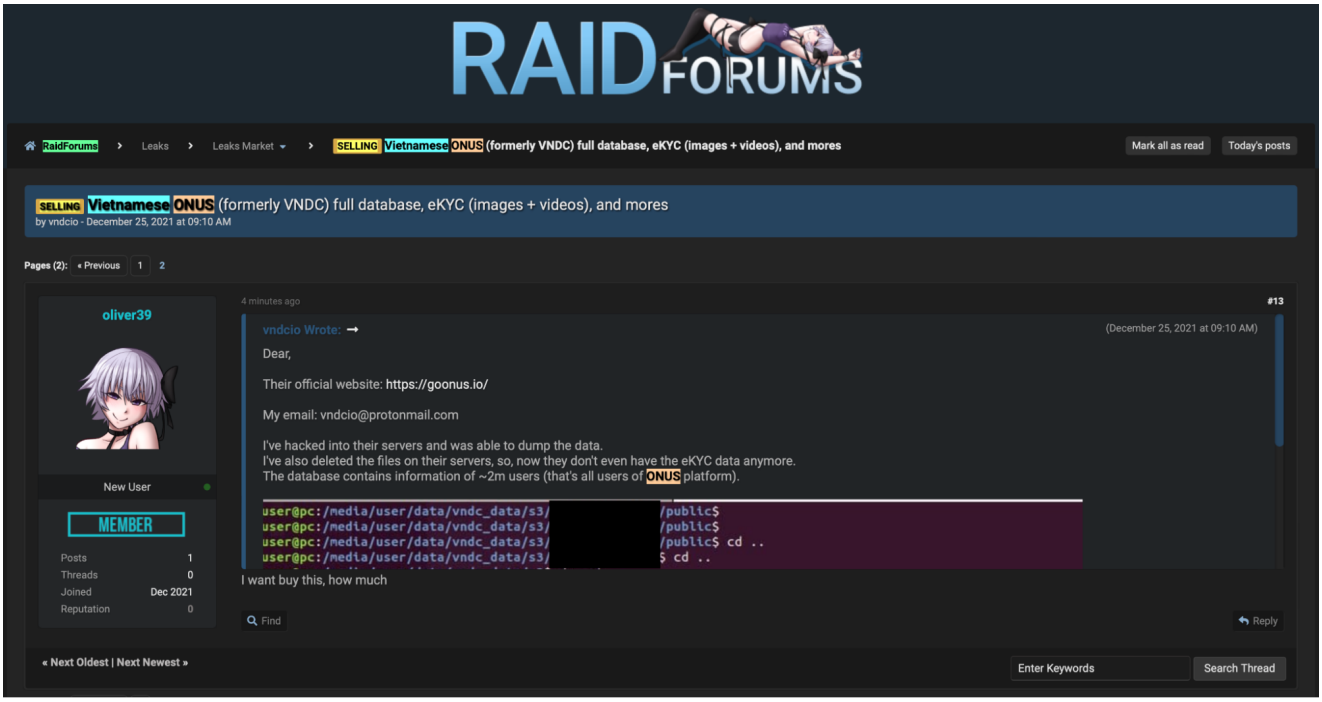
On one such forum, we found a user named *blackblock1234* advertising a 9 TB (!) database of data stolen from a Vietnamese cryptocurrency exchange. The user mentions that the dataset includes names, phone numbers, and email addresses, along with eKYC

documentation such as images and videos. The source code of the exchange itself is also said to be included.




*Post advertising 9TB of Vietnamese Crypto Exchange data on RAIDforums*

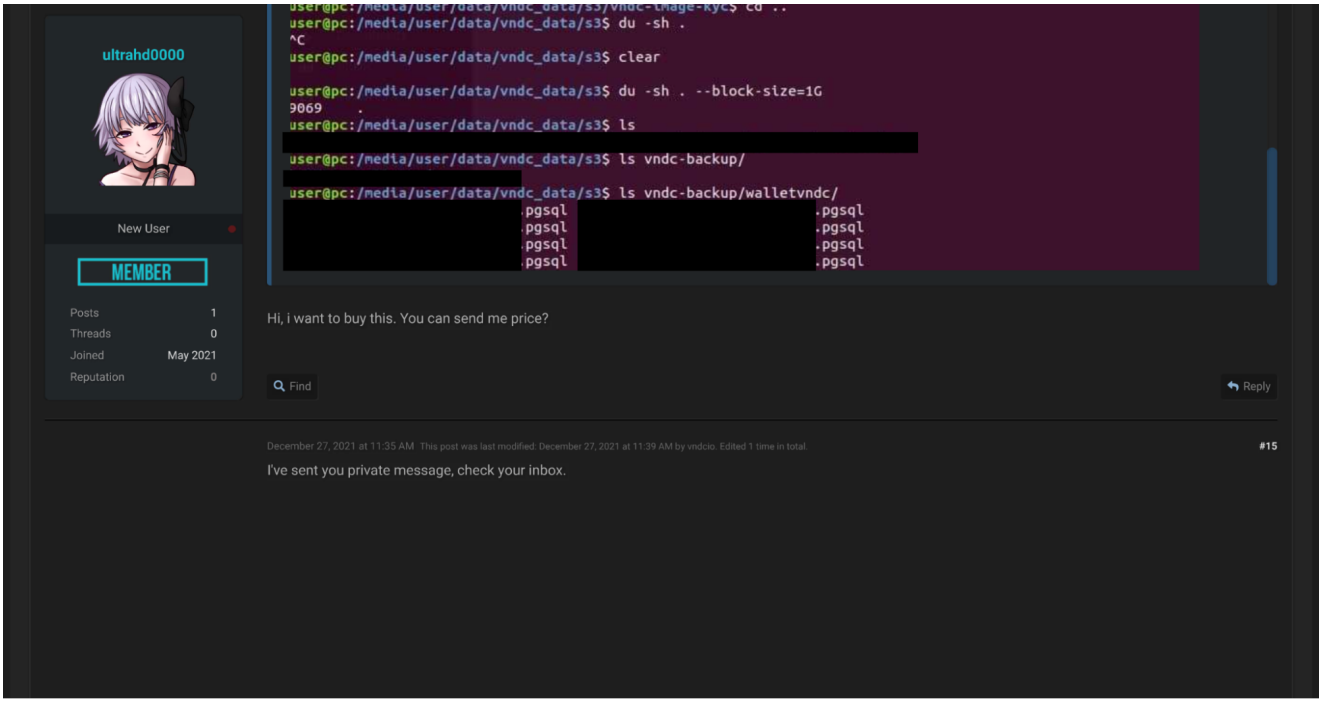
Another user named *vndcio* started a similar thread in RAIDforum's *Leaks Market* subforum. Unfortunately, a cached version of the original post could not be retrieved. However, we can see the content of the original post quoted in a subsequent reply.



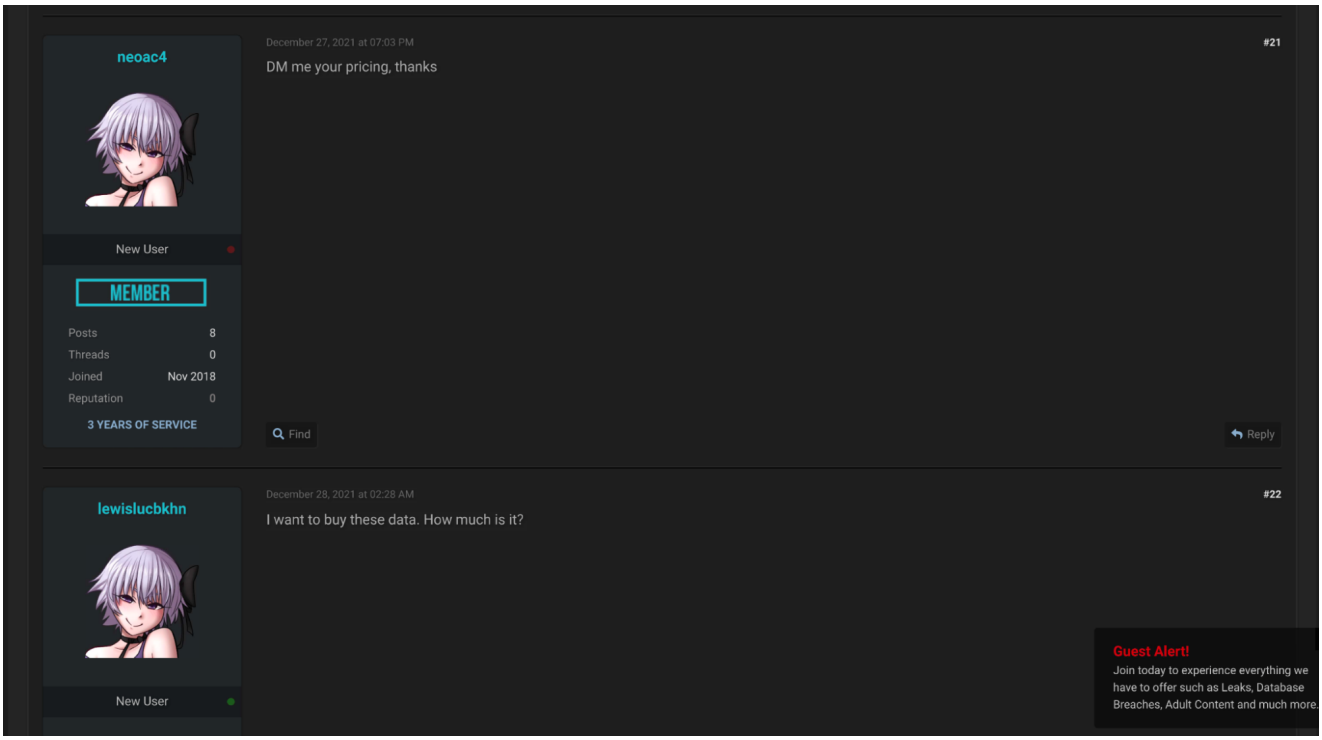

*Quoted reply to vndcio's original post*

As can be seen from the above, the *vndcio* user advertises similar data and provides a screenshot where the directory structure of the leaked files is shown.

Subsequent posts demonstrate the interest that this leak generated, with several users requesting pricing information.



*Pricing request for vndcio leak*



*Additional pricing requests*

At the time of writing, the only threads posted by both the *blackblock1234* and *vndcio* users were the ones advertising this data leak. Both users joined RAIDforums at the end of December 2021, just prior to the release of CyStack's report. This suggests that the accounts

were created solely to advertise the data stolen in this particular attack.

**Recommendations**

Given that this attack relied on the presence of Log4Shell in the target environment, the primary recommendation would be to upgrade Log4j to version 2.3.1, which includes a security fix for this vulnerability.

Admins should also review permissions for AWS resources in their organisation. In this case, the attackers were able to easily pivot to S3 buckets after exploiting Log4Shell due to the *AmazonS3FullAccess* permission having been granted to a compromised access key. This permission should be used sparingly and the principle of least privilege applied.

Monitoring of AWS infrastructure should also be established, with system logs from cloud resources forwarded to a SIEM with automated alerting in place. The SIEM should alert admins to any unexpected access to S3 buckets from web-facing infrastructure, providing an additional layer of security on top of access token permissions. Similarly, outbound connections from AWS resources should be monitored for connections to unknown IP addresses, with domain/IP whitelisting established as necessary.

Since this attack depended on the execution of malware served via exploitation of Log4Shell, a final recommendation would be to enable application whitelisting on cloud servers. This can be achieved using SELinux on Linux servers and is an effective mitigation against malware attacks, as the malware itself is prevented from executing.

**Indicators of Compromise**

| Filename | SHA256 |
| --- | --- |
| kworker | d9e6eaeaacb3feb6e32482301f918f19727466e13bc0bef5323a1c86f42a8ca2 |

| IP Address |
| --- |
| 45[.]147[.]230[.]219 |

For tips and best practices for performing cloud incident response investigations, check out our playbooks:

> **About Cado Security**

Cado Security provides *the* cloud investigation platform that empowers security teams to respond to threats at cloud speed. By automating data capture and processing across cloud and container environments, Cado Response effortlessly delivers forensic-level detail and unprecedented context to simplify cloud investigation and response. Backed by Blossom

Capital and Ten Eleven Ventures, Cado Security has offices in the United States and United Kingdom. For more information, please visit https://www.cadosecurity.com/ or follow us on Twitter @cadosecurity.

Prev Post Next Post