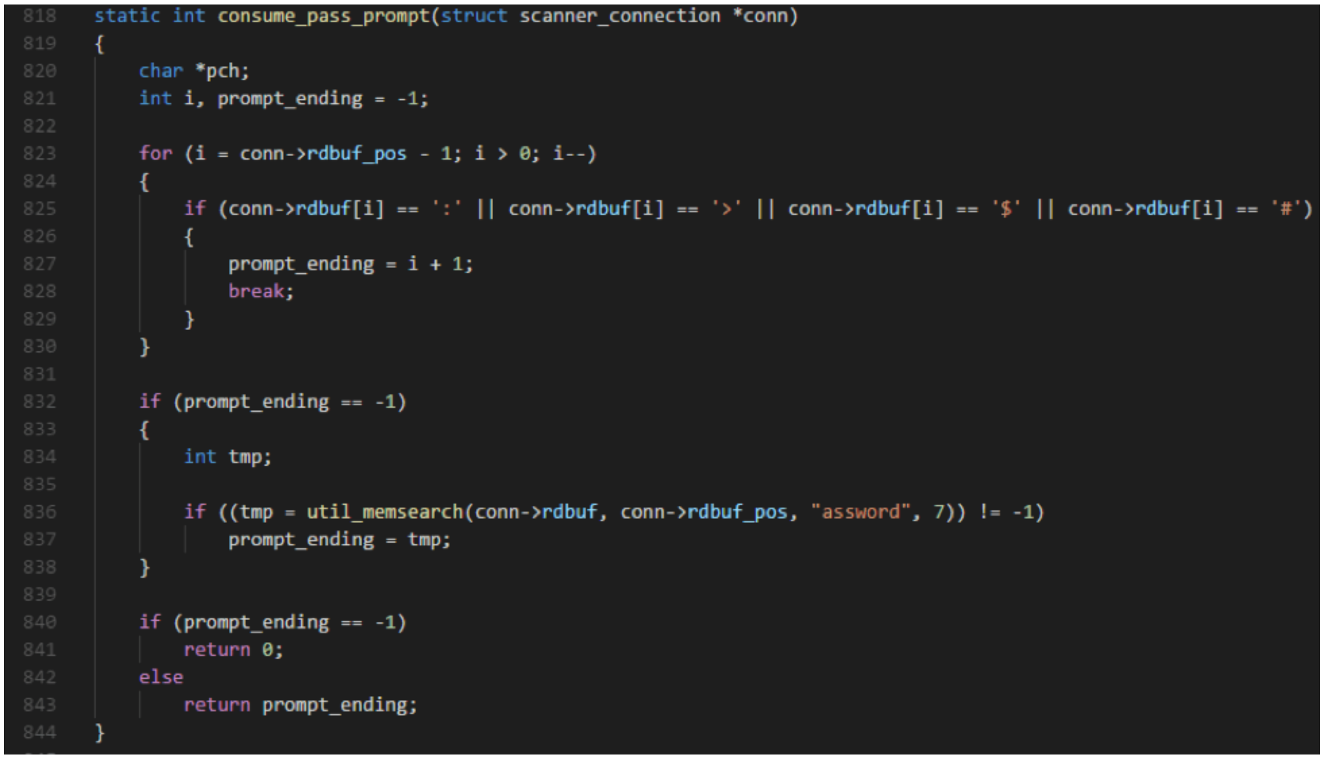
# Searching for the Reuse of Mirai Code: Hide ‘N Seek Bot

fortinet.com/blog/threat-research/searching-for-the-reuse-of-mirai-code--hide--n-seek-bot.html

April 16, 2018



```c
static int consume_pass_prompt(struct scanner_connection *conn)
{
    char *pch;
    int i, prompt_ending = -1;

    for (i = conn->rdbuf_pos - 1; i > 0; i--)
    {
        if (conn->rdbuf[i] == ':' || conn->rdbuf[i] == '>' || conn->rdbuf[i] == '$' || conn->rdbuf[i] == '#')
        {
            prompt_ending = i + 1;
            break;
        }
    }

    if (prompt_ending == -1)
    {
        int tmp;

        if ((tmp = util_memsearch(conn->rdbuf, conn->rdbuf_pos, "assword", 7)) != -1)
            prompt_ending = tmp;
    }

    if (prompt_ending == -1)
        return 0;
    else
        return prompt_ending;
}
```

Threat Research

By Jasper Manuel | April 16, 2018

It’s a common practice in mainstream software development to reuse codes that were made available on the internet. This practice is no different with malware development. Many malware source codes have been leaked and they enable many wannabe hackers and malware authors to learn and make their own malware.

In September 2016, the Mirai source code was leaked on the hacking community Hackforums. Mirai is known to have been used to temporarily cripple high profile services via massive distributed denial of service (DDoS) attacks. Since the release of this code online, many have tried to modify it and as a result many variants and derivations have emerged trying to get a slice of the IoT threat pie. On March 3, 2018, my colleague Dario Durando had the opportunity to present our research about these variants at the RootedCon Security Conference in Madrid, Spain. We identified variants that were derived from the original Mirai source code. From simple modifications (such as just adding additional credentials to the list of credentials available for a brute force attack) to more complex approaches (such as using exploits) to spread the malware to IoT devices, these variants all still use the main Mirai code base.

With this knowledge that leaked malware source codes are used by many malware authors in their own malware programming, we here at FortiGuard Labs became interested into searching out other malware that leverages Mirai code modules. Interestingly, one of the families that showed up in our search was the Hide ‘N Seek (HNS) bot, which was discovered in January of 2018. HNS is a complex botnet that uses P2P to communicate with peers/other infected devices to receive commands. In this article, I will discuss how the Mirai bot code was used in this HNS bot.

**A Quick Review of the Mirai Bot**

The original Mirai malware has the following components:

- Bot – infects and spreads to IoT devices through a brute-force attack and contacts the command and control server (C2) to receive commands from the botnet master/users to launch DoS attacks against specified targets.
- Command and Control server – used to control the infected IoT devices to launch DDoS attacks against specified targets.
- Report server – listens for reports from an infected IoT device to report that a new potential victim IoT device. This report contains the IP and login credentials of the new victim.
- Loader – loads the bot to the new victim device.

Fig 1. How Mirai Works

In this article, we will just focus on the bot. The Mirai bot has 3 main modules:

· Attack – the attack module contains various DoS attack methods (UDP, TCP, HTTP).

· Killer – kills processes (telnet, SSH, HTTP, other bots).

· Scanner – generates a list of random IP addresses to brute force to use within the botnet.

Fig 2. Mirai’s main modules

**The Hide ‘N Seek Bot and How Mirai Code Was Used**

HNS was discovered in January 2018. This IoT malware is more complex than Mirai in the sense that it communicates in a complex and decentralized manner (custom-built peer-to-peer (P2P) communication) in order to receive commands to perform its various malicious routines. HNS doesn’t launch DoS attacks, but was found to have the capability to exfiltrate data and execute additional code.

At first glance, it’s hard to notice that Hide ‘N Seek is using some Mirai modules, especially if you haven’t read the Mirai source code or haven’t analyzed Mirai binaries before. There are some awesome IDA plugins, however, that you can use to identify functions in a binary that have similar functions in another binary. However, they did not work for the samples I analyzed so I had to do the identification manually.

If you try to compile the Mirai source code, you will notice that its encrypted strings are stored in the read-only data segment (.rodata) of the compiled ELF binary. With this in mind, I started to check the .rodata segment of the HNS binary for possible encrypted strings.

Fig 3. .rodata segment containing possibly encrypted strings

When I checked the code reference to one of the sets of data, I was brought to the part of the binary where pointers to this data are passed as the first parameter to the same function. This gave me the idea that the function might be a decryption function.

Fig 4. Code snippet showing the pointers to the data passed as a parameter to the same function

Going into the function reveals that it is, in fact, a decryption routine. The decryption starts with a hardcoded single-byte XOR key. This key is XORed with the first byte of the string, and the result is then added to the key. The sum of the XOR result and the key is used as the key for the second byte. The same procedure applies to the next bytes. I saw two different keys (0xA (ARM) and 0xA0 (x64)) from the binaries I analyzed, but this key can be easily changed.

Fig 5. Decryption function

To make the string decryption easier, a simple IDA python script can be written to automatically find the addresses of the encrypted strings and then apply the decryption algorithm. We can also use it to add comments to contain the decrypted strings.

Fig 6. Decrypted strings

Now that we can see them, some of these strings look familiar. Some of these same strings are also found in Mirai.

Fig 7. The Mirai configuration table

With these strings decrypted, it’s easier to identify the Mirai functions used by HNS by just checking the references to these strings and then comparing them with the functions in the original Mirai source code.

Fig 8. The Mirai “consume_pass_prompt” function

Fig 9. The HNS “consume_pass_prompt” function implementation

While many functions look like they are directly copied, there are also many functions that were clearly modified to fit the needs of this new malware. The HNS bot has three main modules: a scanner, a process killer, and a function to wait for a connection from its peers.

Fig 10. 3 The main modules of HNS: scanner, killer, P2P

Two of the modules, the scanner and the killer, have a very similar code structure to that of the Mirai scanner and killer modules.

For the killer module, they both kill processes associated with other bot,s like QBOT, Zollard, and even Mirai itself.

Fig 11. HNS kills other bots

One difference is that HNS doesn't directly kill processes (by port number) related to HTTP, telnet, and SSH. Instead, the attacker can specify at runtime a port number. The process associated with this port will be killed.

Fig 12. HNS kills the process related to a port specified at runtime

For the scanner, they both generate a list of random set of IP addresses to search for potential victims. Major differences are around the ports to scan and compromise methods to be used. HNS scans ports 80, 8080, 2323, 9527, 23 randomly by initiating a raw socket SYN connection. Once a connection is established, like Mirai, it will try to brute-force its way into the device via telnet using a hardcoded list of credentials. Once successful, it can load itself to the device through several methods, such as echo, HTTP, and TFTP. Unlike with HNS, in the original Mirai the loader is a separate binary, while other Mirai modifications embed the loader into their body.

Fig 13. HNS randomly selects which port to scan

HNS uses at least three exploits on the samples I analyzed. Two are used for propagation. One of these targets Netgear DGN DSL modems/routers (also used by Reaper bot) while the other targets TP-Link routers. While the original Mirai doesn't use exploits to propagate, other modifications use other various exploits to spread.

Fig 14. Netgear DGN DSL modem/router exploit in the HNS implementation

Fig 15. TP-Link router exploit HNS implementation

After logging in successfully via telnet into a ZyXEL PK5001Z modem, the third one (CVE-2016-10401) is used to escalate the user to root using 'su' with password 'zyad5001'.

Fig 16. CVE-2016-10401 HNS implementation

**Other HNS Details**

This article doesn't focus on the HNS malware itself. Instead, this article describes how Mirai code was used in HNS. This allows us to study how we can use this technique to hunt for other malware that uses Mirai code. However, some very interesting features of HNS must also be mentioned here. For example, HNS uses a custom-built P2P communication with peers and/or other infected devices using a randomly generated UDP port or a port specified at runtime. Unlike Mirai, which was designed to launch DoS attacks against certain targets, this IoT malware receives commands through peers to exfiltrate data and execute additional code. This P2P communication makes the malware more complicated to analyze. The decentralized manner of receiving commands also makes it hard to identify where the commands are issued from and where the data drop points are located.

Fig 17. Code snippet showing P2P commands

**Conclusion**

As we have seen in the past, malware source code leaks result in more malware being created. We are now seeing that the Mirai source code leak is going through this same process, and so we expect to see new malware families emerge that leverage the Mirai source code.

As always, by using the knowledge we gained from this study, we here at FortiGuard Labs will continue to watch, and even hunt for malware that uses the Mirai source code.

**Solution**

Fortinet detects the HNS samples as Linux/Hns.A!tr and the exploits used as ZyXEL.PK5001Z.Modem.Backdoor, NETGEAR.DGN1000.CGI.Unauthenticated.Remote.Code.Execution, and TP-Link.Wireless.Router.Backdoor .

-= FortiGuard Lion Team =-

Samples analyzed:

8cb5cb204eab172befcdd5c923c128dd1016c21aaab72e7b31c0359a48d1357e (x64)
095c13175e0908e67289bd5c619745ea905a73600ccb9c3f12df3e1c018e1346 (ARM)
2da20a90a52e51897113438ac819362e5e04f8a7435c578d7d306afb482ac71e (MIPS)

*Check out our latest Quarterly Threat Landscape Report for more details about recent threats.*

*Sign up for our weekly FortiGuard intel briefs or for our FortiGuard Threat Intelligence Service.*

## Related Posts