Brief Overview of the ShadowHammer and ShadowPad attacks

José Manuel Agüero Trejo

***Abstract*—A supply chain attack is a method attack that targets certain elements of an interconnected network in order to affect a related member down the supply chain structure involved. This report describes two recent supply-chain attacks.**

***Index Terms*—supply-chain, cyber-security, ShadowPad, ShadowHammer**

I. ShadowHammer

In January 2019 researchers at Kaspersky Lab found a security breach involving ASUS Live Update Utility that affected over 57,000 users [**?**].

This utility was used to install a backdoor for the distribution of malware possessing a signature from a legitimate ASUS digital certificate. As this tool is pre-installed on most ASUS computers, the attackers gained access to a very large userbase.

The victims were reached by utilising a hard-coded list of addresses integrated into trojanized software and a target pool of unknown users identified by their network adapters MAC addresses.

Legitimate certificates were used to to sign the trojanized updaters; furthermore, the malware would not take action on systems that were not labelled as primary targets in order to remain undetected.

In response, ASUS has released an updated version of this tool that introduced several security improvements to prevent similar attacks from occurring [**?**].

A. ASUS Live Update Tool

ASUS Live Update Tool is a pre-installed utility in charge of guaranteeing that the latest drivers and firmware are being used by the users. This is done by detecting whether any new software versions has been released on the ASUS website and updating the BIOS, drivers, and applications accordingly. [**?**].

B. Mode of operation

Access to an ASUS Live Updater file that contained a digital signature was obtained by one of the techniques described below; a trojanized version of this file was found by the researchers.

1) The attackers replaced a function in the binary file of an early variant of the ASUS Live Updater.
   The new function utilised a hardcoded size and offset values to be copied into the heap memory in order to allocate new memory. An additional offset, injected into the newly allocated memory, was hardcoded as a trigger to start the backdoor; an offset would then point to a position-independent shellcode that further unwraps the malicious code.
   The attackers overwrote a small block of resource contents in the executable file while carefully ensuring that the size of the binary was retained. This would serve as a malware downloader with legitimate certification.
2) The attackers patched a C runtime library function to execute a shellcode loader in order to pass the execution flow to a chosen address that contained a decryption routine. Then, this routine copies another block of encrypted shellcode to a newly allocated memory block with read-write-execute attributes available upon decryption.

Once the binaries have been modified the trojan downloader takes action.

A dynamic import paradigm with a simple hashing algorithm was implemented to collect the MAC addresses of all the available network adapters. Then, the MD5 hash resolved is compared against the list of target hardcoded values. Once the target MAC addresses have been identified, a binary object is downloaded and the first match hash value entry is sent to the server; this is done to obtain a response from the server consisting of an executable shellcode that will be allocated in the memory assigned in the previous step.

A key element of the success of this operation was the attack on the authentication, as the signature of a trusted authority (ASUS) was compromised.

Two different legitimate certificates were utilised to sign the backdoored binaries; these certificates were widely used in a variety of files, thus revoking them entailed a high level of difficulty. A signing timestamp was found to be absent in these binaries in an attempt to hide the time of execution. However, a mandatory Certificate Validity Period allows for the time-frame to be estimated.

A relation with the previous attacked ShadowPad was speculated by the Kaspersky Lab researchers.

II. ShadowPad

Kaspersky Lab's Global Research and Analysis team discovered a backdoor into a widely used server management software product after analysing suspicious Domain Name Server requests observed in a financial transactions processing system [?].

As in the more recent attack described above, the malware was certified by an authority trusted by a large variety of companies ranging from the education, manufacturing and telecommunications industries among many others.

This vulnerability allowed to access and steal data from the large user base of the trusted vendor NetSarang. Furthermore, these malicious requests were generated as a consequence the hidden activity from a recent legitimate software module which would further deploy when an user of interest was found.

The malicious code had the capability of silently infiltrating the target users computer in order to download and execute malware.

NetSarang reacted by promptly releasing a safe updated version of the software. However the system is suspected to remain dormant on many systems; moreover, as previously mentioned, researchers believed more complex versions of ShadowPad were implemented in waves of parallel attacks linked to ShadowHammer.

A. Mode of operation

The platform is designed to deploy in two main stages that aim to gain flexible remote control capabilities [?].

During the first stage, shellcode is embedded into a digitally signed library called NSSOCK2.DLL with the purpose of connecting and gaining configuration information of the command and control validation server (C&C).

Once this is achieved the second stage is triggered; plugins are loaded and injected into the memory of different processes by utilising the DNS protocol and operating the C&C modules. This enables different types of data extraction and manipulation.

These stages are described in detail below:

1) : The main loader in the compromised library is implemented by decrypting a binary function during the auto-initialisation process that is automatically called by the C runtime code. This function then translates into shellcode in order to reallocate code and resolve imported API functions; a UNIX-like timestamp is implemented by the shellcode to remain undetected and an entry point is generated and exploited using a standard Microsoft Visual Studio code.

A unique user id, decryption key, first execution time and execution counter is produced and contained into a block; all generated strings are then encrypted utilising a custom randomisation function and stored into a 552 byte long value. Then, a domain generation algorithm is implemented to create a hostname for accessing the C&C server. Finally DNS requests are sent every 8 hours utilising the information extracted from the network adapter settings.

The module encrypts the request buffer and converts it to a readable format until data has been received or a time out response has been obtained; a XOR-based encryption scheme is used for this purpose. The response is sent to the C&C configuration storage and the registry key is updated.

To finalise this stage a proper key is read or a decryption protocol is implemented utilising the DNS response and the additional data received from the C&C is passed to the second encrypted shellcode presented during stage 2.

2) : The shellcode produced for this stage was generated form a Windows DLL file and uses the previously mentioned Microsoft Visual Studio code to exploit the entry point; such code differs from the standard implementation and produces further *Reason* parameters used to run a custom plugin API.

During the initialisation, memory is allocated for internal structures and a function table is set up. Then the plugin infrastructure is established and a sequential call to the entry point with the additional parameters is executed in order to copy the data from the plugins. Once this has been done a thread is started to monitor registry key changes and load any new available plugins; the API is used to initialise these changes from the root module. An API for reading, writing and deleting arbitrary registry values is also provided by this process.

Finally, a configuration module is overwritten with one of the packets used to activate the second stage shellcode and a connection is mantained with the C&C server utilising one of the initialised plugins. This is done to continuously get and execute commands and send result packets consisting of a shutdown message to finalise the current established session or a plugin ID with additional command data.

All together, this enables to establish a modular backdoor platform capable of maintaining a Virtual File System inside the registry as well as downloading, executing and modifying arbitrary code provided by the C&C server. The VFS and all additional files generated are encrypted and stored ensuring that the location varies from target to target. Furthermore, the domain generation is dynamic to add an extra layer of detection avoidance.

This encrypted payload is capable of being activated remotely due to the connection protocol implemented where a special packet triggers the malware execution, leaving it dormant when no malicious activity is intended at a certain point in time.

When the attack is triggered an URL request yields a response that contains the special character $ and decodes the corresponding string into a binary buffer that is decrypted in order to obtain the real URL to be used by the C&C.

Recent updated versions of ShadwPad follow the same principle of unwrapping the code before activating a system of plugins that serves as bootstrap for the malware to attack.

All together, legitimate requests from a trusted authority are received by the victim to allow for infiltrating the target system, uploading files, creating processes and storing information.