Grasshopper Module Guide – Buffalo v1.0 and Bamboo v1.0

June 2012

CL BY: 2355679
CL REASON: Section 1.5(c),(e)
DECL ON: 20370522
DRV FRM: COL 6-03

1 Overview

Buffalo and Bamboo are related persistence modules that use hosted DLLs inside a Windows Service to persist a payload. Bamboo additionally uses “service hijacking” to run immediately on installation while Buffalo will require a reboot. When a payload is chosen that uses either Buffalo or Bermuda, the module will install a stub service DLL and deploy the payload to the target.

Both Buffalo and Bamboo support 32- and 64-bit EXE, DLL, and GH1 payloads. The bitness of the stub and DLL, GH1 payloads must match the target OS. A 32-bit EXE payload may be installed on a 64-bit target, but not vice versa.

2 Installation

Buffalo and Bamboo use direct registry modification to register a stub as a service DLL in the netsvcs service host. If the module fails to install the payload, it will delete any deployed components and remove the registry modifications.

Bamboo will use a service hijacking technique to run the payload immediately after installation (see section on Service Hijacking).

2.1 Configuration

Field	Default	Description
Service Name	*None*	Overt key value for service stored in registry
Service DLL Path	*None*	Path to service DLL Stub on target If the path does not exist, it is created.
Payload Path	*None*	Path to Payload on target, executed by service DLL If the path does not exist, it is created.
Display Name	*None*	Overt name of service displayed by Windows Services MMC
Description	*None*	Overt description of service displayed by Windows Services MMC
Unhijack DLL Path	*None*	Path to temporary storage of unhijacking DLL (Bamboo Only)

2.2 Service Hijacking

After a service DLL is installed, it is unable to start before a reboot. A technique to start a service DLL immediately after installation is called service hijacking. In this method, an existing, but not currently running, service is temporarily hijacked to run our own stub, then reset after the stub has started.

Service hijacking begins by identifying a service DLL entry that is manual start (not automatically started on boot) and is not currently running. Registry values for the identified service are redirected to our own service DLL. When we start the hijacked service, Windows will load our own service DLL. After our service DLL is started, the registry values for the hijacked service are restored.

Windows svchost caches important registry information about services it has started. In order to completely unhijack the service, we load a support DLL to locate and clear the hijacked service's cache entry. Consequently, if the hijacked service has already been started, the hijack will fail and the payload will not start until reboot or until svchost restarts.

3 Payload Execution

Whenever the system starts, the Windows OS will run the Buffalo/Bamboo service DLL stub. The stub is executed from the netsvcs svchost process with SYSTEM privileges. The behavior of the stub depends on the payload type. Buffalo and Bamboo support three kinds of payload: EXE, DLL, GH1.

3.1 EXE and DLL

If the payload is an EXE or DLL, the stub is configured with the path to the payload and the name of the service that identifies it. Up execution by Windows services, the stub will run the payload.

If the payload is an EXE, Buffalo/Bamboo will execute it with SYSTEM privileges and terminate. If the payload is a DLL, the stub will call LoadLibrary() and begin monitoring the payload.

If the stub is unable to locate or start the payload or if the payload disappears, it will uninstall. During uninstallation, Buffalo/Bamboo will delete the payload, remove the service, and self delete the stub.

The EXE or DLL payload is responsible for deleting itself from the target to trigger uninstallation.

3.2 GH1

If the payload implements the GH1 interface, Buffalo/Bamboo embeds the payload as a resource in the stub and configures the stub with the name of the service that identifies it. Upon execution, the stub will load the payload DLL in memory.

The stub will uninstall itself on demand or failure to start the payload. During uninstallation, Buffalo/Bamboo will remove the service and self delete the stub and payload.

4 Footprint

Buffalo/Bamboo writes unobfuscated binaries to the target filesystem. The service DLL stub is written to the filesystem at a user-specified path. If the payload is an EXE or DLL, it is written to the filesystem at a user-specified path. If Bamboo is used, the unhijack support DLL is written to the filesystem at a user-specified path and promptly deleted after installation.

If the payload is an EXE, the process of the payload executable is visible in the Task Manager during execution.

Buffalo/Bamboo will create a service visible in the Services view of the Microsoft Management Console with the user-specified display name and description.

A registry key will be placed in `HKLM\SYSTEM\CurrentControlSet\services\<ServiceName>` holding the description and the path to the service stub.

The Service Name will be placed in a registry `REG_MUTLI_SZ` value at `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost\netsvcs`. Note: this value is distinct from the subkey of the same name.

5 Receipt XML Format

Buffalo/Bamboo's configuration is recorded in the Grasshopper receipt at build time under `build.xml`. An example and description of the xml format is provided below.

5.1 XML Examples

```
<PersistModule>
   <UUID>9d03da02ab3a47d7bd28c9a776ba9806</UUID>
   <ServiceDll>
      <ServiceName>Cover Name</ServiceName>
      <ServiceDllPath>C:\Test\stub.dll</ServiceDllPath>
      <PayloadPath>C:\Test\payload.exe</PayloadPath>
      <DisplayName>Cover Name</DisplayName>
      <Description>This is a description.</Description>
   </ServiceDll>
</PersistModule>

<PersistModule>
   <UUID>9d03da02ab3a47d7bd28c9a776ba9806</UUID>
   <ServiceDllHijack>
      <ServiceName>Cover Name</ServiceName>
      <ServiceDllPath>C:\Target\stub.dll</ServiceDllPath>
      <PayloadPath>C:\Target\payload.dll</PayloadPath>
      <DisplayName>Cover Name</DisplayName>
      <Description>This is a description.</Description>
      <UnhijackDllPath>C:\Target\unhijack.dll</DllPath>
   </ServiceDllHijack>
</PersistModule>
```

5.2 Field Definitions

UUID

The universally unique identifier for the module variant used in the build.

ServiceDll

The service DLL configuration information used by the Buffalo module.

ServiceDllHijack

The service DLL and hijacking configuration information used by the Bamboo module.

ServiceName
The overt name of the service created by the module. The service name is used as the key in the registry.

ServiceDllPath
The path to the Buffalo/Bamboo service DLL stub started by Windows as a service.

PayloadPath
The path to the payload on the target run by the Buffalo/Bamboo stub.

DisplayName
The overt name of the Windows service created by the module.

Description
The overt description of the Windows service created by the module.

UnhijackDllPath
The path to the unhijacking helper DLL used by the Bamboo module.

6

Appendix A: Change Log

Date	Change Description	Authority
05/2012	Document Initialization	2355679
09/2012	Update for Grasshopper v1.0 Phase 2 Delivery	2355679
11/2012	Update for Grasshopper v1.0.1 Delivery	2355679