Feature, not bug: DNSAdmin to DC compromise in one line

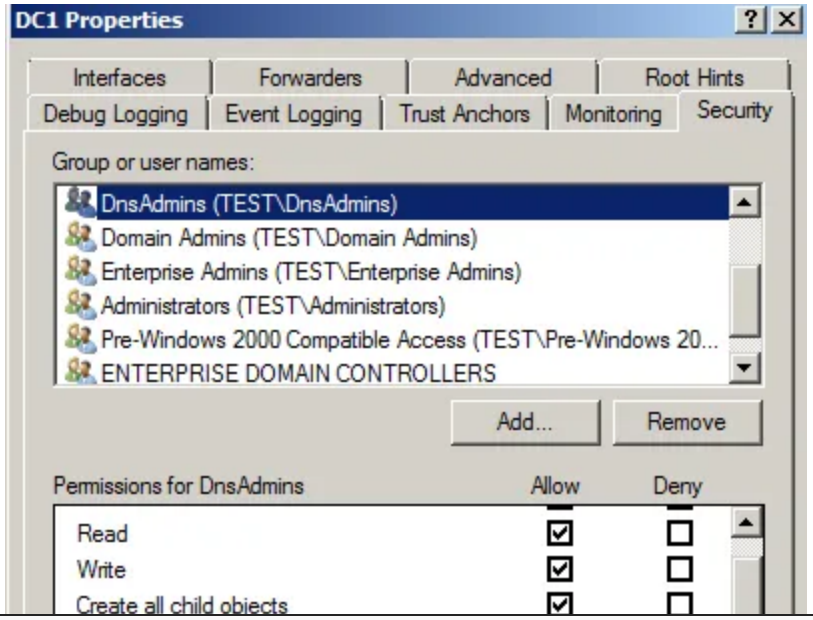
Shay Ber · Follow

as confirmed with Microsoft, it's still a cute trick which can be useful as an AD privilege escalation in red team engagements.

All presented information was gathered by reading the protocol specification ([MS-DNSP], https://msdn.microsoft.com/en-us/library/cc448821.aspx) and reverse engineering the dns.exe binary using IDA.

DNS Server Management Protocol basics

The management protocol layers on top of RPC, which for this protocol's



This is where the protocol specification comes to our aid. Section 3.1.4: Message Processing Events and Sequencing Rules, basically details all operations which the server needs to support. The first one is R_DnssrvOperation, which contains a pszOperation parameter, which determines the operation performed by the server. While scrolling over the huge list of possible pszOperation values, we see this:

ServerLevelPluginDll	On input **dwTypeId** MUST be set to DNSSRV_TYPEID_LPWSTR, and **pData** MUST point to a Unicode string that contains an absolute pathname for server side plug-in binary on the DNS server or an empty Unicode string.

someone must have dug this up before. A google search for ServerLevelPluginDll raised nothing of the sort, however it did pop up the useful dnscmd command line tool, which I hadn't known before.

Luckily, dnscmd already implements everything we need. A quick look at its help message and another glance at https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/dnscmd give us the following option:

```
dnscmd.exe /config /serverlevelplugindll \\path\to\dll
```

Great. Now, for testing purposes, we restart the DNS server service. Whoops — it fails to start, and clearing the registry key value allows it to start. Apparently it needs something more from our DLL. Time to open IDA.

There are several possibilities to quickly reach the functionality we seek to reverse in this case — searching for relevant strings and searching for relevant APIs are usually the easiest and quickest. In our case, going through all xrefs to LoadLibraryW or GetProcAddress gives us what we need — going through the code of the function which LoadLibraryW's our DLL and the one function it is called from, we see that no validation is performed at all on the

```
  g_dllDnsPluginCleanup = GetProcAddress(hLib, "DnsPluginCleanup")
  if (!g_dllDnsPluginCleanup) {...log and return error...}

  if (g_dllDnsPluginInitialize){
    g_dllDnsPluginInitialize(pCallback1, pCallback2);
  }
}
```

Here's a quick PoC to demonstrate how the code for such a DLL should look under Visual Studio 2015:

access for the Everyone SID should do the job), and we can run code as SYSTEM on a domain controller, thus taking control of the domain.

While this demonstrates that it is possible to take over a domain if you're a member of DnsAdmins, it's not limited to just that — all we need to successfully pull off this trick is an account with write access to a DNS server object. The ACLs for these objects are usually, from my experience, not kept as clean or monitored as ACLs for domain admins (or similar groups protected by AdminSDHolder), thus offering a nice chance for a small domain elevation of privilege.

Disclosure timeline

Mar. 31st — Initial disclosure to secure@microsoft.com.

Apr. 1st — Disclosure acknowledged and forwarded for review.

Apr. 8th — MSRC case 38121 opened.

Small update (May 10th): Nikhil Mittal has elaborated on the precise technicalities of getting this feature to work in his lab, here. Thank you!

DNS Red Team Microsoft Active Directory