Win32/Sality newest component: a router’s primary DNS changer named Win32/RBrute

By Benjamin Vanheuverzwijn

Archived: 2026-04-06 03:26:28 UTC

Win32/Sality is a family of malware that has been using a peer-to-peer botnet since at least 2003. It is a file infector and a trojan downloader, the latter of which is mainly used to send spam, although it has been used for different purposes such as faking advertising network traffic, distributed denial of service or VoIP account cracking. All commands and files exchanged through Sality's P2P network are digitally signed, making it resilient to protocol manipulation. Its modular architecture as well as the longevity of the botnet shows good programming practice and an efficient software design.

We’ve been tracking Win32/Sality network for quite some time now and seen more than 115 000 IP addresses reachable from the Internet using so-called “super peers,” which keep the botnet alive and propagate commands to regular peers.

We have seen the same components downloaded over the years with little change to their underlying behavior. Lately, a new component has now appeared with some novel characteristics: the ability to change a residential broadband gateway router's primary DNS address, which is different from the usual FTP password stealer or spambot deployed by Win32/Sality. According to our telemetry data, this component was dropped for the first time at the end of October 2013. It was first publicly discussed by Dr. Web, who has published a technical analysis of one component, the IP address scanner. They named it Win32/RBrute.

This blog will contain

- An overview of the infrastructure supporting the primary DNS changer component
- A technical analysis of the two binaries that support the operation
- A brief analysis of the spread of the operation
- A review of the similarities between the DNS changer component and the other components dropped by Win32/Sality

A new purpose: changing a router's primary DNS

This feature adds a new dimension to the Win32/Sality operation. The first component, detected by ESET as Win32/RBrute.A, scans the Internet for router administration pages in order to change the entry for their primary DNS server. The rogue DNS server redirects users to a fake Google Chrome installation page whenever they are trying to resolve domains containing the words “google” or “facebook”. The binary distributed through this installation page is in fact Win32/Sality itself, providing a way for the Sality botnet's operators to increase its size further by infecting other users behind this router.

The IP address used as the primary DNS on a compromised router is part of the Win32/Sality network. In fact, another malware, detected by ESET as Win32/RBrute.B, is installed by Win32/Sality on compromised computers and can act either as a DNS or a HTTP proxy server to deliver the fake Google Chrome installer.

The Operation

Far from being a new technique these days , changing the primary DNS on a router is quite in vogue right now for everything from the theft of bank credentials to blocking communications with security vendors, especially with recent reports of vulnerabilities in different router’s firmware.

Win32/RBrute.A tries to find the administration web pages for routers by downloading a list of IP addresses from its C&C server to scan and then reporting back its findings. At the time of our investigation, Win32/RBrute.A targeted the following routers:

- Cisco routers matching “level_15_” in the HTTP realm attribute
- D-Link DSL-2520U
- D-Link DSL-2542B

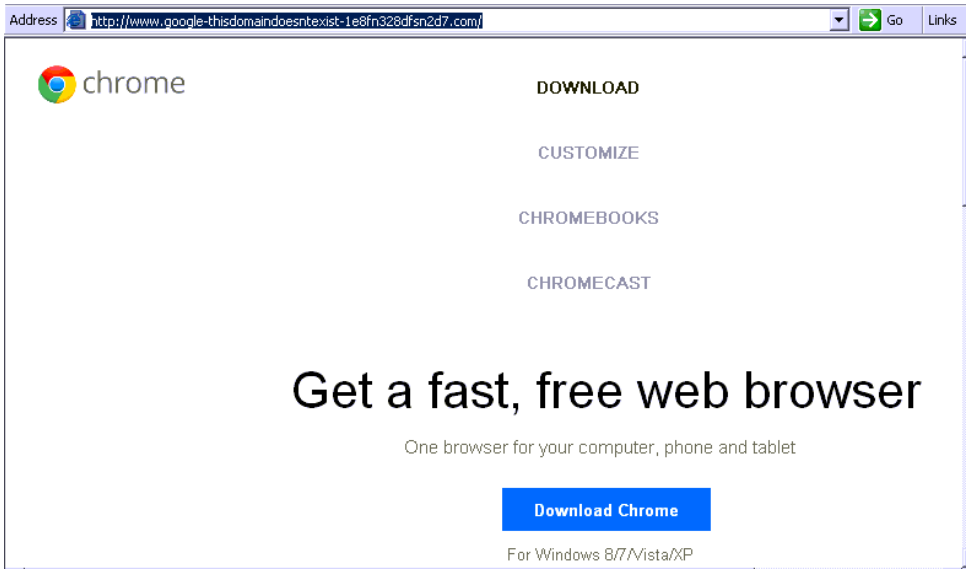
- D-Link DSL-2600U
- Huawei EchoLife
- TP-LINK
- TP-Link TD-8816
- TP-Link TD-8817
- TP-Link TD-8817 2.0
- TP-Link TD-8840T
- TP-Link TD-8840T 2.0
- TP-Link TD-W8101G
- TP-Link TD-W8151N
- TP-Link TD-W8901G
- TP-Link TD-W8901G 3.0
- TP-Link TD-W8901GB
- TP-Link TD-W8951ND
- TP-Link TD-W8961ND
- TP-Link TD-W8961ND
- ZTE ZXDSL 831CII
- ZTE ZXV10 W300

If a web page is found, the C&C sends a short list of about ten passwords to the bot and instructs it to perform a brute force password guess attack against the router. If the bot is able to log in to the router, it will then proceed to change the router's primary DNS server settings. It is interesting to note that only brute force attack is used to gain access to the router's administration portal; no exploit code is used. The authentication is done with usernames of "*admin*" and "*support*", although previous versions also tried "*root*" and "*Administrator*". Below is a list of passwords we have observed being transmitted from the C&C:

- *<empty string>*
- *111111*
- *12345*
- *123456*
- *12345678*
- *abc123*
- *admin*
- *Administrator*
- *consumer*
- *dragon*
- *gizmodo*
- *iqrquksm*
- *letmein*
- *lifehack*
- *monkey*
- *password*
- *qwerty*
- *root*
- *soporteETB2006*
- *support*
- *tadpassword*
- *trustno1*
- *we0Qilhxtx4yLGZPhokY*

In the event of a successful login, the malware changes the primary DNS server to a rogue one, reports a successful infection to the C&C, and continues with scanning the Internet.

Once a router's primary DNS address is compromised, all DNS queries made by users will go through the rogue DNS server, modifying them to point to the fake Chrome installer page whenever "facebook" or "google" domains are resolved.



This example shows a successful redirection for a domain that is not registered but contains the word “google”.

This operation is somewhat similar to DNSChanger, which drove users to install fake software to further spread malware using a rogue DNS service.

Once a computer is infected by running the fake Google Chrome installer, its primary DNS server will be changed to “8.8.8.8” by updating the following registry key:

HKLM/SYSTEM/ControlSet001/Services/Tcpip/Parameters/Interfaces/{network interface UUID}/NameServer = “8.8.8.8”

It should be noted that the IP address "8.8.8.8" belongs to Google Public DNS, a legitimate domain name service operated by Google, and it is not involved with Win32/RBrute.

Since infected PCs will no longer be using the router’s DNS server, they will no longer be affected by its bogus redirections. On the other hand, the router is still compromised and will nag each computer trying to resolve "facebook" or "google" domains through its DNS service until they are infected with Win32/Sality. This tactic is far from stealthy and in fact tries to annoy the user into infecting its system or simply breaking “google” and “facebook” domains for operating systems that are not targeted (e.g. Linux).

Currently, the goal of this operation appears to be solely to increase Sality’s botnet size.

Technical Analysis

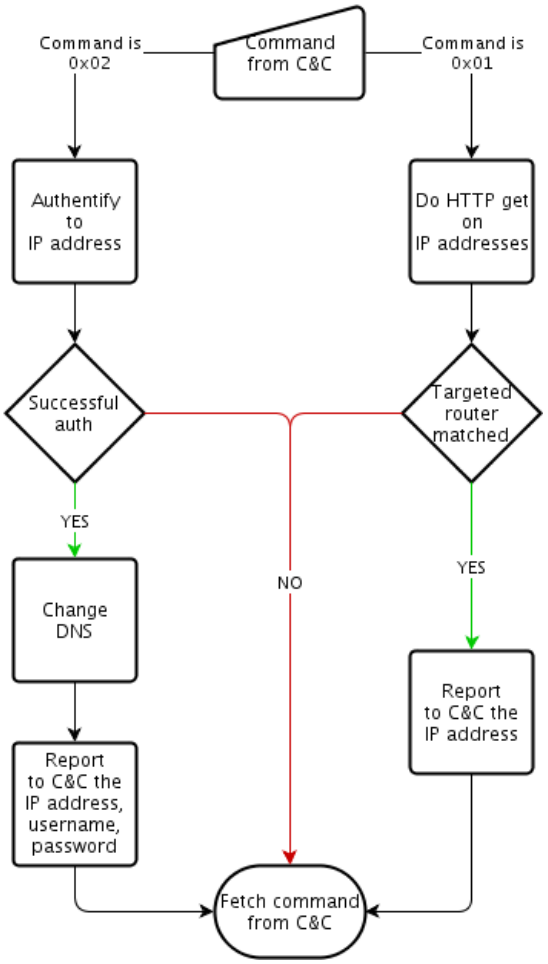
Win32/Sality’s DNS changer component is composed of two binaries: a router scanner and a DNS / HTTP server. Both malware are dropped by Win32/Sality.

Router Scanner Binary - Win32/RBrute.A

At the beginning of the execution, the malware creates a mutex with the name “*19867861872901047sdf*” to avoid running multiple instances.

It then checks a hard-coded IP address every minute to fetch a command; that command is either a scan instruction or a request to try to log onto an IP address to change the primary DNS.

A scan instruction comes with an IP address to start and the number of addresses to try. Win32/RBrute.A will try to do a **HTTP GET** on **TCP/80**, hoping to receive a HTTP Error 401 - Unauthorized. The router model is extracted from the realm attribute of the HTTP authentication schemes. If a targeted router is found, the malware sends back its IP address to the C&C.



Win32/RBrute.A flowchart

The C&C will then issue a request to login to the router using a password provided by the C&C. If the login is successful, the primary DNS server is changed in the router to a host running the Win32/RBrute.B malware.

DNS and HTTP Server Binary - Win32/RBrute.B

This component is divided in three parts: the control thread, the DNS server thread, and the HTTP server thread.

Although both the DNS and the HTTP server thread can be used at the same time, the malware will choose, based on a random value, to be either a DNS or a HTTP server. A constant in the formula ensures that 80% of the infections will act as DNS servers, although we've seen this constant set to 50% at the beginning of the operation.

```
89    if ( !g_mode )
90    {
91      random = get_random() % 1000;
92      rand_less_than_800 = random < 800;
93      LOBYTE(rand_less_than_800) = random <= 800;
94      --rand_less_than_800;
95      LOBYTE(rand_less_than_800) = rand_less_than_800 & 206;
96      g_mode = rand_less_than_800 + 100;
97    }
98    CreateThread(0, 0, thread_1_thread_cleaner, 0, 0, 0);
99    CreateThread(0, 0, thread_2_administration, 0, 0, 0);
100   if ( g_mode == DNS_SERVER )
101   {
102     CreateThread(0, 0, thread_3_dns, 0, 0, 0);
103   }
104   else if ( g_mode == HTTP_SERVER )
105   {
106     CreateThread(0, 0, thread_4_httpd, 0, 0, 0);
107   }
108   Sleep(10000u);
```

Choosing the DNS or HTTP server thread randomly.

If the chosen server thread would not start, the malware will fall back to the other mode:

```
109     if ( thread_started )
110       goto sleep_forever;
111     if ( g_mode == DNS_SERVER )
112     {
113       g_mode = HTTP_SERVER;
114       CreateThread(0, 0, thread_4_httpd, 0, 0, 0);
115     }
116     else if ( g_mode == HTTP_SERVER )
117     {
118       g_mode = DNS_SERVER;
119       CreateThread(0, 0, thread_3_dns, 0, 0, 0);
120     }
```

Fallback to the other mode if the choosen thread could not start.

The operator can also force a thread to start by sending a specially crafted DNS or HTTP request. A mutex with the name "*SKK29MXAD*" ensures that only one instance can run on the host.

Control Thread

The control thread is used to report back to the C&C and to reconfigure the server instance.

Every two minutes, the malware will send a packet to a hard-coded IP address containing information about the machine on which it is running. The C&C will then answer with an IP address that will be used to deliver the infected Chrome installation. If the malware is in DNS mode, the IP address served by the C&C will be that of a rogue HTTP server installed on a Sality-compromised machine. In the other case, the C&C will send the IP address of a server outside of Sality's P2P network, which will be serving the fake Chrome installation page.

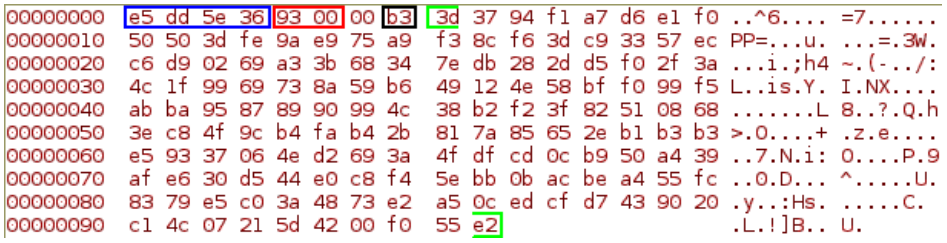
Listed below is the host information sent by the control thread to the C&C:

- Computer name - *GetComputerName()*
- Local time - *GetLocalTime()*
- Country - *GetLocaleInfoA()*
- Windows directory - *GetWindowsDirectoryA()*
- Windows product name - from the registry key "*HKEY_LOCAL_MACHINE/SOFTWARE/Microsoft/Windows NT/CurrentVersion/Product Name*"
- CPU names - from the registry keys "*HKEY_LOCAL_MACHINE/HARDWARE/DESCRIPTION/System/CentralProcessor/<CPU #>/ProcessorNameString*"
- Memory stats - *GetMemoryStatusEx()*
- Result of *IsDebuggerPresent()*
- Memory usage of the malware - *GetProcessMemoryInfo()*
- Uptime of the malware - in minutes
- Number of threads n use

The information packet has the format:

```
0x00    DWORD       Checksum (CRC32)
0x04    WORD        Payload length
0x06    BYTE        Unused
0x07    BYTE        Mode (HTTP - 0x32 or DNS - 0x64)
0x08    BYTE[]      Host information
```

The screenshot below shows an information packet sent to the C&C.



Host information packet sent to the C&C.
In blue: payload checksum, in red: payload length, in black: encrypted server mode, in green: encrypted host information

The host information, green in the previous example, is the following string encrypted with RC4:

9BC13555|24.03.2014 21:56:27|United States|C:\WINDOWS|Microsoft Windows XP|proc#0 QEMU Virtual CPU version 1.0|1|358|511|1117|1246|0|2|0|0|

The C&C will then answer with a packet with the service IP address to use:

```
0x00   DWORD        Checksum (CRC32)
0x04   WORD         Payload length
0x06   BYTE         Unused
0x07   BYTE         Command (start - 0x02 or stop - 0x03 the service)
0x08   DWORD        Service IP address (Win32/RBrute.B HTTP server or
rogue HTTP server)
```

DNS Server Thread

The DNS server looks for requests that contain “google” or “facebook” in the domain name. If it finds one, the DNS response it will send back will contain the IP address of a Win32/RBrute.B HTTP server on the Sality network. If the query doesn't contain “facebook” or “google”, it will relay the query to Google’s DNS servers (“8.8.8.8” or “8.8.4.4”) and will forward the response to the client.

Sending a packet to the server on **UDP/53** with “0xCAFEBABE” as the payload will set the “*udme*” flag in the Windows registry key "*HKEY_CURRENT_USER/SOFTWARE/Fihd4*". This flag ensure that the DNS server thread will start at the next reboot, overriding the random process. The server will reply “0xDEADCODE” to confirm the command.

HTTP Server Thread

When receiving a browser request by a user that has been redirected, the HTTP server thread will first look at the browser User-Agent and will have a different behavior consequently.

If the User-Agent contains “linux” or “playstation”, the server will silently drop the connection (how rude!). If the User-Agent makes reference to a mobile (matching one of the following words: “android”, “tablet”, “Windows CE”, “blackberry” or “opera mini”), the server will serve Win32/Sality (!) malware 5% of the time even though these are mobile devices User-Agent; otherwise, the request is dropped. Finally, if the User-Agent contains “opera”, “firefox”, “chrome”, “msie” or anything else, the user will be served the Win32/Sality.

The User-Agent will affect the port on which the query is made on the rogue HTTP server distributing the malware.

User-Agent	Port used
Android, tablet, windows ce, blackberry, opera mini	8979
Linux, playstation	None
Opera	4979
Firefox	5979
Chrome	6979
Msie	7979
Others	6979

Any **HTTP GET** request sent to these ports will serve the fake Chrome installation page… even if you're browsing with Chrome!

Akin to the DNS server thread, the botmaster can affect the HTTP server behavior by sending a specially- crafted HTTP packet. Specifically, sending a GET or POST request with the User-Agent "*BlackBerry9000/5.0.0.93 Profile/MIDP-2.0 Configuration/CLDC-2.1 VendorID/831*" will set the "*htme*" flag in the registry key "*HKEY_CURRENT_USER/SOFTWARE/Fihd4*", effectively ensuring that malware will start the HTTP server thread upon reboot, overriding the random process. The server will send "*<html>kenji oke</html>*" to confirm a successful execution.

The HTTP server also keeps a list of allowed files to be served. If a browser makes a HTTP query on a domain matching "google" or "facebook" to a file not in the list, the server will reply with a HTTP 200 OK, with the following payload:

<html><meta http-equiv="refresh" content="0; url=/"></html>

redirecting the browser to the front page -- hence serving the fake Chrome installation page. For example, if the user browses to "*http://google.com/does-not-exist*" and "*does-not-exist*" is not in the allowed files list, the user will be redirected to "http://google.com" instead of having the usual HTTP 404 error.

We should also note that every **HTTP GET** query made on the HTTP server that contains the string ".exe" will be forwarded to the rogue HTTP server, regardless of the allowed files list. The rogue server will always answer with an infected binary.

Similarities with other Sality components

Based on the following observations, we believe that the main file infector as well as all the components previously described are all developed by the same group of people. By looking at each of the components binaries, they all share the same technical details and coding style.

No persistence needed

None of the dropped Sality components, including those discussed before, needs a way to be persistent across system reboot, although some modules might store configuration in the registry. They are always downloaded and launched by the persistent layer: the file infector.

Buffer Initialization

The operators have the standard practice of initializing their buffers with the '0' value. The compiler "visual-c++" doesn't optimize the following C code when the operators compile a software:

char buf[4096] = {0};

This is compiled into the code displayed in the following screenshots.

```
mov     [ebp+buf], 0
mov     ecx, 3FFh
xor     eax, eax
lea     edi, [ebp+buf+1]
rep stosd
stosw
stosb
```

Un-optimized initialization of a buffer of size 4096 bytes.

This assembly stub is seen in every component dropped by Win32/Sality.

Bypassing the Firewall

All components that need to receive connections from the Internet share the same code to add a specific rule in the Windows Firewall authorizing incoming requests to go through. It will add the value "*[malware file name]:*:Enabled:ipsec*" to the following registry key "*HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/SharedAccess/Parameters/FirewallPolicy/StandardProfile/AuthorizedAppl* to achieve this goal. The following screenshot shows a subset of the "*add_to_firewall_exception()*" function.

```
if ( GetModuleFileNameA(0, module_filename, 0x200u) )
{
  if ( RegOpenKeyA(
         HKEY_LOCAL_MACHINE,
         "SYSTEM\\CurrentControlSet\\Services\\SharedAccess\\Parameters\\FirewallPolicy\\StandardProfile\\AuthorizedApplications\\List",
         &key_handle) )
  {
    result = -1;
  }
  else
  {
    wsprintfA(enabled_ipsec, "%s:*:Enabled:ipsec", module_filename);
    len = strlen(enabled_ipsec);
    if ( RegSetValueExA(key_handle, module_filename, 0, 1u, enabled_ipsec, len) )
    {
```

A subset of "add_to_firewall_exception()" function, shared by almost all components of Win32/Sality.

In Win32/RBrute.B, this function is called at the beginning of the malware execution:

```
  force_flag[0] = 0;
  memset(&force_flag[1], 0, 1020u);
  *(_WORD *)&force_flag[1021] = 0;
  force_flag[1023] = 0;
  add_to_firewall_exception();
  SetErrorMode(0x8002u);                        // SEM_N
  WSAStartup(0x202u, &WSAData);
  g_current_process = GetCurrentProcess();
  GetModuleFileNameA(0, module_filename, 0x400u);
  InitializeCriticalSection(&CriticalSection_1);
  InitializeCriticalSection(&CriticalSection_2);
  random = 0;
  for ( i = 0; i < 1000; ++i )
    random += (get_random() & 0xFFFFui64) % (i + 1000);
  lstrcpyA(mutex_name, "SKK29MXAD");
  Sleep(10000u);
  mutex = CreateMutexA(0, 1, mutex_name);
  if ( GetLastError() == 183 )
  {
    ReleaseMutex(mutex);
    CloseHandle(mutex);
    result = 0;
  }
```

Calling add_to_firewall_exception() before creating the mutex in the WinMain() function of Win32/RBrute.B.

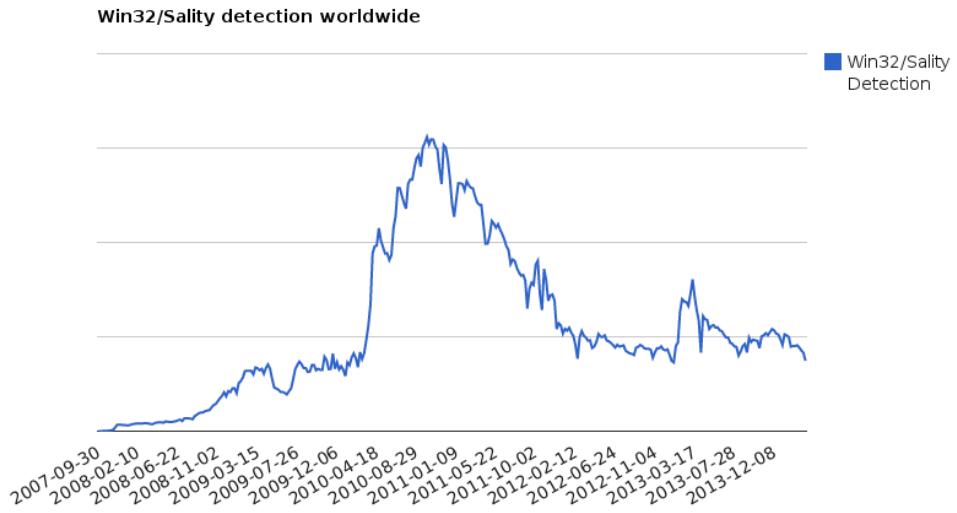
Same thing found in the Win32/Sality's spambot component:

```
  buffer[0] = 0;
  memset(&buffer[1], 0, 1020u);
  *(_WORD *)&buffer[1021] = 0;
  buffer[1023] = 0;
  add_to_firewall_exception();
  SetErrorMode(SEM_NOGPFAULTERRORBOX);
  mutex = CreateMutexA(0, 1, "qiwuyeiu2983");
  if ( GetLastError() == ERROR_ALREADY_EXISTS )
  {
    ReleaseMutex(mutex);
    CloseHandle(mutex);
    ExitProcess(0);
  }
```

Win32/Sality's spambot component calling the add_to_firewall_exception() function.

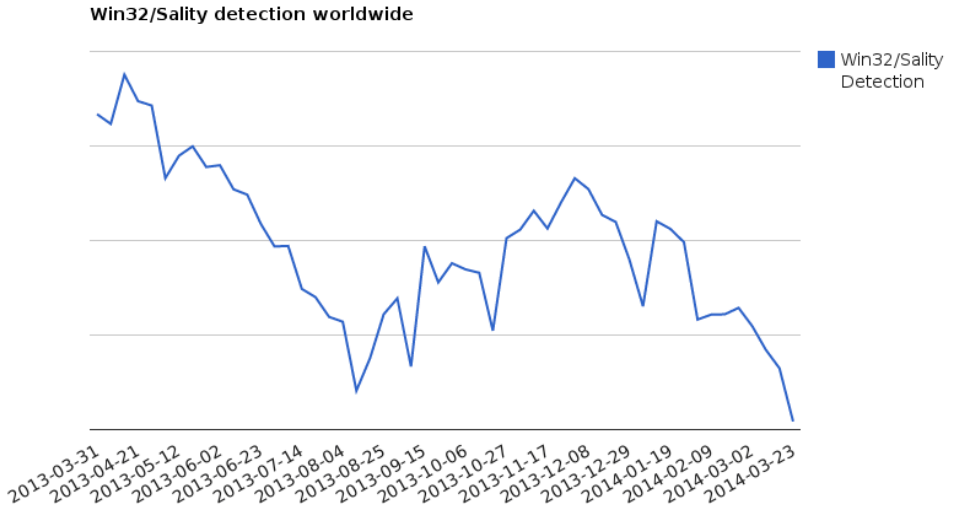
Infection Statistics

Our data show that the detection for Win32/Sality is currently decreasing or at least staying stable since 2012. We believe that the reduced number of detections is due to the reduced efficiency of the current infection vectors. This might explain why the operators are looking for new ways to spread Win32/Sality.



Win32/Sality detection worldwide

If we take a look at the detections for the last year, we can see a small increase, around December 2013, in Win32/Sality detections that coincides with the date where the DNS changer component was released in the wild, although those numbers should be taken with a grain of salt since other factors could contribute to variation in its spreading, like being dropped by another botnet.



Win32/Sality detections last year

We're not sure about the effectiveness of the Win32/Sality router DNS changer operation, since a lot of router configuration portals listen only on the private address space (e.g., 192.168.0.0/16) -- making them non-accessible from the Internet. Also, the router password brute force is not very aggressive, only trying a list of about ten passwords.

Conclusion

The usual infection vectors of Win32/Sality might not be sufficient enough to keep the botnet alive; hence the botnet controllers are deploying new component to grow the botnet. DNS hijacking on routers can be quite effective if done correctly. It can reach a lot of users behind a single router, especially on public access points. As routers are not commonly protected by security solutions, it provides an unrestricted environment to attackers allowing them to try several techniques to steal users' information. An existing technology that could fix the problem is DNSSEC, since the result of a DNS request is cryptographically signed and hence not prone to tampering. A good security practice that would reduce the scope of the problem is to change the default password on router's web interface.

File Analyzed

The following are the SHA-1 hashes of files observed during our monitoring of Win32/RBrute malware.

Win32/RBrute.A

8f4e43675948e806d99125e916191e04f8840b46
f8031d843626ac198a6f3c056f57098012e178e2
21649df3044f2203403a855108c1db1d95a2ab46

Win32/RBrute.B

5d1263c1c707ce163c9b36452dcb7340a7fd8909
73e2fe07a3f875521b5bfe8c3dd8fd6b6819c8f8
