symantec. Security Response

The Downadup Codex

A comprehensive guide to the threat's mechanics.

Edition 1.0

Contents

Introduction

Since its appearance in late-2008, the Downadup worm has become one of the most wide-spread threats to hit the Internet for a number of years. A complex piece of malicious code, this threat was able to jump certain network hurdles, hide in the shadows of network traffic, and defend itself against attack with a deftness not often seen in today's threat landscape. Yet it contained few previously unseen features. What set it apart was the sheer number of tricks it held up its sleeve.

It all started in mid-October of 2008, we began to receive reports of targeted attacks taking advantage of an as-yet unknown vulnerability in Window's remote procedure call (RPC) service. Microsoft quickly released an out-of-band security patch (MS08-067), going so far as to classify the update as "critical" for some operating systems—the highest designation for a Microsoft Security Bulletin.

It didn't take long for malware authors to utilize this vulnerability in their malicious code. In early November, W32.Kernelbot.A and W32.Wecorl appeared, demonstrating limited success in exploiting MS08-067. Still much of the talk at the time focused on the potential havoc that could be caused by this vulnerability, as opposed to damage caused by these threats.

It wasn't until late November that W32.Downadup appeared (also called Conficker by some news agencies and antivirus vendors). This threat achieved modest propagation success, partly due to its borrowing from the work of the Metasploit Project, which had been actively developing more reliable proof-of-concept methods to take advantage of the vulnerability.

The infection numbers for W32.Downadup began to steadily rise. But a limiting factor to its success was that its propagation routine depended on a publicly available GeoIP data file used to determine IP location. When the GeoIP authors decided to remove it from the location called by the worm, the absence of this file made it difficult for the worm to spread as rapidly, reducing its propagation to local networks already infected.

Not to be outmaneuvered, the Downadup authors packaged this GeoIP file within a new variant—W32.Downadup.B—along with a Swiss Army-like collection of secondary tricks in the hopes that this would help the threat spread far and wide.

This was one thing that set the Downadup worm apart from its counterparts of the last few years—its technical versatility. These secondary tricks weren't new; there were just so many of them. It scanned the network for vulnerable hosts, but didn't flood it with traffic, selectively querying various computers in an attempt at masking its traffic instead. It attempted to bruteforce commonly used network passwords. It took advantage of Universal Plug and Play to pass through routers and gateways. And when the network proved too secure, it used a rather clever AutoPlay trick to get users to execute it from removable drives.

The threat even protected itself from takeover. Transferred payload files were encrypted, as well as digitally signed, and only the Downadup authors had the key. A "hotpatching" routine for MS08-067 prevented further exploitation by other attackers or threats. The threat's authors went to great lengths to prevent buffer overflow exploitation of their own code. No one was going to hijack this worm's network of potential bots.

But the hidden danger behind all of this was the potential payload—Downadup contained the ability to update itself or receive additional files for execution. Again, not a new technique, but in this case the threat was generating a list of 250 new domains to connect to—every day. Any one of these domains could potentially contain an update that, if downloaded, would allow the threat to perform further malicious actions. What sort of actions? Anything the authors wanted really. Not only that, but the threat contained its own peer-to-peer (P2P) updating mechanism, allowing one infected computer to update another. Blocking access to the domains might protect you from one vector, but blocking a P2P update is a different matter.

Infection rates began to decline in mid-February, as news of the threat spread and network administrators that had not applied MS08-067 scrambled to protect their networks. This could be in part due to the fact that, with propagation success, the threat garnered a significant amount of attention. The domains meant to update the threat were being closely watched by security vendors, researchers, and the media.

This attention isn't surprising given the threat's complexity and the way it has utilized so many time-tested malicious code tricks, reinventing a few along the way. There is no doubt that Downadup has been the hot topic for 2009 thus far, and this attention has generated a considerable amount of research.

All was quiet on the Downadup front, that is until early March, when W32.Downadup.C began to appear on previously infected Downadup computers. More of an update than a new worm, this version didn't include a propagation technique. New to this version was a function to end a variety of security-related processes. And where previous versions generated a list of 250 daily domains, this one created 50,000.

To date, Symantec Security Response published 14 blog entries detailing the various features of Downadup between November 2008 and February 2009. This included a multi-part analysis of the features of the worm, where each entry in the series focused on a particular function. This paper brings these entries together into one source.

As information on our blog was release on a day-by-day basis, this paper is organized chronologically, not only covering the technical aspects, but also providing the historical context on the emergence and growth of the Downadup worm. And to provide a view of the whole picture, this paper also includes new, as-yet unpublished entries from our researchers.

As a comprehensive collection of our analysis of the worm, we present The Downadup Codex.

Editor's Note

This paper is a collection of all current Downadup research published through March 13, 2009. As with any constantly evolving threat, new variants could present themselves at any time and may render portions of this report out-of-date. Since Downadup appears to be one such threat, we will attempt to release updated editions of this report periodically, as new information presents itself.

Edition 1.0

Originally published March 13, 2009.

Increase in exploit attempts against MS08-067

Originally published November 22, 2008 by the Security Intel Analysis Team

Microsoft Security bulletin MS08-067 was an out-of-band security update that was released on October 23, 2008, to address a critical remotely exploitable vulnerability that was being exploited in the wild. The Microsoft Windows Server Service RPC Handling Remote Code Execution Vulnerability that was addressed by the patch affects Windows 2000, XP, Server 2003, Vista, and Server 2008 to varying degrees. Ultimately the issue can be exploited by a remote attacker to install malicious applications on a target computer without the victim's knowledge.

Microsoft released a detailed matrix describing the risk that this vulnerability presents to different versions of Microsoft Windows. When reading this matrix it becomes clear that this issue is exploitable by an unauthenticated attacker on Windows 2000, Windows XP, and Windows 2003. But, it is not exploitable on default configurations of Windows XP because the Windows Firewall blocks connect attempts to the required RPC interface. However, if the firewall is disabled, or the firewall is enabled but file/printer sharing is also enabled, then the issue is remotely exploitable on Windows XP. An attacker would need to authenticate to Windows Vista and Windows Server 2008 in order to exploit this issue.

Several public exploits are currently available that leverage this issue. Typically an exploit needs to be reliable for a worm to incorporate the exploit into its propagation routines. The nature of this vulnerability made it difficult for exploit authors to construct a single exploit that would successfully leverage the issue for all versions of Microsoft Windows at once. So, exploits were released that targeted specific versions of Microsoft Windows first, and the first public exploit to surface that wasn't a simple crash proof-of-concept leveraged the issue on Microsoft Windows platforms that were localized for traditional Chinese markets. Over the past month, exploit authors have discovered far more reliable methods to exploit this vulnerability and have released more stable exploits. The most reliable public exploit is incorporated into the Metasploit Framework—it contains many configurations that can be used to leverage this issue for a large array of Windows versions.

When we first noticed worm-like malicious applications exploiting this vulnerability they were using the primitive exploits that were available at the time. In other words, exploits that targeted Chinese Windows systems. However, over the last 24 hours we are observing a new worm. It exploits MS08-067, but it uses the routines from the Metasploit Framework to exploit the following platforms:

- Windows 2000 Universal
- Windows 2003 SP1 English
- Windows 2003 SP2 English
- Windows XP SP2 English
- Windows XP SP2 Arabic
- Windows XP SP2 Portuguese
- Windows XP SP2 Russian
- Windows XP SP2 Danish
- Windows XP SP2 Dutch
- Windows XP SP2 Finnish
- Windows XP SP2 French
- Windows XP SP2 Greek
- Windows XP SP2 Hungarian
- Windows XP SP2 Hebrew
- Windows XP SP2 Italian
- Windows XP SP2 Norwegian
- Windows XP SP2 Polish
- Windows XP SP2 Italian
- Windows XP SP2 Spanish
- Windows XP SP2 Swedish

The routine to attack Windows 2000 systems is very reliable; however, at the moment, the reliability of the routines that attack other platforms is not known.

The worm targets TCP port 445 to exploit the issue, and if it successfully exploits the issue, the worm then creates an HTTP server on the compromised computer on a random port, for example:

```
http://[EXTERNAL IP ADDRESS OF INFECTED MACHINE]:[RANDOM PORT]/[RANDOM STRING]
```

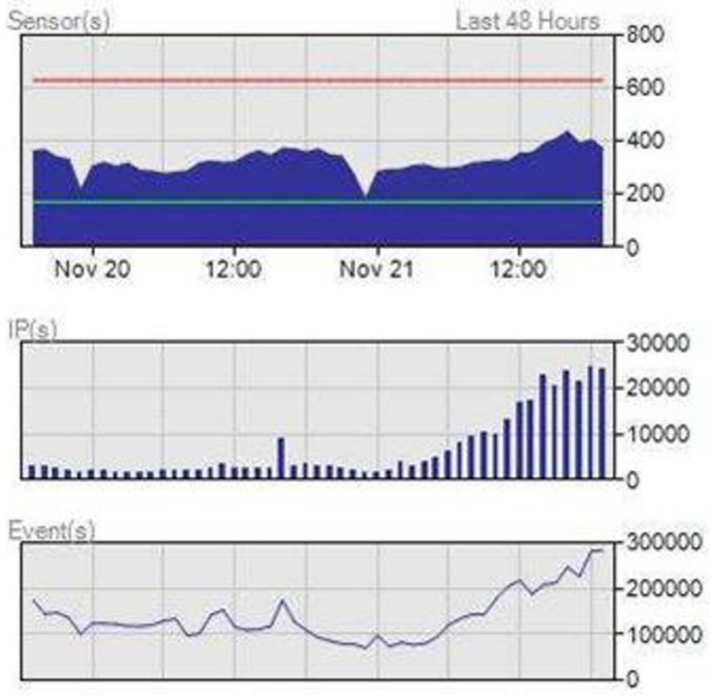
The worm then sends this URL as part of its payload to remote computers. Upon successful exploitation, the remote computer will then connect back to this URL and download the worm.

We are currently observing an increase in IPs generating activity over TCP port 445 and we believe that this activity is at least in part related to the propagation of this malicious code, as shown in Figure 1.

SANs are also reporting a spike in activity on TCP port 445. However, this was not the main reason behind our ThreatCon update. The aggressive propagation of this malicious threat in our honeypot network was the main reason behind the update. We decided that the activity was significant enough to remind our customers of the importance of installing the MS08-067 updates. Symantec antivirus currently detects this threat as W32.Downadup, so please make sure that your antivirus software is up to date.

Figure 1

IPs generating activity on TCP port 445



We also recommend that the following mitigating strategies are applied:

- Block access to TCP port 139 and 445 at network perimeters.
- Ensure that computers that are connected to the network have host-based firewall software installed.
- Ensure that antivirus software is installed on all clients connected to the network and that the software is up to date.

And, please install the update from MS08-067 as soon as possible. Microsoft has suggested a number of additional workarounds in the security bulletin, such as disabling the browser service. We advise customers to review their suggestions as well.

Symantec IPS will detect and block this attack with the following signatures:

- MSRPC Server Service Buffer Overflow
- RPC Server Service BO2

W32.Downadup infection statistics

Originally published January 6, 2009 by the Security Intel Analysis Team

The W32.Downadup worm was the first worm discovered in the wild that was successfully leveraging MS08-067 in a widespread fashion. Symantec carried out an in-depth analysis of this threat and discovered that infected hosts will generate 250 pseudo-random domain addresses each day, in preparation of attempting to contact them later on to download and execute an update binary.

This is an interesting and increasingly popular technique that malicious code authors have been deploying. It allows them to more easily evade domain and server takedowns, because until they choose to register a domain associated with a given day, the security industry is unable to know for sure which domain will be used and therefore has little to target. Fortunately, by reverse engineering the domain-generation algorithms, we are able to proactively identify and blacklist the domains.

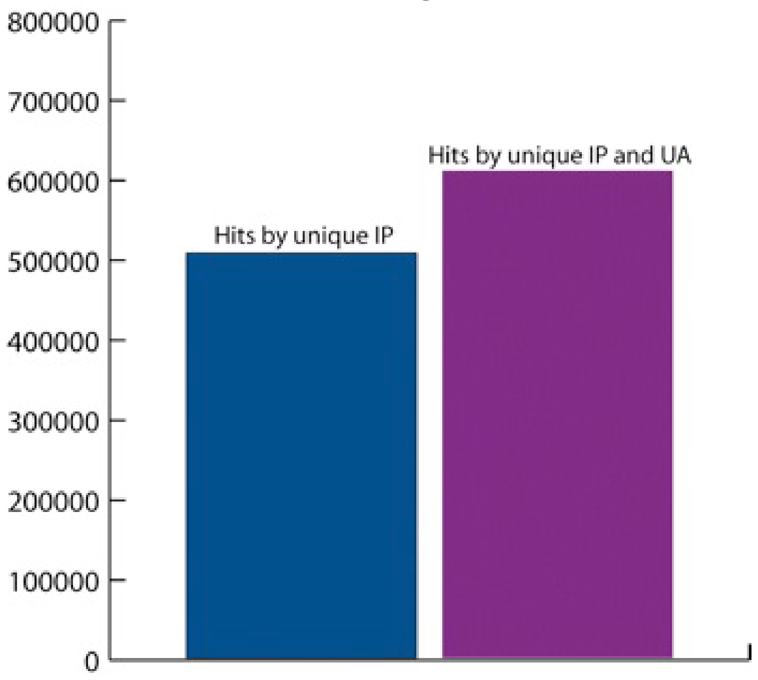
What's also interesting about this method of obtaining binary updates is that it does allow for the number of infections to be approximated by monitoring contact attempts against generated domains. By pre-calculating and registering future domains, the Symantec Intelligence Analysis Team was able to observe contact attempts made by numerous infections. Over the course of a week, we observed over three million unique IP addresses attempting to obtain a download file from our server. However, we believe that the number of infections is higher than this estimate due to multiple internal infections that may be using network address translation (NAT) behind a single external IP address. Also, it's possible that an infected computer does not contact all 250 generated domains each day. If this latter possibility is the case, then we may only be seeing a subset of the actual total number of infected computers in this bot network.

For instance, we have been able to show that multiple infections are coming from a single IP address by identifying unique user-agent strings coming from the same IP. The following graph shows the statistics, over a 72-hour period, of unique IP addresses versus unique IP address and user-agent pairs, as shown in Figure 2.

While on the topic of user-agents, when contacting one of the generated domains to obtain a binary, an infected computer sends a specific user-agent string as part of the HTTP request. User-agent strings contain version information about the associated operating system (OS) and Web browser, and can be used to collect interesting statistics. For example, Windows XP SP1 can be identified by a user-agent containing Windows NT 5.1. Systems running Windows XP SP2 and later can be identified by Windows NT 5.1; SV1. By analyzing the user-agent strings associated with each unique request, we are able to approximate the distribution of infected operating system types. The following graphic shows the OS distribution observed over a 72-hour period, as shown in Figure 3.

Figure 2

**Unique IP addresses vs.
Unique IP address and user-agent pairs**



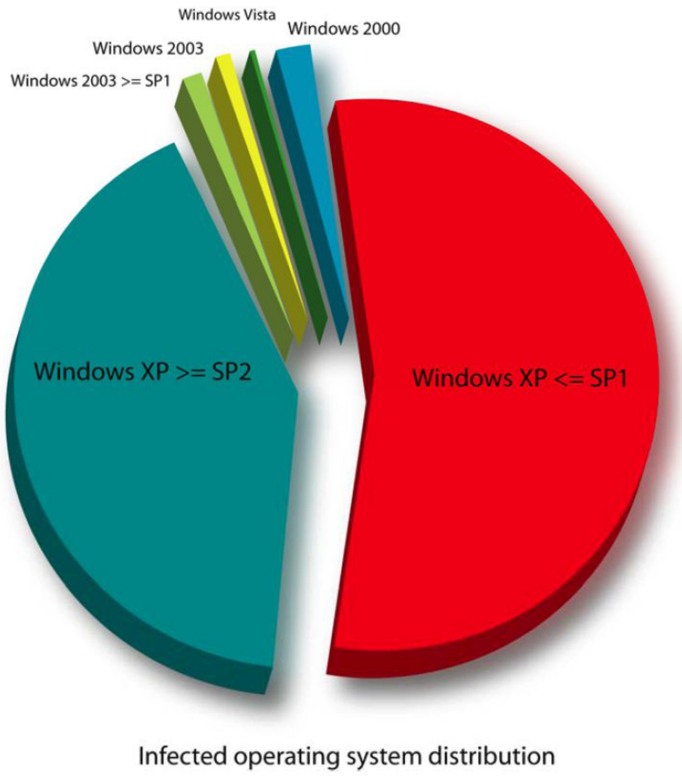
As can be seen, the most commonly infected systems appear to be Windows XP SP1 and earlier. Over 500,000 of the infected computers that contacted our server were running these operating system versions. Close behind was Windows XP SP2 and later systems. Windows 2000 and Windows 2003 had smaller shares.

We believe that the W32.Downadup propagation routine has been very aggressive. It will continue to infect computers in the near future and receive updates via the aforementioned mechanism. Symantec discovered a new variant of this worm on December 30, 2008, dubbed W32.Downadup.B. This updated version contains additional propagation routines and what appears to be an altered domain generation routine. It's not currently known if this new version was seeded to W32. Downadup infections or has independently spread through its own propagation routines.

We strongly encourage all users to ensure that the patches available in MS08-067 have been applied and that antivirus products are fully up-to-date to ensure that this threat does not find its way onto computers.

Figure 3

OS distribution over a 72-hour period



New variants of W32.Downadup.B find new ways to propagate

Originally published January 9, 2009 by Symantec Security Response

Symantec has observed an increase in infections relating to W32.Downadup over the holiday period and is urging organizations to apply the patch for Microsoft Windows Server Service RPC Handling Remote Code Execution Vulnerability as soon as possible.

A new variant of this threat, called W32.Downadup.B, appeared on December 30th and can not only propagate by exploiting the Microsoft Windows Server Service RPC Handling Remote Code Execution Vulnerability, but can also spread through corporate networks by infecting USB sticks and accessing weak passwords. These propagation methods are nothing new; W32.Spybot, W32.Randex, and W32.Mytob variants all use almost identical methods to spread, but this variant requires more effort to protect corporate networks.

W32.Downadup.B creates an autorun.inf file on all mapped drives so that the threat automatically executes when the drive is accessed. The threat then monitors for drives that are connected to the compromised computer in order to create an autorun.inf file as soon as the drive becomes accessible. The worm also monitors DNS requests to domains containing certain strings and blocks access to those domains so that it will appear that the network request timed out. This means infected users may not be able to update their security software from those Web sites. This can be problematic as worm authors generally dish out new variants constantly.

Symantec researchers are seeing considerable detections of both variants of W32.Downadup and W32.Downadup.B. As illustrated by the following infection maps based on data from the past 60 days, the infections are geographically quite widespread. The highest infection rates typically correspond to countries with high rates of computer/Internet usage.

Figure 4

Infections of W32.Downadup

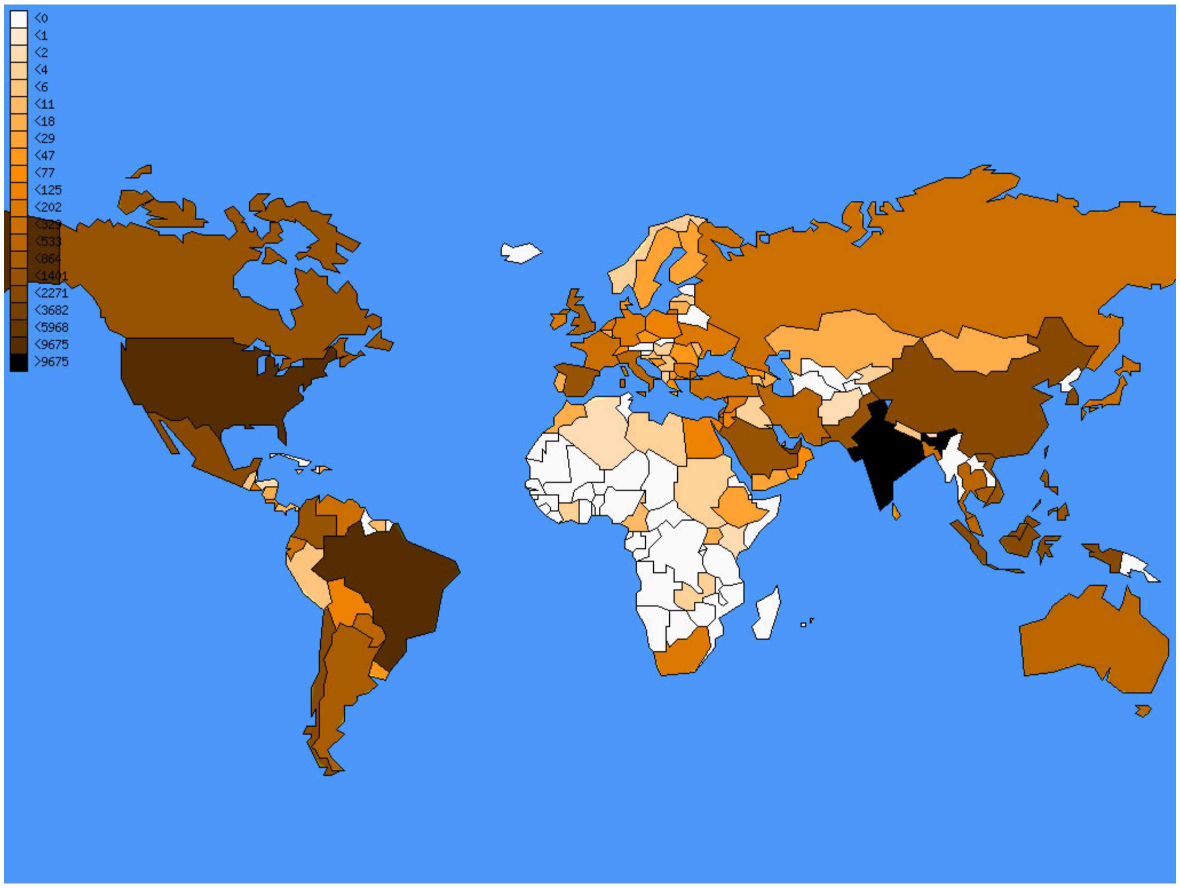
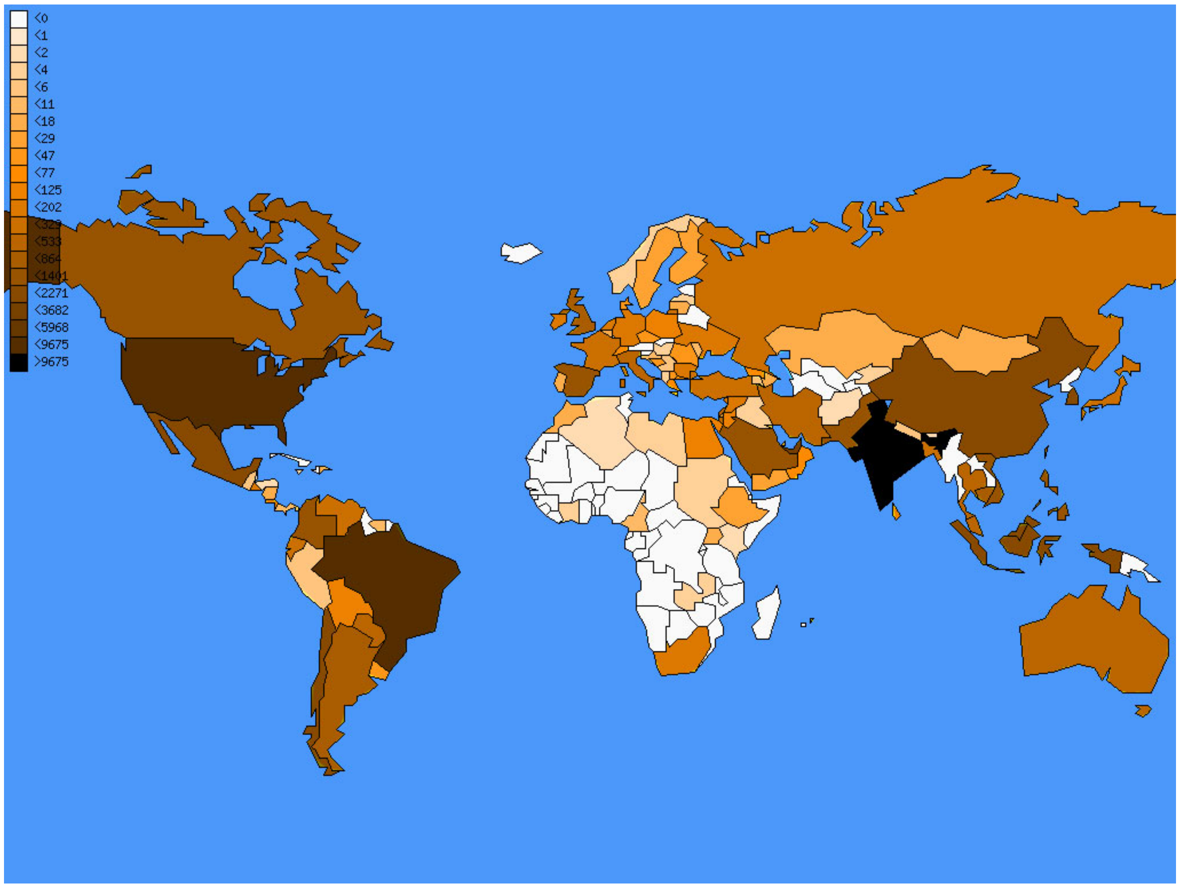


Figure 5

Infections of W32.Downadup.B



Symantec strongly encourages users to patch their system against the Microsoft Windows Server Service RPC Handling Remote Code Execution Vulnerability, take steps to control the execution of applications referenced in the autorun.inf files that may be located on removable and network drives, and enforce a strong password policy on all computers within their networks. Particularly during holiday periods patch updates can be missed and is an opportune time for malware to spread. Consider implementing an automated patch management solution to help mitigate risk.

Click here to obtain more information about how to prevent a threat from spreading using the AutoPlay feature.

W32.Downadup and W32.Downadup.B statistics

Originally published January 16, 2009 by the Security Intel Analysis Team

As regular readers of the Symantec Security Response Blog know, we've been monitoring W32.Downadup statistics for some time. We've previously published two blog entries regarding infection statistics for both the .A and .B variants. The Symantec Intelligence Analysis Team has been monitoring infections since mid-December. We recommend that readers familiarize themselves with the information in the previous blogs, as well as the Symantec Security Response writeups for the worm, before reading the rest of this article.

W32.Downadup is an extremely interesting piece of malicious code and one of the most prolific worms we've seen in years. This is largely attributed to the fact that it is capable of trivially exploiting users who are running unpatched Windows XP SP2 and Windows 2003 SP1 systems. Other worms released over the past few years have largely targeted older system versions, which have an ever decreasing distribution.

The Symantec Intelligence Analysis Team has recently begun monitoring W32.Downadup.B infections using the same method used to monitor W32.Downadup. Basically, both worms use custom date-based algorithms to generate 250 domain names per day. These domains are then contacted by each infection in an attempt to obtain an update binary. By reverse engineering the algorithms and generating tools to mimic the domain generation routine, we are able to predict domains that will be contacted by infected systems on future dates. We take advantage of this knowledge by preemptively registering domains that will be queried in the future and on the associated day, logging all of the results.

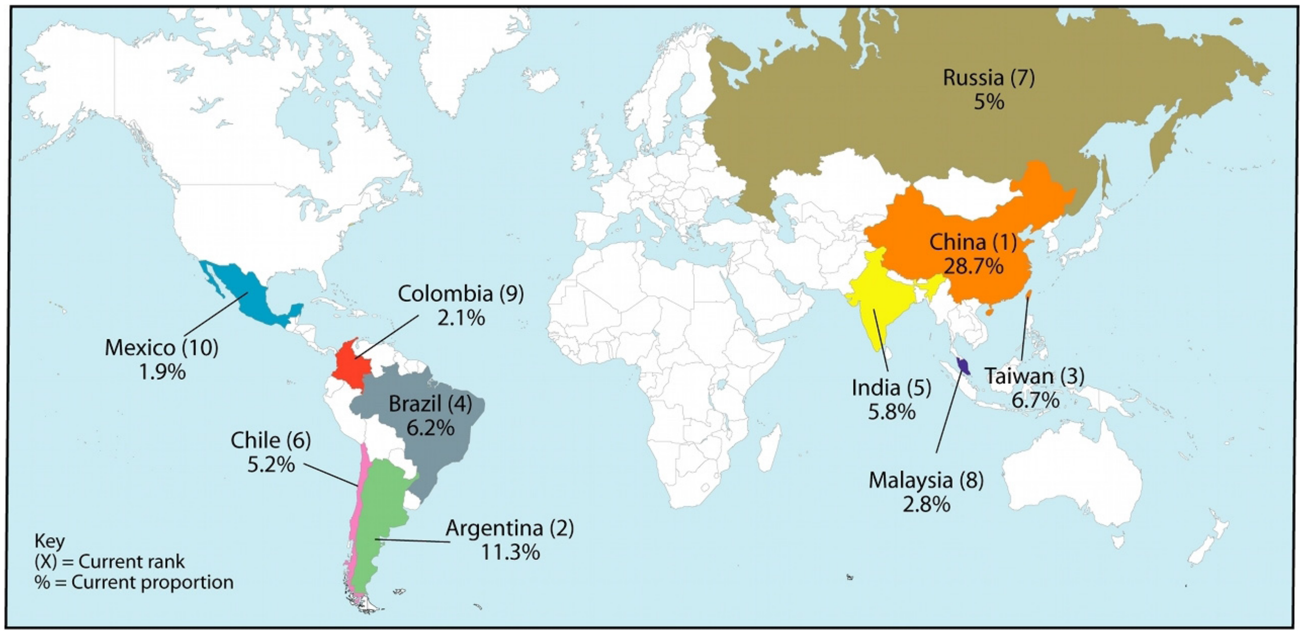
The logs can then be used to tell us a number of interesting things. The string below is an example of what one of the log entries looks like:

```
x.x.x.x [16/Jan/2009:09:45:09 -0700] "GET /search?q=0 HTTP/1.0" 404 282 "-"
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)"
```

First, we get the connecting IP. This could be an externally facing infected system or an Internet gateway used by multiple systems using network address translation (NAT). Access to the IP addresses allows us to roughly approximate (although conservatively) the number of infections and also geographically map out infection density. For instance, the following image illustrates the top 10 countries by W32.Downadup infection count.

Figure 6

Top ten W32.Downadup countries



The IP data shows us that China and Argentina are by far the most infected areas. Both East Asia and South America are the main areas of infection. In total we've observed over three million unique IP addresses infected with W32.Downadup.

The logs also tell us other valuable information. In our previous blog entry, we showed the operating system distribution of infected systems. These can be obtained using the User-Agent string sent to the server when querying for the update. The User-Agents also allow us to more accurately approximate the total number of infections. By creating unique pairs of IP addresses and User-Agent strings, we can identify additional systems that are using NAT behind a gateway. This still does not give us the total picture; however, it does reveal a significant number of additional infections. For instance, as mentioned previously, we observed three million unique IPs, but we also observed 3.7 million unique IP / User-Agent pairs.

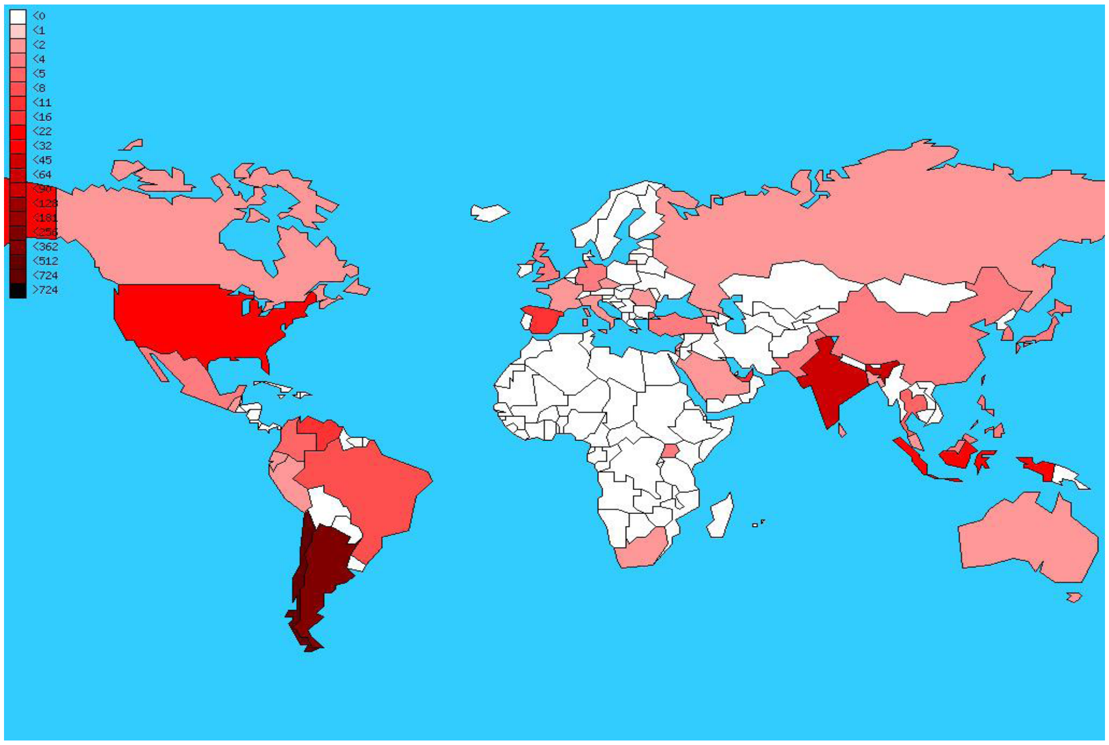
There is one more piece of interesting information that can be obtained from the logs. That's the infection count reported by the infected system, using the q= parameter. The value used to populate this field is updated by the infected and attacking system every time an incoming connection is made to the infected system over HTTP for downloading a copy of the W32.Downadup binary. Connections are triggered through successful exploitation of the MS08-067 vulnerability.

The q= value is most interesting to us for the purposes of identifying aggressive and potentially longer term infections. The former allows us to identify possibly large and vulnerable internal networks. The latter allows us to speculate (although tentatively) on possible infection starting points. Originally we believed that this field may be valuable for calculating total infection counts, however we believe that this is inaccurate and our reasoning for this be will explained shortly.

Of the approximately three million W32.Downadup infections observed, we chose to map out those that appeared to be very successful in their exploitation efforts. We isolated all infected systems that reported a total number of successful attacks greater than 10,000. This allowed us to isolate just over 1,100 infected systems. Most ranged between 10,000 and 50,000 reported attacks. Two systems, located in Argentina, reported over 100,000 successful attacks. The following map shows geographic density of the infections reporting these attacks:

Figure 7

Infected systems with greater than 10,000 successful attacks



The map shows that Chile and Argentina have the highest density of infections, reporting high quantities of attacks. This corresponds to the previously shown image of top infection areas, with Argentina in second place and Chile in sixth. Although we can't say conclusively, the higher infection counts of these systems may be indicative of the longevity of infection and could therefore be an indicator as to the starting point of infection. There is no real way to know for sure at the moment, but it is an interesting possibility nonetheless.

As noted earlier, we feel that using the q= parameter to approximate a total infection count is inaccurate. It does indicate the total number of successful exploitation attempts and subsequent malware uploads; however, this may in fact not be an indicator of successful infection. Consider, for instance, a system that has antivirus software running with up-to-date definitions, but is not patched against the MS08-067 vulnerability. An infected system could successfully exploit this host and upload a copy of the worm; however, due to the antivirus technology on the system this would not represent an infection. Now, given the system was not infected and therefore did not have the vulnerable code hot-patched (a feature of the worm), it is still open to attack by other infected systems. As such, a single vulnerable system with up-to-date antivirus could result in many infections misreporting their infection count. Then, realistically assume that numerous systems on the net are vulnerable to MS08-067 but may be impervious to infection and you start to see a large amount of skewing.

As further consideration, take the previous example of aggressive infections. We observed approximately 1,100 systems contacting our server, which reported over 10,000 infections (in fact many were 200-500% more than this). If these numbers were to be believed, these 1,100 systems would be responsible for 11 million infections. If we were to accumulate the observed q= values for all 3.7 million uniquely identifiable W32.Downadup infections we've observed, we feel that this value would be extremely large and would likely be very inaccurate.

The last major point of interest related to the statistics we've collected are the W32.Downadup.B numbers. We recently began monitoring these infections and used a list of 600,000 unique IP addresses known to be infected with this variant. We compared these IP addresses with the 3 million unique W32.Downadup IPs and came up with a surprising result. Only approximately 68,000 IPs are duplicates. First of all, this tells us that W32.Downadup infected systems were not used to seed the W32.Downadup.B binary. It would seem that the pseudo-random domain-based update mechanism has not yet been utilized. Second, it tells us that, given the large number of systems already infected with the .B variant, it is likely seeing large success over the new propagation vector that involves brute-forcing file shares. Because W32.Downadup will patch the system against further exploitation of MS08-067, it makes sense that there would not be as much cross-over of IP addresses.

Some of the cross-over can be written off as systems that were infected with W32.Downadup having been disinfected. Others may be that a system infected with W32.Downadup, which is no longer vulnerable to the MS08-067 vector, may still be vulnerable to file share brute-forcing. Additionally, some of the cross-over is likely coming from systems being infected through USB key transfer (another propagation vector leveraged by the .B variant) that are behind a shared gateway, and thus the same IP address, as systems infected with the original variant.

We're continuing to collect and analyze data related to infections, documenting how this worm family evolves, and watching to see if the domain-base update mechanism used. This is by far one of the most prolific worms in many years and has an extremely large infection base that could do a lot of damage.

Peer-to-peer payload distribution

Originally published January 19, 2009 by Eric Chien

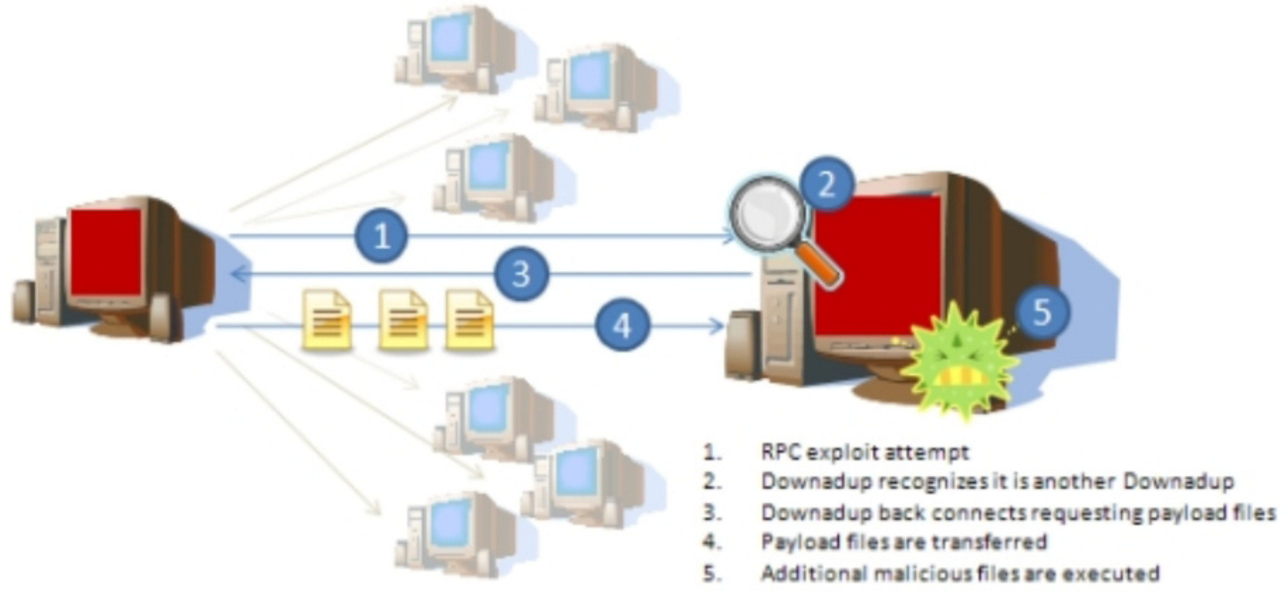
While many researchers, including us, are speculating on the magnitude of infections from Downadup (a.k.a. Conficker), we are also all waiting for the other shoe to drop. At this point, Downadup has replicated to potentially millions of machines but there has been no additional payload—yet. Ten years ago, just replicating was enough motivation in creating malicious code, while today the vast majority of malware has a monetary motivation. Based on previous variants and characteristics of the code, we believe the worm is associated with a well-known malware gang that has previously distributed a variety of adware, and more recently misleading applications (a.k.a. rogue antispyware products).

The worm actually has two mechanisms to receive additional payload files. One has been reported on widely, in which a list of domain names is generated every day and contacted for updates. Eventually, one of these domains is likely to be registered by the malicious code authors where they will host the additional payload file. However, security vendors such as Symantec are also watching these domains, which make them less than ideal for the malware authors, especially if they are quickly shut down.

So, another mechanism exists to distribute the payload files and it is more difficult to track and equally more difficult to shut down. The worm uses a (potentially inefficient) peer-to-peer (P2P) mechanism that allows it to share files between infections.

Figure 8

Downadup peer-to-peer mechanism


1
2
3
4
5
1. RPC exploit attempt
2. Downadup recognizes it is another Downadup
3. Downadup back connects requesting payload files
4. Payload files are transferred
5. Additional malicious files are executed

During the process shown above, Downadup not only patches the RPC vulnerability in memory, but uses this patch to recognize incoming exploit attempts from other Downadup-infected machines. The worm is able to analyze the incoming shellcode and checks if it matches its own exploit shellcode. If the shellcode matches, information is extracted from the shellcode that allows the worm to connect back to the other infected machine. This "back connect" uses the HTTP protocol, but on a randomly selected port. The other infected machine then responds with a packet of data consisting of the payload files.

Downadup can transfer multiple payload files using this mechanism. Each is possibly encrypted (or at least digitally signed) and contains a header containing a file identifier and a date timestamp. The file identifier allows the worm to check if it already knows about this file and determine if it needs to be updated. The date timestamp is used as an expiration date and if the file is past its expiration date, it is discarded. The payload files are continually reviewed and those that are past their expiration are culled. These payload files are then saved in the registry and provided when other peers request them and allows the payload files to be maintained across reboots.

These payload files can then either be saved to disk and executed or loaded directly to memory. Thus, additional payload files can end up being executed with no files hitting the disk.

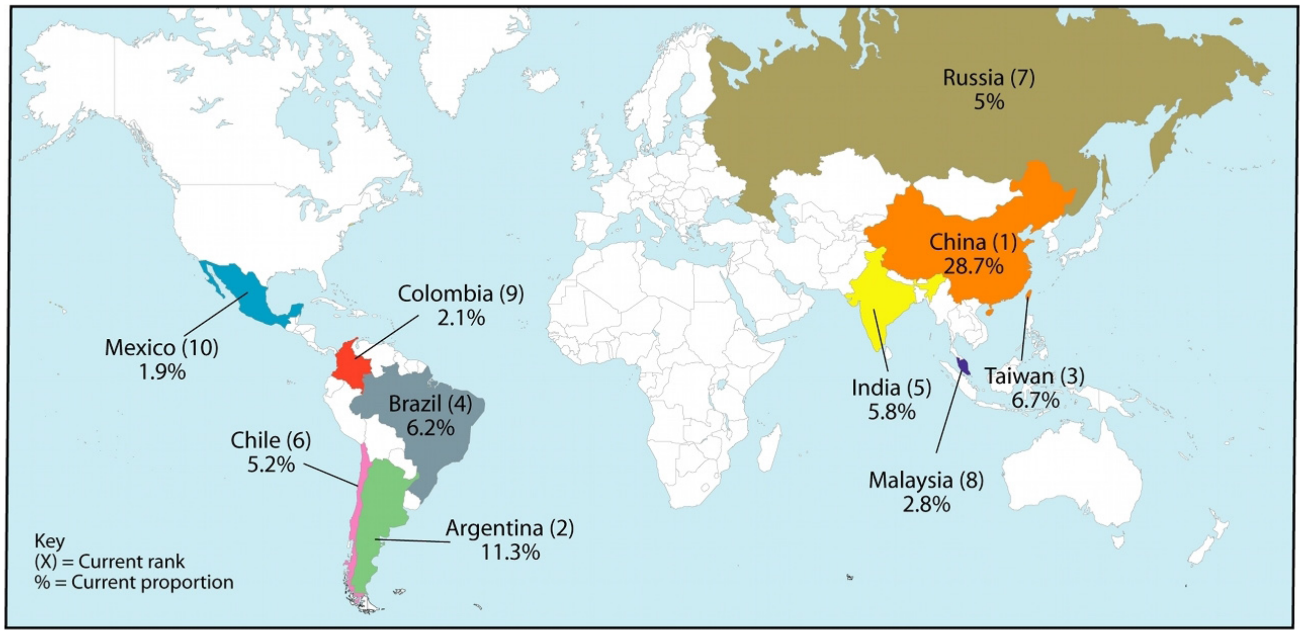
So, while we know Downadup’s method of operation, we still await its motive.

Geo-location, fingerprinting, and piracy

Originally published January 20, 2009 by Patrick Fitzgerald

Downadup has been the most prolific worm that we have seen for some time. While as a part of this series we are documenting some of the more interesting, technical sides of this threat, other non-technical aspects of this threat also present noteworthy issues. The map below shows the top 10 countries rated by infection prevalence.

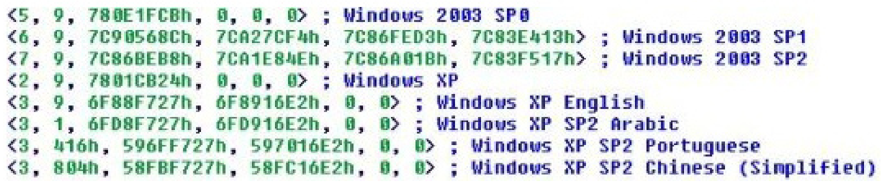
Figure 9
Top 10 countries ranked by W32.Downadup infections



As shown in figure 9, China has almost three times as many infections as the second-most infected country, Argentina. Even more notably, some of the nations with higher computer usage such as the United States or Korea are trailing considerably. So, what is causing this skew?

Downadup uses an remote procedure call (RPC) exploit as its main vector for propagation. This exploit is only effective against machines that have not applied the MS08-067 patch. In addition, the ability to exploit the vulnerability effectively requires knowledge of both the operating system (OS) version (e.g. Windows XP vs. Windows 2003) and the language of the targeted machine. The below figure is a snippet of a larger list in Downadup of configuration values needed for different versions and languages.

Figure 10
Downdup configuration values: snapshot



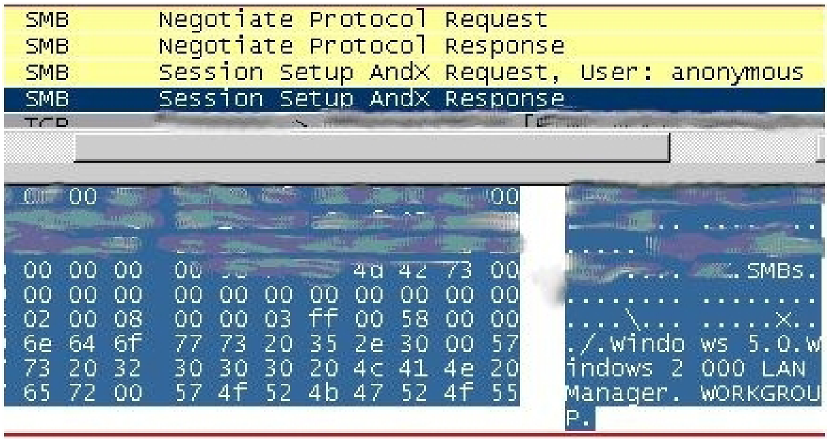
```
<5, 9, 780E1FCBh, 0, 0, 0> ; Windows 2003 SP0
<6, 9, 7C90568Ch, 7CA27CF4h, 7C86FED3h, 7C83E413h> ; Windows 2003 SP1
<7, 9, 7C86BEB8h, 7CA1E84Eh, 7C86A018h, 7C83F517h> ; Windows 2003 SP2
<2, 9, 7801CB24h, 0, 0, 0> ; Windows XP
<3, 9, 6F88F727h, 6F8916E2h, 0, 0> ; Windows XP English
<3, 1, 6FD8F727h, 6FD916E2h, 0, 0> ; Windows XP SP2 Arabic
<3, 416h, 596FF727h, 597016E2h, 0, 0> ; Windows XP SP2 Portuguese
<3, 804h, 58FBF727h, 58FC16E2h, 0, 0> ; Windows XP SP2 Chinese (Simplified)
```

First, to attempt to determine which version of Windows the remote host is running, Downadup fingerprints the remote host by sending an SMB Session Setup Request. The remote machine provides the OS version and service pack as part of its response. For example, the figure below shows a remote host responding with Windows 2000 as its OS version.

The second, and more difficult step, is determining the remote machine's language version. Downadup guesses at the language version by using IP geo-location. Recent versions of Downadup contain RC4 encrypted IP geo-location information. By looking up the remote machine's IP address in the geo-location information, Downadup is able to match the IP address to a country and then maps that country to a particular language. Downadup's geo-location data appears more effective for certain countries such as China and Argentina.

Figure 11

W32.Downadup remote host response



```
SMB      Negotiate Protocol Request
SMB      Negotiate Protocol Response
SMB      Session Setup AndX Request, User: anonymous
SMB      Session Setup AndX Response
```

```
[redacted]
00 00 00  00 [redacted] 4d 42 73 00   [redacted] ..SMBs.
00 00 00  00 00 00 00 00 00 00 00   ........ ........
02 00 08  00 00 03 ff 00 58 00 00   ....\... .....X..
6e 64 6f  77 73 20 35 2e 30 00 57   ./.Windo ws 5.0.W
73 20 32  30 30 30 20 4c 41 4e 20   indows 2 000 LAN
65 72 00  57 4f 52 4b 47 52 4f 55   Manager. WORKGROU
                                    P.
```

It should be noted that these techniques are far from 100% reliable and in some cases cannot even be utilized (for instance, private IP addresses in a NAT setup). In these cases, Downadup guesses at the version of the remote host and uses a set of mostly ineffective defaults for the language, except for Chinese and Brazilian Portuguese. If the existing host is Chinese or Brazilian Portuguese, Downadup will assume the remote host is Chinese or Brazilian Portuguese. This likely provides increased efficacy in countries such as China and Brazil.

A second possible explanation for the skew is that on October 20, 2008, Microsoft rolled out an updated Windows Genuine Advantage (WGA) system to help combat the high rate of piracy of its Windows platform. One of the side effects of this policy is that people using illegal copies of Windows will be more likely to disable automatic updates from Microsoft. The fear is that a subsequent update may adversely affect their experience with Windows in a similar way the "black screen" that affected many users in China operating illegal copies of Windows. Without automatic updates, it is highly unlikely that many of these users are manually installing critical updates such as MS08-067.

Figure 12 shows the rate of software piracy in 2006.

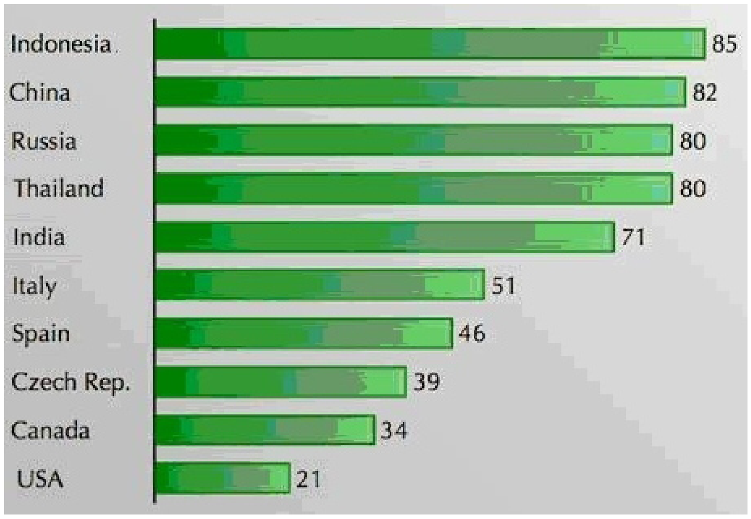
What is interesting about the data shown in figure 12 is that China, India, and Russia all have a high percentage of pirated software in use and these countries also feature in the top 10 countries ranked by W32.Downadup infections, as shown in figure 9 above. The lack of patching due to piracy may be a contributory factor to high infection rates in those countries. People with illegal copies of Windows who choose to disable automatic updates can create an ideal breeding ground for malicious code authors to proliferate their wares.

So, the bottom line is that while Downadup has been highlighted in the press due to extremely high infection numbers, not all countries are affected equally. Some residents of certain countries may be wondering what all the hype is about, while others can't understand why they haven't heard more about it.

Figure 12

Software piracy in selected countries, by percentage (2006)

Source: http://arstechnica.com/news.ars/post/20080122-bsa-piracy-economic-impact-is-tens-of-billions-of-dollars.html



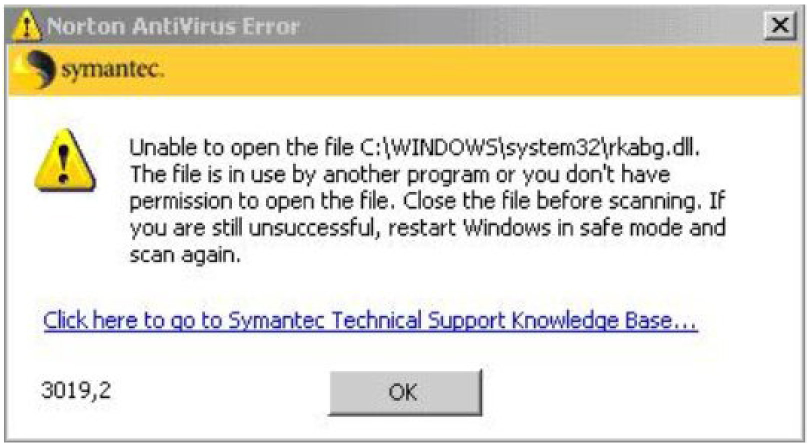
A lock with no key

Originally published January 21, 2009 by Ka Chun Leung

We know that W32.Downadup.B is aggressive when it comes to infecting computers. So, let's talk about some of the tricks it uses to stay on a computer once an infection is successful. One of our test computers was infected with W32.Downadup.B. I scanned it with an old, original shipping version of Norton Antivirus 2006 and the following error message appeared:

Figure 13

Error message from old version of Norton AntiVirus 2006



Norton AntiVirus Error

Unable to open the file C:\WINDOWS\system32\rkabg.dll. The file is in use by another program or you don't have permission to open the file. Close the file before scanning. If you are still unsuccessful, restart Windows in safe mode and scan again.

Click here to go to Symantec Technical Support Knowledge Base...

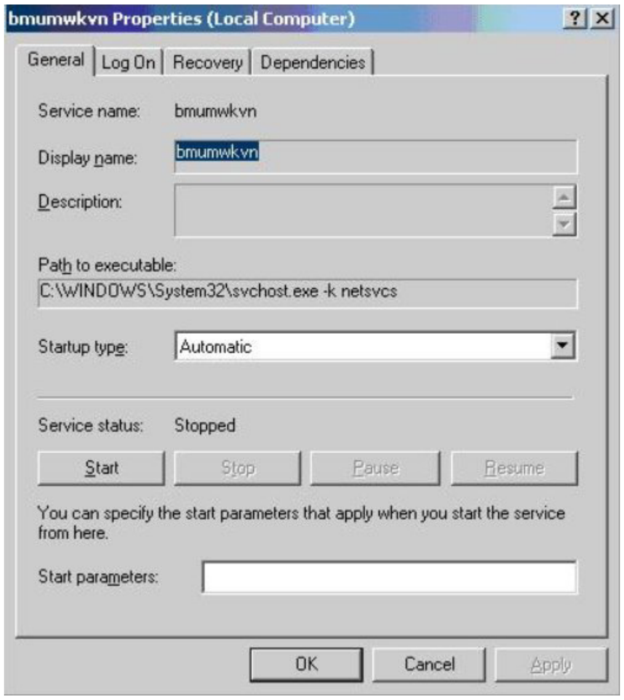
3019,2

A process on the computer has "locked" the file, which prevents anybody else from accessing it. Now, antivirus software has many ways of getting around this sort of lock. Why isn't it working here?

When W32.Downadup.B infected this computer, we saw it drop the file **C:\WINDOWS\system32\rkabg.dll** and then watched it install itself as a service (in this case as "bmumwkvn") in the netsvcs service group. So, I should be able to unlock the file by stopping this service.

Figure 14

W32.Downadup.B service



bmumwkvn Properties (Local Computer)

Service name: bmumwkvn

Display name: bmumwkvn

Description:

Path to executable:
C:\WINDOWS\System32\svchost.exe -k netsvcs

Startup type: Automatic

Service status: Stopped

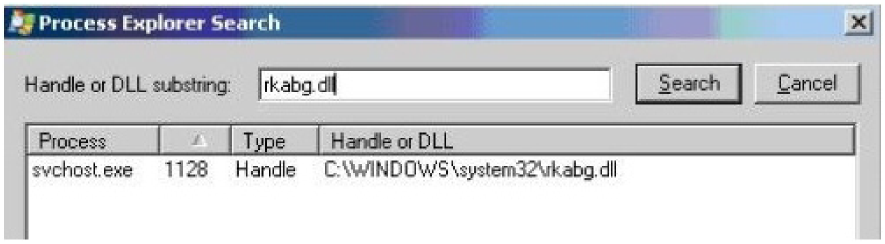
You can specify the start parameters that apply when you start the service from here.

Start parameters:

However, the service was already stopped. A closer look at the running services and processes on my computer reveals nothing suspicious, but something must be maintaining that lock. Process Explorer from Sysinternals (now a part of Microsoft) has a useful feature to find which process is accessing a file.

Figure 15

Process Explorer view of service



Handle or DLL substring: rkabg.dll

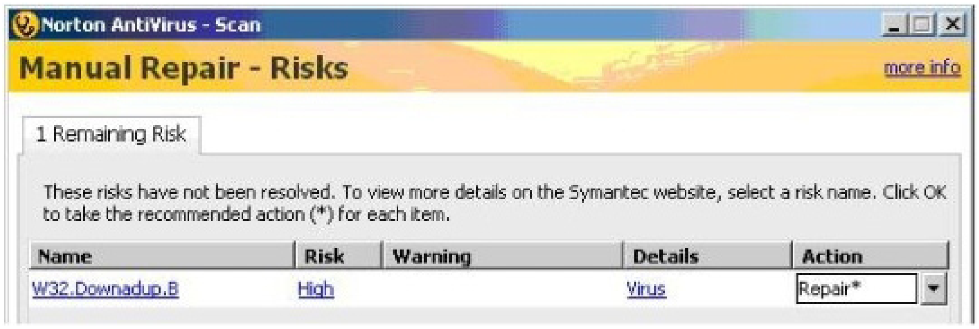
Process		Type	Handle or DLL
svchost.exe	1128	Handle	C:\WINDOWS\system32\rkabg.dll

 Svchost.exe—the process that hosts the service created by W32.Downadup.B—is holding the lock and running, even though the W32.Downadup.B specific service isn't stopped. Some further analysis reveals that the W32. Downadup.B service runs when the computer boots. It then injects code into the service host before unloading itself. This trick works against a surprising array of different antivirus software.

After closing the file handle in Process Explorer, the scan works as expected.

Figure 16

Working scan after closing file handle



These risks have not been resolved. To view more details on the Symantec website, select a risk name. Click OK to take the recommended action (*) for each item.

Name	Risk	Warning	Details	Action
W32.Downadup.B	High		Virus	Repair*

So, while this may have been an issue in 2006, up-to-date versions of Symantec's products no longer have this issue. Unfortunately for this and other threats, those users with unpatched older products are alerted anyway (even without the signature detecting the threat) due to the fact the file is locked.

Don't forget—if you believe your computer may be infected with Downadup and are having trouble detecting and removing it, you can always download our free fixtool.

Small improvements yield big returns

Originally published January 22, 2009 by Elia Florio

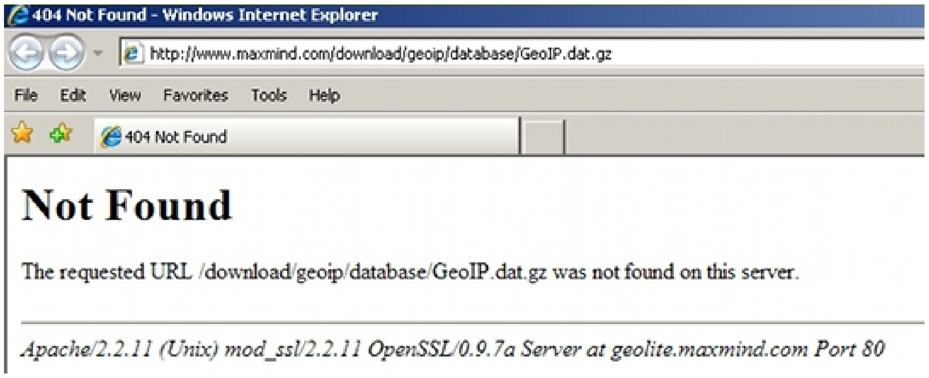
Back in November 2008, Symantec raised the ThreatCon level in response to a significant increase of exploitation activity of MS08-067, even when other vendors were still downplaying or ignoring this large increase of network attacks. This was just the beginning of Downadup saga.

W32.Downadup wasn't the first worm exploiting MS08-067, but it clearly had something "special" when compared to its previous competitor threats (see W32.Kernelbot.A and W32.Wecorl). From the programming style, the tricks, and the ideas used in the W32.Downadup code, we could easily say that W32.Downadup wasn't the average threat that we would normally see in the wild. The first variant of the worm was able to infect an estimated 500,000 machines due to an aggressive infection routine and a sophisticated exploitation algorithm, which makes use of geo-location and OS fingerprinting.

However, the first variant of Downadup wasn't able to achieve the same damages of its successor, mostly due to two reasons: it was able to spread only with one mechanism and it had a single point of failure, which was the data file used to perform geo-location of IP addresses. This file (GeoIP.dat.gz) was freely available on the Web back in November 2008 and was downloaded by the worm directly from the Web site using a hard-coded URL in its code. At some stage, MaxMind removed this GeoIP file from their Web site, probably because of the denial-of-service (DoS) effect they would have been experiencing on their servers, which were being contacted by all machines infected by W32.Downadup. This change somehow affected the exploitation abilities of the first variant.

Figure 17

MaxMind removes GeoIP file


404 Not Found - Windows Internet Explorer
http://www.maxmind.com/download/geoip/database/GeoIP.dat.gz
File
Edit
View
Favorites
Tools
Help
404 Not Found

Not Found

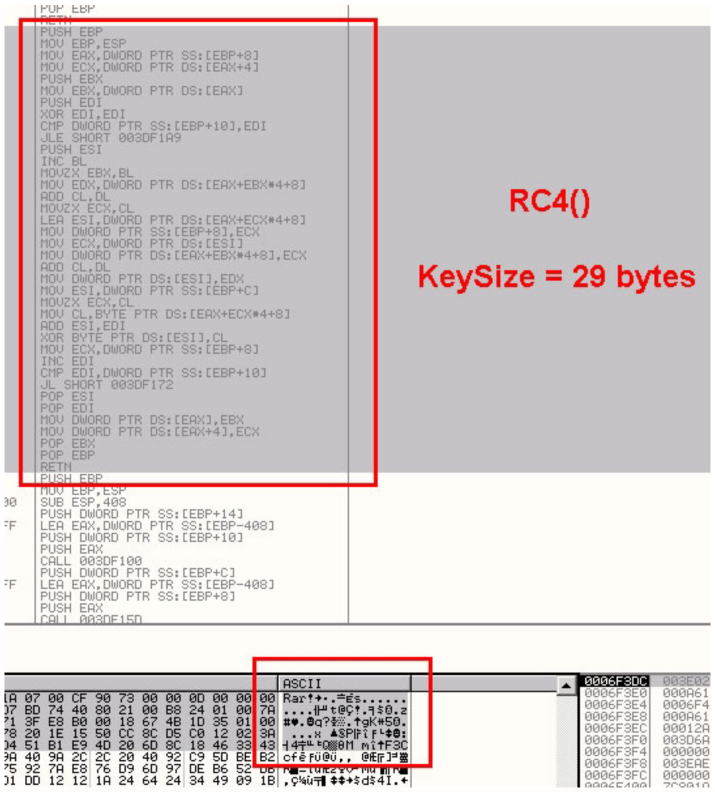
The requested URL /download/geoip/database/GeoIP.dat.gz was not found on this server.

Apache/2.2.11 (Unix) mod_ssl/2.2.11 OpenSSL/0.9.7a Server at geolite.maxmind.com Port 80

When this happened, Downadup authors had several good reasons to release an updated version of the threat and so they decided to fix the GeoIP file problem and increase the propagation of the worm for the next release. When W32.Downadup.B came out on December 30, 2008, the new variant was able to copy across USB and/or network drives and could also infect machines by brute-forcing user passwords, all of which dramatically increased its ability to spread. In addition, the authors fixed the GeoIP problem from the first variant by inserting the GeoIP file directly into the appended data of the threat file. In fact, W32.Downadup.B samples have almost 75K of extra data tagged onto the end, encrypted with RC4 using a 29-byte key. After decryption, this buffer reveals a RAR

Figure 18

Extra encrypted data in W32.Downadup.B



```
POP EBP
RETN
PUSH EBP
MOV EBP,ESP
MOV EAX,DWORD PTR SS:[EBP+8]
MOV ECX,DWORD PTR DS:[EAX+4]
PUSH EBX
MOV EBX,DWORD PTR DS:[EAX]
PUSH EDI
XOR EDI,EDI
CMP DWORD PTR SS:[EBP+10],EDI
JLE SHORT 003DF1A9
PUSH ESI
INC BL
MOVZX EBX,BL
MOV EDX,DWORD PTR DS:[EAX+EBX*4+8]
ADD CL,DL
MOVZX ECX,CL
LEA ESI,DWORD PTR DS:[EAX+ECX*4+8]
MOV DWORD PTR SS:[EBP+8],ECX
MOV ECX,DWORD PTR DS:[ESI]
MOV DWORD PTR DS:[EAX+EBX*4+8],ECX
ADD CL,DL
MOV DWORD PTR DS:[ESI],EDX
MOV ESI,DWORD PTR SS:[EBP+C]
MOVZX ECX,CL
MOV CL,BYTE PTR DS:[EAX+ECX*4+8]
ADD ESI,EDI
XOR BYTE PTR DS:[ESI],CL
MOV ECX,DWORD PTR SS:[EBP+8]
INC EDI
CMP EDI,DWORD PTR SS:[EBP+10]
JL SHORT 003DF172
POP ESI
POP EDI
MOV DWORD PTR DS:[EAX],EBX
MOV DWORD PTR DS:[EAX+4],ECX
POP EBX
POP EBP
RETN
PUSH EBP
MOV EBP,ESP
00   SUB ESP,408
     PUSH DWORD PTR SS:[EBP+14]
FF   LEA EAX,DWORD PTR SS:[EBP-408]
     PUSH DWORD PTR SS:[EBP+10]
     PUSH EAX
     CALL 003DF100
     PUSH DWORD PTR SS:[EBP+C]
FF   LEA EAX,DWORD PTR SS:[EBP-408]
     PUSH DWORD PTR SS:[EBP+8]
     PUSH EAX
     CALL 003DF15D
```

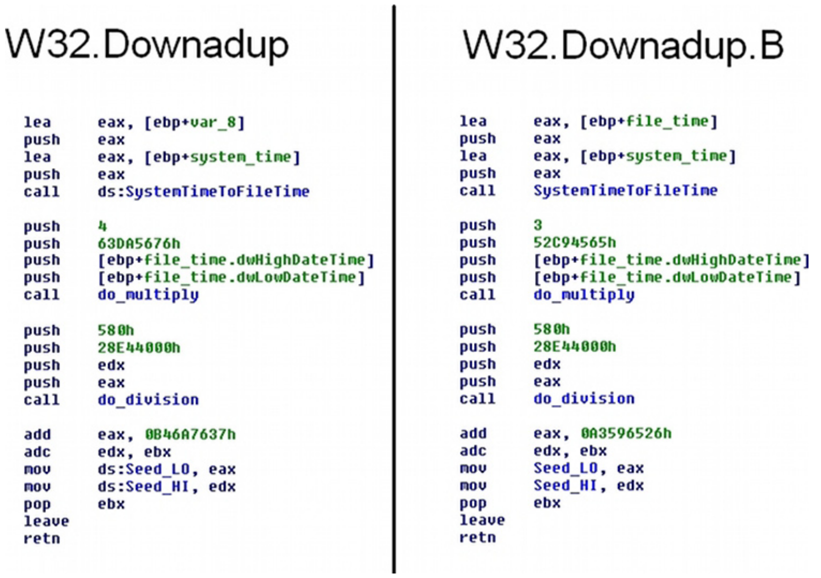
```
                                            ASCII                           0006F3DC  003E02
1A 07 00 CF 90 73 00 00 0D 00 00 00         Rar........s......              0006F3E0  000A61
07 BD 74 40 80 21 00 B8 24 01 00 7A         ....t@Ç..$0.z                   0006F3E4  0006F4
71 3F E8 B0 00 18 67 4B 1D 35 01 00         #..q?...gK#50.                  0006F3E8  000A61
78 20 1E 15 50 CC 8C D5 C0 12 02 3A         ...x .SP......:                 0006F3EC  00012A
04 51 B1 E9 4D 20 6D 8C 18 46 33 43         .4....Q.M m..F3C                0006F3F0  003D6A
9A 40 9A 2C 2C 20 40 92 C9 5D BE B2         cf.fü@ü,, @..]..                0006F3F4  000000
75 92 7A E8 76 D9 6D 97 DE B6 52 DB                                         0006F3F8  003EAE
01 DD 12 12 1A 24 64 24 34 49 09 1B         ,ç¼ü. ...$d$4I..                0006F3FC  000000
```

archive that contains all of the GeoIP information necessary for the targeted exploitation routine. This data is decrypted and decompressed in memory on-the-fly by the malware, and then re-encrypted to avoid memory forensics.

Another interesting change introduced by authors in the W32.Downadup.B code is the variation of the pseudo-random domain name generation algorithm. Researchers at Symantec and other security companies were able to reverse-engineer the Downadup code and successfully crack the domain-generation algorithm. It is well known that the worm generates a set of 250 different domains every day; therefore, being able to predict these domains may help in tracking infected computers and also preventing further infections. The authors introduced small changes in the .B variant to produce a slightly different list of pseudo-random domains, most likely to attempt to stymie these reverse-engineering efforts.

Figure 19

Comparison of domain-generation code



W32.Downadup

```
lea     eax, [ebp+var_8]
push    eax
lea     eax, [ebp+system_time]
push    eax
call    ds:SystemTimeToFileTime

push    4
push    63DA5676h
push    [ebp+file_time.dwHighDateTime]
push    [ebp+file_time.dwLowDateTime]
call    do_multiply

push    580h
push    28E44000h
push    edx
push    eax
call    do_division

add     eax, 0B46A7637h
adc     edx, ebx
mov     ds:Seed_LO, eax
mov     ds:Seed_HI, edx
pop     ebx
leave
retn
```

W32.Downadup.B

```
lea     eax, [ebp+file_time]
push    eax
lea     eax, [ebp+system_time]
push    eax
call    SystemTimeToFileTime

push    3
push    52C94565h
push    [ebp+file_time.dwHighDateTime]
push    [ebp+file_time.dwLowDateTime]
call    do_multiply

push    580h
push    28E44000h
push    edx
push    eax
call    do_division

add     eax, 0A3596526h
adc     edx, ebx
mov     Seed_LO, eax
mov     Seed_HI, edx
pop     ebx
leave
retn
```

The PRNG* algorithm relies on a seed value that will be the same across infected systems every day. The seed is generated using a set of 64-bit mathematical operations using both static values and the numeric values of the current year, month, and day. These values are three magic numbers used respectively as multiplier (M), divisor (D), and additive (A) constant. The PRNG routine is a 200-byte piece of code that performs different floating-point operations and uses a second internal multiplier value (M2), which is also hardcoded. What's changed between the two worm variants are these magic values M, D, A, and M2, while the logic of the PRNG algorithm is exactly the same. What Downadup is doing with these domains is simple: downloading and executing additional malicious content. It's still a mystery as to what this additional content is, but we are speculating that it is somehow related to eastern-European cyber criminals and misleading applications.

A final question that came to mind was, seeing as the list of the future domains was publicly disclosed on the Web, why hadn't any other cyber criminals taken advantage of the predictions? We know that security vendors registered some of these domains for monitoring purposes, so why didn't some other miscreant try to register one of these domains to push out another Trojan to all machines infected with Downadup? Cyber criminals are not new to these things and it won't be the first case of a stolen botnet. Well, we found a valid explanation for this when we looked into Downadup downloading additional code. As we have said previously, the authors of Downadup are not beginners and they may have the feeling that someone—sooner or later—would break their domain prediction algorithm. So, to avoid losing their botnet, they put a secondary (strong) protection into the threat, which makes it impossible for anyone (other than the original authors) to upload new malicious components onto compromised machines.

* Thanks to the Symantec DeepSight team and to Symantec's Aaron Adams for the superb analysis while reversing the PRNG algorithm.

Attempts at smart network scanning

Originally published January 23, 2009 by Eric Chien

The ability of a threat to widely replicate often depends on its algorithm of finding other computers on the Internet, which are represented by an IP address. Downadup uses a variety of techniques to scan for new machines in order to maximize its infection abilities and at the same time minimize the chance of being noticed on a host.

Brute-force network scanning can cause noticeable slowdowns and network issues on the infected machine. Downadup attempts to limit its impact in two ways. Firstly, the worm contacts two well-known Web sites and calculates the computer's average bandwidth, then uses this value to configure how many simultaneous remote procedure call (RPC) exploit scans are allowed at one time. Secondly, a pause—between 100 milliseconds and two seconds—is taken after each scan, depending on the type of scan and if the computer is currently being used. (Downadup checks active usage by determining if a keystroke was made in the previous five minutes.)

Downadup attempts four different scans that are repeated in an infinite loop. It scans for machines on the same subnet, machines it has successfully infected previously, machines nearby those already infected, and randomly selected machines.

First, Downadup sequentially scans all the IPs in the same subnet of the infected machine, starting from the first IP in the subnet. This can include multiple subnets for multi-homed machines (machines with more than one IP address).

Next, Downadup attempts to exploit previously infected machines. This serves two purposes—one, to reinfect machines that may have been cleaned up and two, to initiate the peer-to-peer (P2P) communication channel to receive payload files (as described in the blog article Downadup: Peer-to-Peer Payload Distribution). The worm only remembers the last 100 successfully infected machines.

Then, Downadup begins generating random IP addresses to attack. In addition to what is likely a bug rather than a feature in the random generation routine, certain IP addresses are ruled out, therefore potentially limiting certain networks from being attacked. Downadup is only able to generate approximately a quarter of the four billion possible IP addresses, which limits its ability to reach certain IP addresses via the RPC exploit.

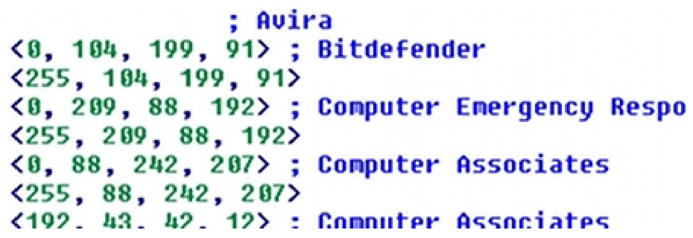
Finally, in parallel, Downadup will also scan machines near other machines that were successfully exploited. For each exploited machine, Downadup scans the class C-sized (/24) block of the IP address and the previous ten class C-sized (/24) blocks. For example, if the successfully exploited machine is 208.77.188.166, Downadup will scan the range 208.77.**178.1** to 208.77.**188.255**.

Further, Downadup doesn't scan every IP address in the calculated ranges. For example, invalid IP ranges such as 127.x.x.x or 169.254.x.x. are skipped. But more importantly, Downadup carries a large blacklist of IP ranges that belong to security vendors. A snippet of the list is shown in Figure 19.

By not attempting to exploit security vendors, Downadup potentially avoids honeypot systems. This blacklist is also used to reject back-connect attempts as well, preventing security vendors from contacting infected hosts and gaining payload files.

Figure 20

Blacklisted IP ranges in Downadup



```
                    ; Avira
<0, 104, 199, 91> ; Bitdefender
<255, 104, 199, 91>
<0, 209, 88, 192> ; Computer Emergency Respo
<255, 209, 88, 192>
<0, 88, 242, 207> ; Computer Associates
<255, 88, 242, 207>
<192, 43, 42, 12> ; Computer Associates
```

Downadup will then refresh the list of IP addresses configured on the local machine. If any have changed since any of the related scans started, the scans will be terminated because the exploit is designed to connect back to the previously configured IP address.

Knowing what IP address to connect back to raises another issue for Downadup. With many home users behind wireless routers, firewalls, and using network address translation (NAT), many infected machines are normally not contactable from external machines. Downadup goes to great lengths to bypass these issues. We'll investigate these techniques in a future blog article in this W32.Downadup series.

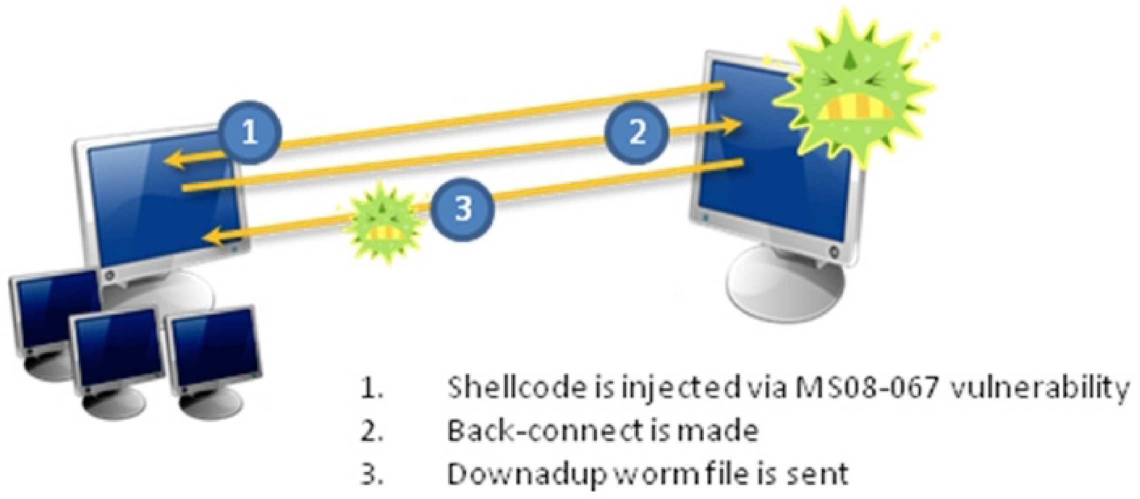
Playing with Universal Plug and Play

Originally published January 28, 2009 by Eric Chien

Among other methods, Downadup infects other machines via a remote procedure call (RPC) exploit against the MS08-067 vulnerability. Using the vulnerability, the worm injects shellcode that connects back to the infecting machine. This is known as a back-connect. The back-connect works via HTTP on a randomly selected port and the infecting machine responds to incoming requests by providing the entire worm file. The shellcode receives this file and executes it on the remote host, causing it to then become infected.

Figure 21

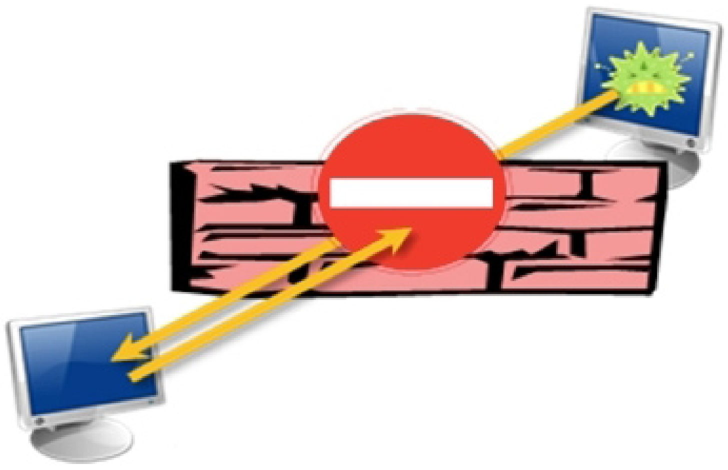
Downadup back-connect feature


1
2
3
1. Shellcode is injected via MS08-067 vulnerability
2. Back-connect is made
3. Downadup worm file is sent

However, many home users today use routers or other Internet gateway devices that by default prevent external machines from connecting their home machines, in addition to using network address translation (NAT). This would normally prevent Downadup's shell-code from successfully completing infection because the back-connect would fail.

Figure 22

Failed back-connect due to gateway device



To bypass this issue, Downadup needs to perform three actions; determine if the machine is behind a gateway device, obtain the IP address on the outside interface of the gateway device (this is the IP that is visible to external machines), and also make sure incoming connection attempts to certain ports on the gateway device are passed through to the internal machine. This is known as port forwarding. Downadup uses the Universal Plug-and-Play (UPnP) protocol to achieve these tasks.

The UPnP protocol is supported by default in many common gateway devices that are in use in home user environments. To achieve the first task, Downadup harnesses UPnP's discovery protocol, which is based on the Simple Service Discovery Protocol (SSDP). The discovery protocol allows machines on the network to find gateway devices that are also on the network.

As part of SSDP, Downadup sends an M-SEARCH request to the multicast address 239.255.255.250 on port 1900/udp and then listens for responses. The M-SEARCH request requires a header known as the search target (represented by "ST") that represents the types of devices or services Downadup is looking for. These are represented by universal resource identifiers (URIs). Downadup looks for the following four devices or services:

1. urn:schemas-upnp-org:device:InternetGatewayDevice:1
2. urn:schemas-upnp-org:service:WANIPConnection:1
3. urn:schemas-upnp-org:service:WANPPPConnection:1
4. upnp:rootdevice (represents all UPnP devices)

Here is an example of the contents of an M-SEARCH request packet:

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
ST: urn:schemas-upnp-org:device:InternetGatewayDevice:1
MAN: "ssdp:discover"
MX: 3
```

If a matching device exists on the network, the device will respond with a message that contains an additional URL that provides information about the device and the services the device supports. After verifying the device is suitable, Downadup sends a UPnP GetStatusInfo request to ensure the device is currently connected on the external wide area network (WAN) interface. This completes the first requirement of determining the infected machine is behind a gateway device.

Next, an UPnP GetExternalIPAddress command is sent to the device to obtain the external IP address, completing the second requirement of having the IP address on the external interface. This is the IP address that is visible to other machines on the Internet.

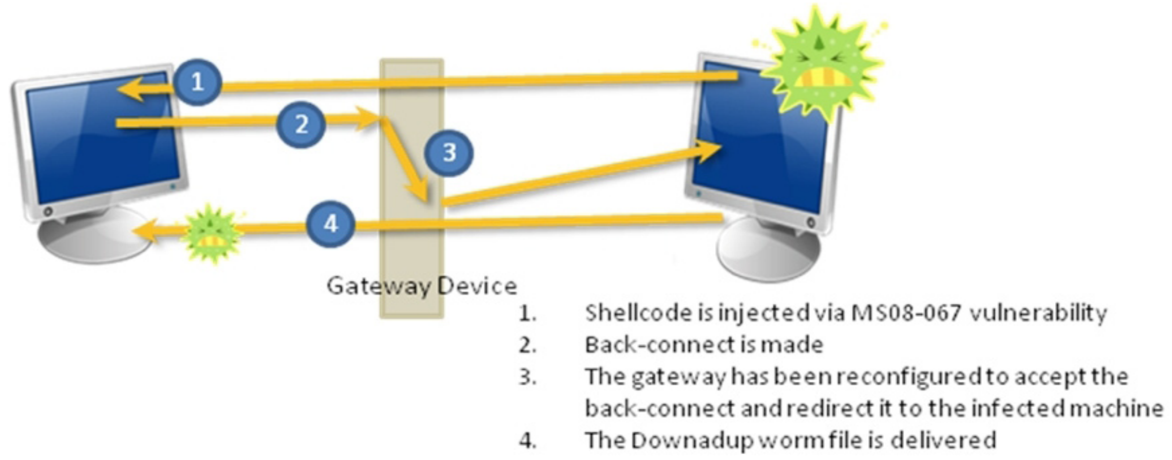
Finally, Downadup needs to port forward incoming requests through the gateway and to itself on the internal machine. A port forwarding request requires a few parameters; the relevant ones in this case are the port forwarding entry description, the external listening port, and the internal IP and port to forward to.

The description for the port forwarding entry is generated from the Volume Serial Number and the Computer Name of the infected system. This forms a relatively unique description. Downadup sends the GetGenericPortMappingEntry command to the gateway device to enumerate all of the existing port forwarding entries in order to see if any of the descriptions match the generated description. If they match, the worm assumes they are prior entries created by itself and are deleted using the UPnP DeletePortMapping command.

A new port forwarding entry is then added using an AddPortMapping request. Downadup attempts to use port 80 for the external port and the internal port is randomly generated. If the configuration change fails, two more attempts will be made, but with a randomly generated external port number between 1024 and 10000.

Figure 23

Back-connect after modifying gateway



1. Shellcode is injected via MS08-067 vulnerability
2. Back-connect is made
3. The gateway has been reconfigured to accept the back-connect and redirect it to the infected machine
4. The Downadup worm file is delivered

This completes the process, allowing back-connects to be passed from the external network into the internal network. It therefore allows Downadup to successfully complete its infection in common home-user scenarios.

Astute readers may have noted that while this procedure allows Downadup to infect other machines when it is behind a gateway device, machines behind their own gateway devices are still protected (if they are dropping RPC and other network traffic). Unfortunately, Downadup can be introduced into a private network through other replication means and once inside a private network is vigorous in infecting the entire local network.

Advanced users may wish to consult their gateway device manual and determine if they can disable UPnP to prevent undesired modifications to their gateway security.

Locking itself out

Originally published February 18, 2009 by Eric Chien

While Downadup's RPC exploit method of spreading has been highlighted in several recently posted blog articles, the worm spreads via other methods as well. One of the potentially more noticeable methods is through network shares, especially in enterprise environments.

Downadup attempts to copy itself to other machines using the administrative network share (ADMIN$) that exists by default on Microsoft Windows machines. However, copying itself to the share requires authentication. This requirement leads to some noticeable side effects.

Downadup first enumerates all of the servers in the network by making a NetServerEnum request, which returns all of the visible Windows machines on the network. Downadup then attempts to infect each of these machines.

To become authenticated, the credentials of the locally logged-on user are tried first. However, if that does not work, Downadup begins trying different user name and password pairs.

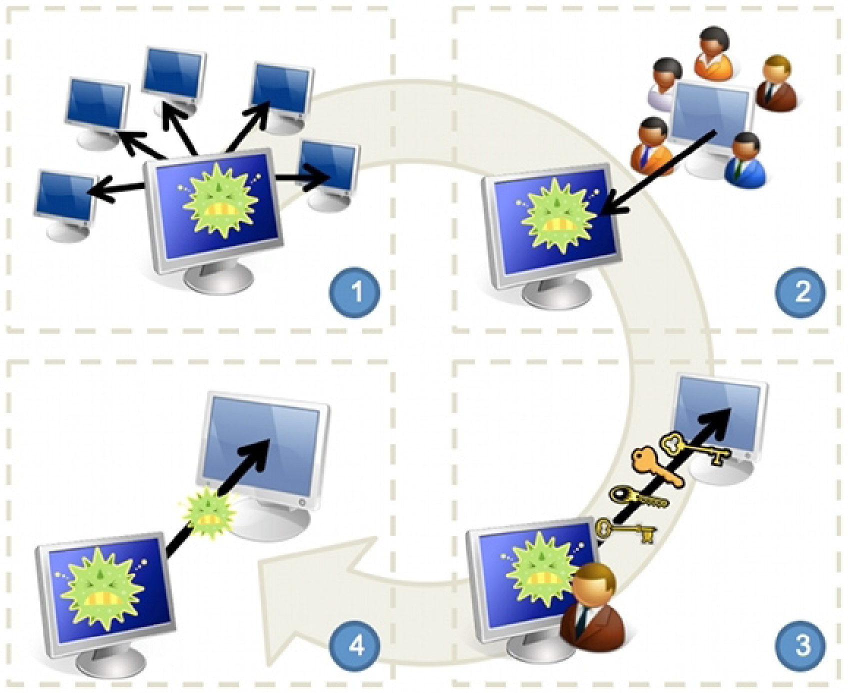
The remote server is queried for all of the user names available. Fortunately, most Windows XP and later systems will not provide this information by default and in those cases all of the user names on the local machine will be used instead.

Rich with user names, Downadup now tries to connect to the remote server with each user name and a variety of passwords, including:

- The user name
- The user name concatenated together twice (e.g. joesmithjoesmith)
- The user name reversed (e.g. htimseoj)
- Almost 250 common passwords such as "password", "123", and "admin"

Figure 24

Password authentication method

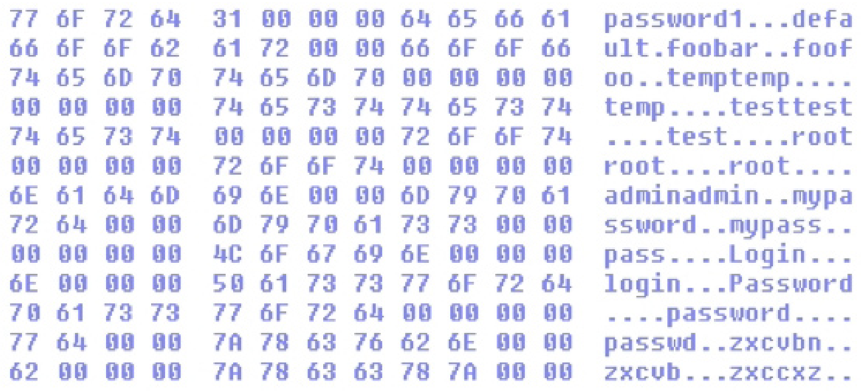


1. All machines are found on the network
2. Usernames are obtained from each machine
3. Different passwords are guessed for each user
4. Once authenticated, Downadup is copied to the remote machine

Although it is potentially clever, an immediate side effect of this password guessing is a rush of incoming calls to enterprise IT helpdesks from users who have been locked out of their account due to security policy rules that cause account lockout after several invalid attempts. This side effect can become even more problematic because Downadup does have the ability to enumerate all existing user names; therefore, account lockout can suddenly occur for all of the users in an organization—even if only a single user or machine is infected.

Figure 25

A snippet of some of the passwords used



```
77 6F 72 64  31 00 00 00 64 65 66 61  password1...defa
66 6F 6F 62  61 72 00 00 66 6F 6F 66  ult.foobar..foof
74 65 6D 70  74 65 6D 70 00 00 00 00  oo..temptemp....
00 00 00 00  74 65 73 74 74 65 73 74  temp....testtest
74 65 73 74  00 00 00 00 72 6F 6F 74  ....test....root
00 00 00 00  72 6F 6F 74 00 00 00 00  root....root....
6E 61 64 6D  69 6E 00 00 6D 79 70 61  adminadmin..mypa
72 64 00 00  6D 79 70 61 73 73 00 00  ssword..mypass..
00 00 00 00  4C 6F 67 69 6E 00 00 00  pass....Login...
6E 00 00 00  50 61 73 73 77 6F 72 64  login...Password
70 61 73 73  77 6F 72 64 00 00 00 00  ....password....
77 64 00 00  7A 78 63 76 62 6E 00 00  passwd..zxcvbn..
62 00 00 00  7A 78 63 63 78 7A 00 00  zxcvb...zxccxz..
```

If Downadup correctly guesses a password before being locked out, it will copy itself to the System32 folder on the remote machine via the administrative share (ADMIN$) as a random file name with a random extension. (Later, when the file is executed, it will remove itself and copy itself as another random file name with the extension "DLL".) The file time of the file is then changed to the same file time as kernel32.dll to avoid suspicion.

Once the file is placed on the remote machine, it still needs to be executed. A scheduled job on the remote machine is created so that the file is executed on the next hour, based on the local time of the infecting machine. So, if the infecting machine time is 2:36 PM, the remote machine will have a scheduled job at 3:00PM to execute the file. Execution actually happens via rundll32.exe because the file copied over is a DLL.

After all servers, user names, and passwords are tried, Downadup will wait 40 minutes and then try all over again. This of course has the potential to cause accounts to be locked out again, resulting in yet more IT helpdesk calls.

A new Downadup variant?

Originally published February 23, 2009 by Patrick Fitzgerald

Over the last few days many reports have emerged concerning a new variant of Downadup (a.k.a. Conficker), which has been dubbed Downadup.B++ or Conficker.C. While one could categorize Downadup into three variants (or even more), Symantec products will detect all known variants of Downadup as either W32.Downadup or W32.Downadup.B.

Unfortunately, in addition to differences in names, variant differentiation also exists between vendors. Some vendors have a different detection for every single Downadup binary—with a differing MD5 hash—resulting in more than 30 different Downadup "variants". Some others don't differentiate at all and just have a single name with no variant differentiation.

However, the important point regarding Downadup is not whether this is another variant, but rather is it a new variant; i.e., if it has been released recently. Fortunately, Downadup.B++ / Conficker.C is not a newly released variant. This variant has been around since the main outbreak of Downadup, and most vendors already have detections for it.

The main item that has prompted the industry to highlight this sample as another variant is the emergence of its peer-to-peer behavior. This behavior was analyzed and discussed previously, in detail, by Eric Chien in the blog entry Downadup: Peer-to-Peer Payload Distribution. Symantec customers have been protected from this Downadup.B++ / Conficker.C variant for some time now, as long as they have kept their definitions up to date.

Advanced crypto protection

Originally published February 23, 2009 by Elia Florio

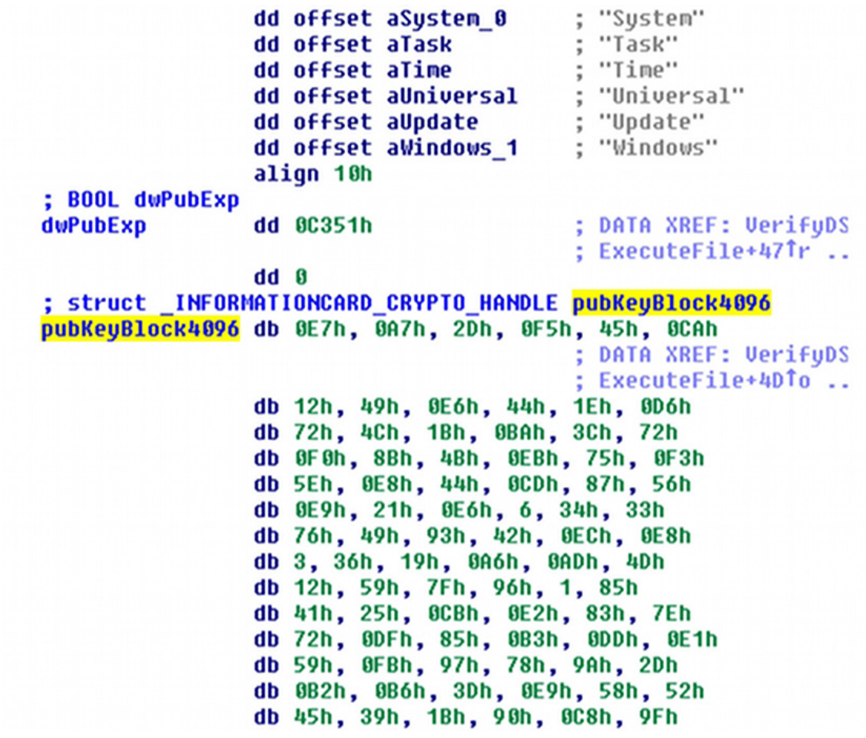
The conclusion of my previous blog posed an interesting question to readers: "...seeing as the list of the future domains was publicly disclosed on the Web, why hadn't any other cyber criminals taken advantage of the predictions?" Antivirus companies and many independent security researchers were able to crack the domain prediction algorithm used by the worm, so it is reasonable to believe that other people were able to achieve the same result, but with different intentions. In fact, predicting what the next domain will be creates the perception that someone can take control over the botnet, and, for example, start pushing a bank Trojan to these millions of machines infected with Downadup. We already know that there's no honor among thieves, and so it won't be a surprise if we see future Downadup domains registered by other criminals. There are also credible reports that this is already happening because some of the future domains pointed to well-known IP addresses used by the ASProx gang in the past.

Unfortunately, the takeover is not such a simple job. Registering a domain with the intent of stealing the Downadup botnet is not enough. Nobody, except Downadup's authors, will be able to push files to the botnet because of some advanced security measures utilized by the gang.

We analyzed the downloading routine of W32.Downadup.B and we found that is secured by asymmetric cryptography. The digital signature and very long crypto keys make this job impossible. The payload downloaded from one of the domains is encrypted with RC4 (using a 64-bytes key) and then digitally signed by the authors using their private key. In fact, the worm binary contains the corresponding public key (4096-bit) and a value that looks like the public exponent. The algorithm looks similar to RSA encryption, or perhaps it's just one of the asymmetric algorithms available today.

Figure 26

Public crypto key hardcoded in the worm binary

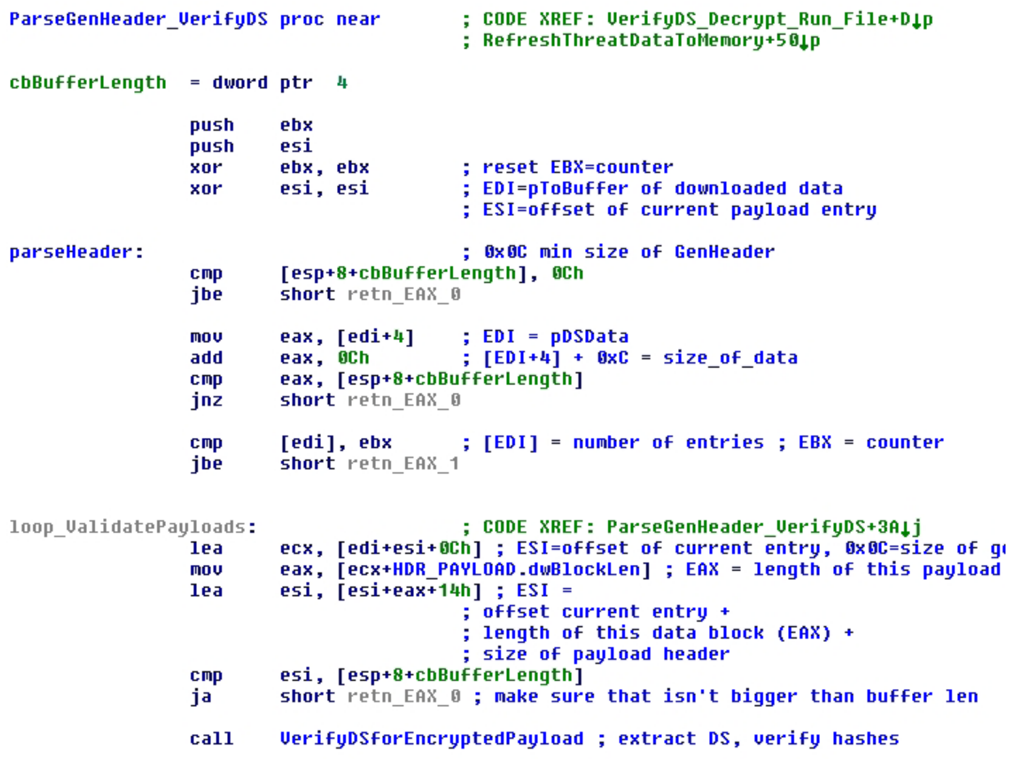


```
                    dd offset aSystem_0      ; "System"
                    dd offset aTask          ; "Task"
                    dd offset aTime          ; "Time"
                    dd offset aUniversal     ; "Universal"
                    dd offset aUpdate        ; "Update"
                    dd offset aWindows_1     ; "Windows"
                    align 10h
; BOOL dwPubExp
dwPubExp            dd 0C351h                ; DATA XREF: VerifyDS
                                             ; ExecuteFile+47↑r ..
                    dd 0
; struct _INFORMATIONCARD_CRYPTO_HANDLE pubKeyBlock4096
pubKeyBlock4096 db 0E7h, 0A7h, 2Dh, 0F5h, 45h, 0CAh
                                             ; DATA XREF: VerifyDS
                                             ; ExecuteFile+4D↑o ..
                    db 12h, 49h, 0E6h, 44h, 1Eh, 0D6h
                    db 72h, 4Ch, 1Bh, 0BAh, 3Ch, 72h
                    db 0F0h, 8Bh, 4Bh, 0EBh, 75h, 0F3h
                    db 5Eh, 0E8h, 44h, 0CDh, 87h, 56h
                    db 0E9h, 21h, 0E6h, 6, 34h, 33h
                    db 76h, 49h, 93h, 42h, 0ECh, 0E8h
                    db 3, 36h, 19h, 0A6h, 0ADh, 4Dh
                    db 12h, 59h, 7Fh, 96h, 1, 85h
                    db 41h, 25h, 0CBh, 0E2h, 83h, 7Eh
                    db 72h, 0DFh, 85h, 0B3h, 0DDh, 0E1h
                    db 59h, 0FBh, 97h, 78h, 9Ah, 2Dh
                    db 0B2h, 0B6h, 3Dh, 0E9h, 58h, 52h
                    db 45h, 39h, 1Bh, 90h, 0C8h, 9Fh
```

At the moment, no download has been observed from Downadup-infected machines, so nobody is able to tell you what the payload is, presumed to be downloaded over HTTP at some point in the future by the botnet. So, we can't tell you today what the unwanted gift will be, but we can tell you what it will look like:

Figure 27

Code snippet of the header-parsing routine

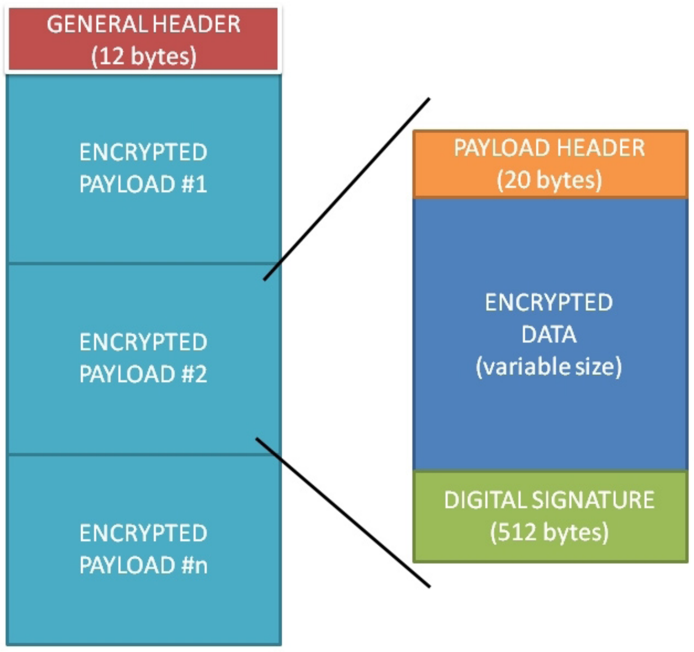


```
ParseGenHeader_VerifyDS proc near      ; CODE XREF: VerifyDS_Decrypt_Run_File+D↓p
                                       ; RefreshThreatDataToMemory+50↓p

cbBufferLength  = dword ptr  4

                push    ebx
                push    esi
                xor     ebx, ebx       ; reset EBX=counter
                xor     esi, esi       ; EDI=pToBuffer of downloaded data
                                       ; ESI=offset of current payload entry

parseHeader:                           ; 0x0C min size of GenHeader
                cmp     [esp+8+cbBufferLength], 0Ch
                jbe     short retn_EAX_0

                mov     eax, [edi+4]   ; EDI = pDSData
                add     eax, 0Ch       ; [EDI+4] + 0xC = size_of_data
                cmp     eax, [esp+8+cbBufferLength]
                jnz     short retn_EAX_0

                cmp     [edi], ebx     ; [EDI] = number of entries ; EBX = counter
                jbe     short retn_EAX_1


loop_ValidatePayloads:                 ; CODE XREF: ParseGenHeader_VerifyDS+3A↓j
                lea     ecx, [edi+esi+0Ch] ; ESI=offset of current entry, 0x0C=size of g
                mov     eax, [ecx+HDR_PAYLOAD.dwBlockLen] ; EAX = length of this payload
                lea     esi, [esi+eax+14h] ; ESI =
                                       ; offset current entry +
                                       ; length of this data block (EAX) +
                                       ; size of payload header
                cmp     esi, [esp+8+cbBufferLength]
                ja      short retn_EAX_0 ; make sure that isn't bigger than buffer len

                call    VerifyDSforEncryptedPayload ; extract DS, verify hashes
```

Looking at the code snippet above, we can say that "package" downloaded by Downadup will have a general header with a size of 12 bytes, followed by multiple instances of encrypted and digitally signed payloads. Each of these payloads will have its own header (20 bytes) and will contain an appended blob of data at the end (512 bytes), which is the unique digital signature of the payload. The following diagram shows the structure of the whole package and that of a single payload.

Figure 28

Structure of a W32.Downadup.B payload


GENERAL HEADER (12 bytes)
ENCRYPTED PAYLOAD #1
ENCRYPTED PAYLOAD #2
ENCRYPTED PAYLOAD #n
PAYLOAD HEADER (20 bytes)
ENCRYPTED DATA (variable size)
DIGITAL SIGNATURE (512 bytes)

Downadup authors tried to put basic safety checks in place so the code verifies that the received package is bigger than 12 bytes, and also that the size stored in the general header matches the size of data received. There are also checks to avoid reading after the end of the data (ironically, they seem very afraid of having buffer overflows in the code!).

The structures and the fields of the headers are defined below:

GENERAL_HEADER (size = 12 bytes)

```
00   DW     num _ of _ entries    ; num of payloads in the package
04   DW     size _ of _ data      ; size of data without this header
08   DW     (not _ used)          ; ?
```

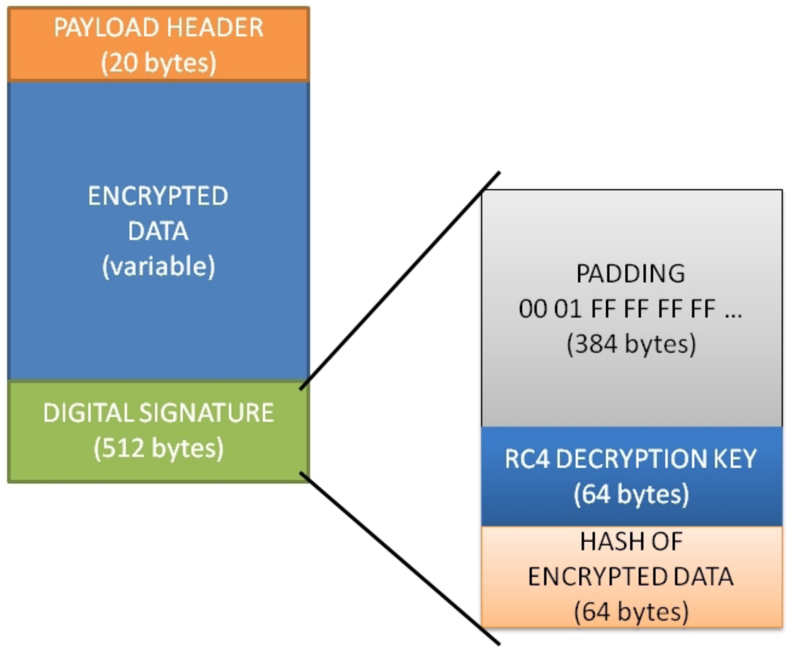
PAYLOAD_HEADER (size = 20 bytes)

```
00   DW     dwFileNumber ; unique identifier for this payload
04   DW     dwUnknownX            ; ?
08   DW     dwExpTime _ lo        ; expiration time (low)
0C   DW     dwExpTime _ hi        ; expiration time (high)
10   DW     dwLengthOfFile        ; size of encrypted data
```

The verifier routine runs a loop that parses the header of each payload to locate the end of the encrypted data (dwLengthOfFile) where the Digital Signature is stored. Using the asymmetric algorithm and public exponent/modulus values, the code decrypts the 512-byte digest and extracts two important things: the real decryption key (64-bytes, used later with RC4 algorithm) and the hash value of the encrypted data (64-bytes). The following diagram shows what the worm expects after asymmetric decryption of the digital signature.

Figure 29

Structure of digital signatures in payloads


PAYLOAD HEADER (20 bytes)
ENCRYPTED DATA (variable)
DIGITAL SIGNATURE (512 bytes)
PADDING 00 01 FF FF FF FF ... (384 bytes)
RC4 DECRYPTION KEY (64 bytes)
HASH OF ENCRYPTED DATA (64 bytes)

Finally, the worm performs all of the validations that will allow it to verify if the payload is trusted and has been created by the real author, by following these steps:

1. It verifies that the hash extracted from the signature matches with the hash calculated over the encrypted data (the encrypted message has not been altered);
2. It runs RC4 decryption over the encrypted data with the 64-bytes key extracted from the signature;
3. It calculates the hash of the decrypted data after RC4 and verifies that this new hash matches with the decryption key (i.e., the encryption key is the hash of the plaintext message).

The hashing algorithm used by the worm does not look similar to any of the well-known algorithms. It produces a hash value of 64-bytes (512-bits), but it's not the popular SHA-512. We just know that the hash value of null (=empty file) produced by this custom algorithm is the following sequence:

```
6B 7F 33 82 1A 2C 06 0E CD D8 1A EF DD EA 2F D3 C4 72 02 70 E1 86 54 F4 CB 08 EC E4 9C CB 46 9F
8B EE EE 7C 83 12 06 BD 57 7F 9F 26 30 D9 17 79 79 20 3A 94 89 E4 7E 04 DF 4E 6D EA A0 F8 E0 C0
```

After all of this analysis, the conclusion is that the Downadup authors have taken extreme measure to protect their job and this again demonstrates the level of professionalism of these criminals. We suppose that these crooks are afraid of just two things right now: the $250K bounty reward offered by Microsoft, and the risk of having an exploitable buffer overflow in their code. In fact, while the crypto protection can't be easily bypassed, it may be possible that their code has some exploitable overflow condition that can allow some other bad guy to push a malformed package that will DoS or exploit the worm. The code seems to have basic checks against large buffers; but, for example, there could be problems with a buffer smaller than expected or with integer wrap caused by invalid sizes. But, that's another story.

Big thanks to Eric Chien for his help with this analysis.

Propagation by AutoPlay

Written by Ben Nahorney and John Park

Downadup has become one of the most prolific worms in recent years based on its exploitation of MS08-067. But taking advantage of this vulnerability is far from its only propagation technique, and its methods to spread are not just limited to those that work over network connections. As with many threats that come before it, W32.Downadup.B takes advantage of the AutoPlay feature available in Windows operating systems (OS). Yet the worm exhibits a couple of variations on the traditional AutoPlay tricks so often seen from other threats.

The first of these tricks falls into the realm of social engineering. Say you happen to have a USB drive infected with W32.Downadup.B and you insert the drive into a USB port of a computer running Windows XP. When AutoPlay is triggered (assuming you haven't disabled it already) you may see a dialog box similar to Figure 30.

A cursory glance at the dialog box gives the impression that the first choice offered will open a folder on the drive so you can view the files. But looking closer, three things stand out. For starters, the first sentence hints that Windows wants to know what you would like to do with the *file* you are about to open. Secondly, the type of file is a *Program*, as indicated by the icon displayed below the first sentence. Finally, the first option available offers to "open the folder to view files using the *program provided on the device*". Windows is not offering to open it with one of its built-in utilities—it's asking you if you want to launch a program on the USB device.

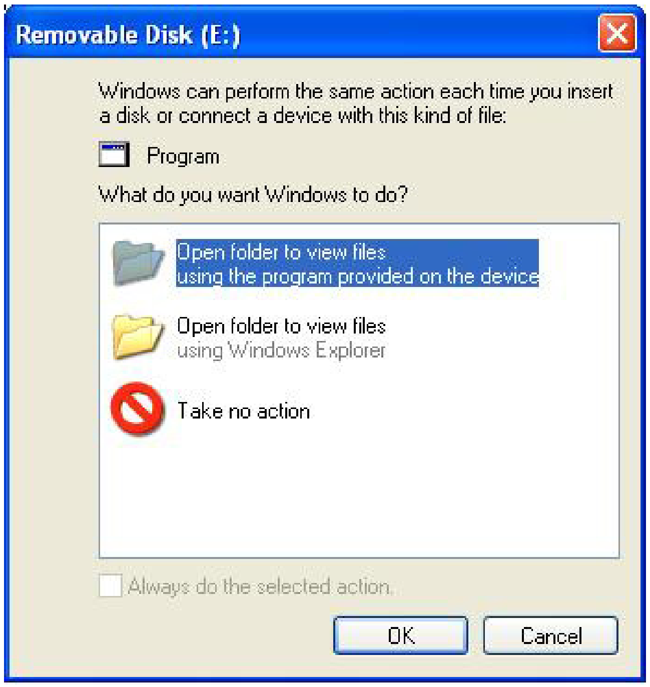
Now a novice or intermediate computer user may not understand what Windows is asking in this case, even if they did read the details carefully. To Microsoft's credit, they have addressed this to some extent in Windows Vista and later versions of their operating systems. If the same USB drive is plugged into a computer running Vista, they will see the AutoPlay dialog box in Figure 31.

The option to run the malicious program is now listed under **Install or run program**, separated from standard Windows features that are under **General options**. Another clue is the option to "Always do this for *software* and games".

The problem here is that many people do not read AutoPlay alerts carefully. The Downadup authors know this and are attempting to take advantage of it, having written their autorun.inf file in such a way that it appears as though you are selecting an option to open a folder. Ultimately this is a lie, craftily presented in order to get you to execute the malicious code from the USB device.

Figure 30

Downadup AutoPlay in Winodws XP



Removable Disk (E:)

Windows can perform the same action each time you insert a disk or connect a device with this kind of file:

Program

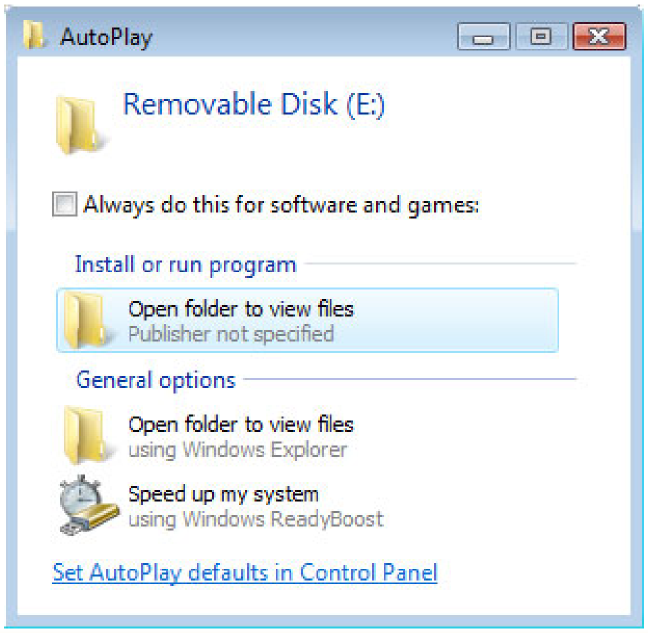
What do you want Windows to do?

- Open folder to view files using the program provided on the device
- Open folder to view files using Windows Explorer
- Take no action

☐ Always do the selected action.

Figure 31

Downadup AutoPlay in Windows Vista



AutoPlay

Removable Disk (E:)

☐ Always do this for software and games:

Install or run program

- Open folder to view files — Publisher not specified

General options

- Open folder to view files — using Windows Explorer
- Speed up my system — using Windows ReadyBoost

Set AutoPlay defaults in Control Panel

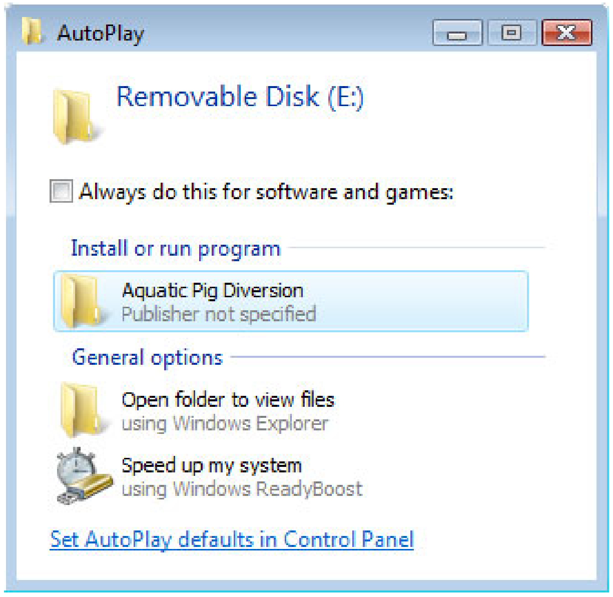
To demonstrate how this is done, let's take a look at the functional text within the autorun.inf file used by the worm:

```
[autorun]
Action=Open folder to view files
Icon=%systemroot%\system32\shell32.dll,4
Shellexecute=RUNDLL32.EXE .\RECYCLER\[RANDOM NUMBERS]\[5-8 RANDOM LETTERS].[RANDOM EXTENSION]
UseAutoPlay=1
```

The two lines responsible for this social engineering trick are the `Action=` and `Icon=` lines. The first one displays the text "Open folder to view files"— the same text Windows uses for opening a folder in Windows Explorer. When writing an autorun.inf file, you can generally get this to say anything you would like Figure 32.

Figure 32

Arbitrary text in AutoPlay notification



The `Icon=` line helps the ruse by calling the fifth icon located in the shell32.dll file—a Windows folder icon. When combined, these two lines in the autorun.inf file give the impression that choosing this option only opens a folder.

Now it's the `Shellexecute=` line that carries the malicious punch. When W32.Downadup.B copies itself and its autorun.inf file to a removable drive, it puts its malicious DLL in the hidden %DriveLetter%\RECYCLER folder—the Recycle Bin for the drive in question.

When quickly perusing a removable drive, you might not notice its presence here, since the folder is hidden by default. Even if you did look carefully, you'd find another hidden folder within the Recycle Bin, with a name like "S-n-n-nn-nnnnnnnnnn-nnnnnnnnnn-nnnnnnnnn-nnnn", where each "n" is a random number. This is meant to mimic folders often used by Security Identifiers (SIDs)—a unique number assigned by the OS to identify a user or group within a network of Windows computers. Folders based on SIDs are not an uncommon sight when viewing the hidden contents of a %DriveLetter%\RECYCLER folder. By mimicking these folders, the threat does not look out of place.

As a final precaution against discovery, the actual W32.Downadup.B file that is copied to this folder is given a random name, followed by a random extension. Any random extension, that is, except "DLL"—the actual file type. What we have here is a DLL in sheep's clothing. In the end, without the clues present in the autorun.inf file, the malicious .dll file looks like nothing more than junk within an SID folder in the removable drive's Recycle Bin.

So here is where it gets its teeth: when the `Shellexecute=` line runs, it calls rundll32.exe—the process responsible for executing DLLs and then placing their libraries into system memory. In this case the malicious W32.Downadup.B libraries are loaded and the compromised computer is infected by the worm.

The final `UseAutoPlay=1` line of the autorun.inf file attempts to run its code without notifying the user with an AutoPlay dialog box, similar to the ones shown above. Unless the feature has been disabled, this results in the automatic execution of the DLL on Windows XP SP2 and older operating systems.

This functionality is fairly cut-and-dry in terms of how AutoPlay operates, but W32.Downadup.B puts one more wrench into the works—it adds junk data to the autorun.inf file. It's not a small amount of junk data either, taking up the vast majority of the file.

Yet AutoPlay still succeeds in carrying out the instructions in the file. This is because, when Windows recognizes the file as an autorun.inf file, it starts hunting for very particular instructions that it expects to see. First it searches out the expected `[autorun]` header, then any text that falls within a limited series of commands considered valid within an autorun.inf file. Any characters other than what is expected within the limited confides of AutoPlay commands are completely ignored. Not only that, but the commands don't have to be uninterrupted strings, and Windows will ignore junk data that is inserted into the middle of a valid AutoPlay string, as shown in Figure 33.

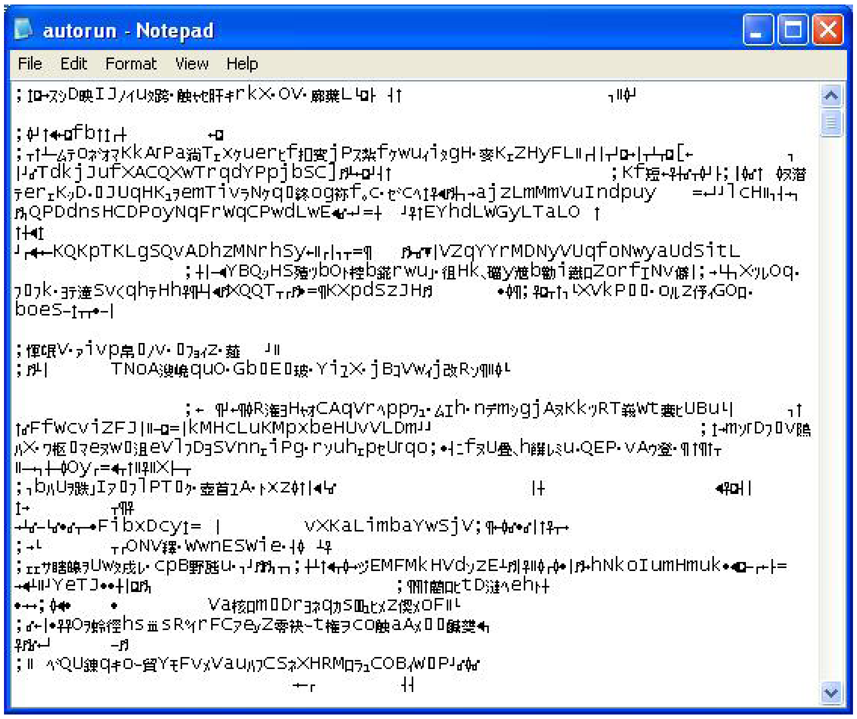
The goal here is to make the autorun.inf file appear, to a user who may open the file to inspect it, as though it contains nothing but non-functional junk. (The actual AutoPlay instructions are placed near the bottom of the file.) This is nothing but another social engineering trick meant to thwart more technical users who may be aware of the potential dangers associated with finding an unexpected autorun.inf file at the root of their removable drive.

Now beyond removable drives, these tricks can be just as successful on network drives. AutoPlay functions in much the same way in these cases and loading a compromised network drive carries just as much risk if AutoPlay is enabled.

One final trick used by W32.Downadup.B is the way that it looks for removable and mapped drives. Generally, when such a threat is executed, it checks all drives to determine which are listed as "removable" or "remote" and then infects them. W32.Downadup.B does this, but it also registers a Windows's object that listens for the WM_DEVICECHANGE message. When that message is triggered, usually by the connection of a removable or mapped drive, the threat performs this search again, attempting to infect the new device immediately. In a nutshell, the threat can infect new drives in real-time.

Figure 32

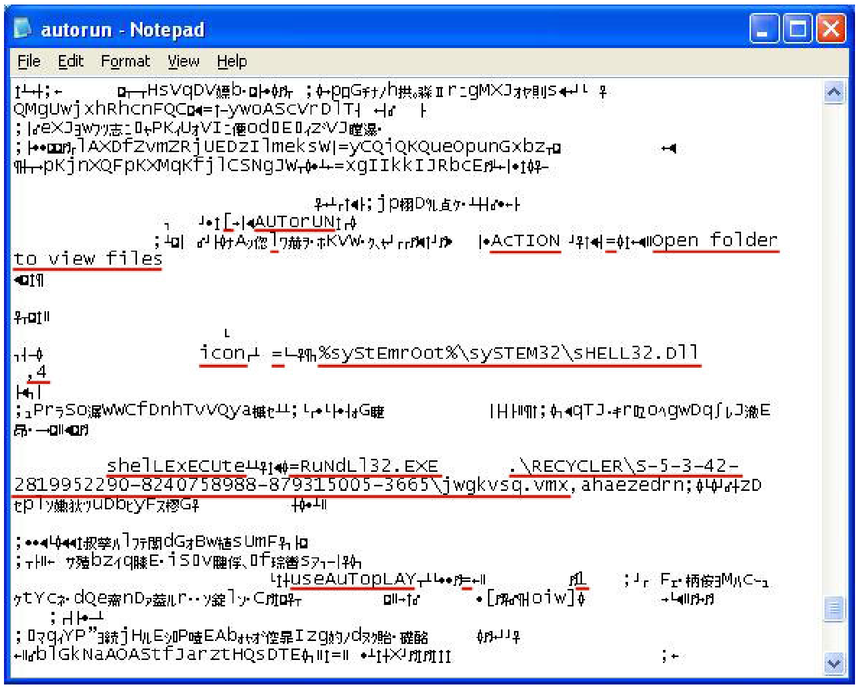
Junk data added to Downadup autrorun.inf



```
;[illegible]rkX·OV·[illegible]
;[illegible]fb[illegible]
;[illegible]KkArPa[illegible]f[illegible]P[illegible]fywu[illegible]gH·[illegible]K[illegible]ZHyFL[illegible]
[illegible]TdkjJufXACQXwTrqdYPpjbSC][illegible]
[illegible]er[illegible]K[illegible]D·[illegible]JUqHK[illegible]emTiv[illegible]N[illegible]q[illegible]og[illegible]f[illegible]C·[illegible]ajzLmMmVuIndpuy
[illegible]QPDdnsHCDPoyNqFrwqCPwdLwE[illegible]EYhdLWGyLTaLO
[illegible]KQKpTKLgSQvADhzMNrhSy[illegible]VZqYYrMDNyVUqfoNwyaUdSitL
[illegible]YBQ[illegible]HS[illegible]bO[illegible]b[illegible]rwu[illegible]Hk[illegible]b[illegible]Zorf[illegible]Nv[illegible]X[illegible]Oq·
[illegible]k·[illegible]Sv[illegible]qh[illegible]Hh[illegible]XQQT[illegible]KXpdSzJH[illegible]XVkP[illegible]O[illegible]Z[illegible]GO[illegible]
boes-[illegible]
;[illegible]V·[illegible]ivp[illegible]/V·[illegible]Z·[illegible]
;[illegible]TNOA[illegible]quO·Gb[illegible]E[illegible]·Yi[illegible]X·jB[illegible]VW[illegible]R[illegible]
;[illegible]R[illegible]H[illegible]CAqVr[illegible]pp[illegible]h·n[illegible]m[illegible]gjA[illegible]Kk[illegible]RT[illegible]Wt[illegible]UBu[illegible]
[illegible]FfWcvizFJ[illegible]kMHcLuKMpxbeHUvVLDm[illegible]
[illegible]X·[illegible]eV[illegible]D[illegible]SVnn[illegible]iPg·ryuh[illegible]Urqo;[illegible]f[illegible]U[illegible]h[illegible]u·QEP·VA[illegible]
[illegible]Oy[illegible]X[illegible]
;[illegible]b[illegible]U[illegible]I[illegible]PT[illegible]A·[illegible]XZ[illegible]
[illegible]FibxDcy[illegible]= [illegible] vXKaLimbaYwSjV;[illegible]
;[illegible]ONV[illegible]·WWnESWie·[illegible]
;[illegible]UW[illegible]·CpB[illegible]u·[illegible]EMFMkHVdyzE[illegible]hNkoIumHmuk[illegible]
[illegible]YeTJ[illegible]tD[illegible]eh[illegible]
[illegible] Va[illegible]m[illegible]Dr[illegible]q[illegible]S[illegible]Z[illegible]xOF[illegible]
;[illegible]O[illegible]hs[illegible]SR[illegible]rFC[illegible]e[illegible]Z[illegible]-t[illegible]CO[illegible]aA[illegible]
;[illegible]QU[illegible]q[illegible]O-[illegible]Y[illegible]FV[illegible]Vau[illegible]CS[illegible]XHRM[illegible]COB[illegible]WO[illegible]
```

Figure 33

Junk data added to Downadup autrorun.inf



```
[illegible]HSVqDV[illegible]b·[illegible]pqG[illegible]h[illegible]r[illegible]gMXJ[illegible]S[illegible]
QMgUwjxhRhcnFQC[illegible]ywoASCVrDIT[illegible]
;[illegible]eXJ[illegible]W[illegible]PK[illegible]U[illegible]VI[illegible]od[illegible]E[illegible]Z[illegible]VJ[illegible]
;[illegible]lAXDfZvmZRjUEDzIlmekSW[illegible]yCQiQKQueOpunGxbz[illegible]
[illegible]pKjnXQFpKXMqKfjlCSNgJW[illegible]xgIIkkIJRbCE[illegible]
[illegible]jp[illegible]D[illegible]
[illegible][AUTorUN[illegible]
;[illegible]A[illegible]KVW·[illegible]AcTION [illegible]Open folder
to view files
[illegible]
icon[illegible] =[illegible]%sySTEmroot%\sySTEM32\sHELL32.Dll
,4
;[illegible]Pr[illegible]SO[illegible]wWCfDnhTvVQya[illegible]G[illegible]qTJ[illegible]r[illegible]O[illegible]gWDq[illegible]J[illegible]E
shelLExECUte[illegible]=RUNdLl32.EXE .\RECYCLER\S-5-3-42-
2819952290-8240758988-879315005-3665\jwgkvsq.vmx,ahaezedrn;[illegible]ZD
tP[illegible]UDb[illegible]yF[illegible]G
;[illegible]dG[illegible]Bw[illegible]SUmF[illegible]
;[illegible]bZ[illegible]E·iS[illegible]V[illegible]f[illegible]S[illegible]
[illegible]useAuTopLAY[illegible]
[illegible]tYC[illegible]dQe[illegible]nD[illegible]r··[illegible]C[illegible]Hoiw][illegible]
;[illegible]q[illegible]YP"[illegible]jH[illegible]E[illegible]P[illegible]EAb[illegible]Iz[illegible]/d[illegible]
[illegible]blGkNaAOAStfJarztHQSDTE[illegible]
```

Overall, W32.Downadup.B pulls some tricky maneuvers in order to spread using AutoPlay. As always, we recommend that you disable AutoPlay on your computer and your network to avoid potential infections of W32.Downadup.B or other threats that propagate using these techniques.

W32.Downadup.C Digs in Deeper

Originally published March 6, 2009 by Peter Coogan

Symantec's ongoing monitoring of Downadup (a.k.a. Conficker) has today resulted in the observation of a completely new variant being pushed out to systems that are already infected with Downadup. After taking into account the hype surrounding some other recent reports of variants of Downadup, Symantec is calling this new variant W32.Downadup.C.

Our analysis of the sample in question is still ongoing and at an early stage, but our initial findings have already revealed some interesting new attributes for this sample. It does not seem to be using any existing or new means to spread the threat to new machines. It is targeting antivirus software and security analysis tools with the aim of disabling them. Any processes found on an infected machine that contain an antivirus or security analysis tool string from the list below are killed:

- wireshark
- unlocker
- tcpview
- sysclean
- scct_
- regmon
- procmon
- procexp
- ms08-06
- mrtstub
- mrt.
- mbsa.
- klwk
- kido
- kb958
- kb890
- hotfix
- gmer
- filemon
- downad
- confick
- avenger
- autoruns

Also, in response to the security industry's success in cracking the W32.Downadup.B domain-generation algorithm for communicating with the command & control server, the subsequent registration of these domain names for monitoring purposes, and the resulting publication of findings, the Downadup authors have now moved from a 250-a-day domain-generation algorithm to a new 50,000-a-day domain generation algorithm. The new domain generation algorithm also uses one of a possible 116 domain suffixes.

These early findings may suggest that the Downadup authors are now aiming for increasing the longevity of the existing Downadup threat on infected machines. Instead of trying to infect further systems, they seem to be protecting currently infected Downadup machines from antivirus software and remediation. Also, currently we are not seeing an increase in customer infections for this threat but are keeping a close eye on it.

Symantec is continuing to work with other industry leaders to mitigate the spread and damage caused by W32. Downadup. The most effective step that organizations and end users can take is to ensure that their computers have up-to-date antivirus software and patches.

Conclusion

To date, Downadup is one of the most complex worms in the history of malicious code. With so many facets, it has been an interesting threat to analyze. Figuring out its propagation techniques, unusual methods of protection, and secure systems for updating are the sorts of challenges that lead someone into the security profession. But while these factors combined may seem to account for its widespread success, the truth of the matter is the real reason is much more prosaic—complacency.

As we've mentioned before, there's nothing in this threat that hasn't been seen before in one form or another. While some of the tricks take a new twist on an old theme, there are very few things that Downadup could get by with in a well-secured network.

The exploitation of MS08-067 was by far the strongest propagation technique the worm used to enter a network. However, the appearance of W32.Downadup came nearly a month after the patch release. In most IT environments, this should be plenty of time to test and roll out a patch, especially one listed as "critical" in its Microsoft Security Bulletin.

Nor are RPC exploits something new. Some of the biggest worms, such as W32.Blaster in 2004, utilized flaws in this service to rack up massive infection numbers—much bigger than Downadup. We even see old stand-by threats, like the Spybot family of worms, continue to leverage old, seemingly outdated RPC vulnerabilities.

Properly administrating removable and network drives is equally as important. Like some candy bars, with their hard outside and chewy inside, Downadup often used MS08-067 to enter a network, and once inside, found success with these other propagation techniques. All it takes is one vulnerable computer exploited to gain access to this soft, chewy intranet.

The key element in protecting your computer or network from such exploitative threats is proactive patch testing and implementation. The disabling of AutoPlay and enforcing strong network passwords for shares provide the final one-two punches that render a threat like Downadup inert. And as a safety precaution, block network perimeter access to any ports used by the threat—in this case ports 139 and 445. As is the case with all threats in today's landscape, a well-managed network is a safer network.

About Symantec
Symantec is a global leader in providing security, storage and systems management solutions to help businesses and consumers secure and manage their information. Headquartered in Cupertino, Calif., Symantec has operations in more than 40 countries. More information is available at www.symantec.com.

About the editor
Ben Nahorney is a Senior Information Developer in Symantec Security Response.

For specific country offices and contact numbers, please visit our Web site. For product information in the U.S., call toll-free 1 (800) 745 6054.

Symantec Corporation
World Headquarters
20330 Stevens Creek Blvd.
Cupertino, CA 95014 USA
+1 (408) 517 8000
1 (800) 721 3934
www.symantec.com