DropSmack: How cloud synchronization services render your corporate firewall worthless

Jake Williams
CSRgroup Computer Security Consultants
jwilliams@csr-group.com

BlackHat Europe 2013

Abstract:

Cloud backup solutions, such as Dropbox, provide a convenient way for users to synchronize files between user devices. These services are particularly attractive to users, who always want the most current version of critical files in each location. Many of these applications "install" into the user's profile directory and the synchronization processes are placed in the user's registry hive (HKCU). Users without administrative privileges can use these applications without so much as popping a UAC dialog. This freedom makes illicit installations of these applications all the more likely.

Cloud backup providers are marketing directly to corporate executives offering services that will "increase employee productivity" or "provide virtual teaming opportunities." Offers such as these make it more likely than ever that any given corporate environment has some cloud backup solutions installed.

Some theoretical research papers have previously identified the possible risk that cloud backup solutions may pose for data exfiltration. These applications pose serious risks for Data Loss Prevention (DLP) applications since normal channels monitored by DLP are bypassed. It is far more difficult for DLP to detect files written to the user profile than files being attached in the browser to a web based email or files being moved to a removable drive.

The contributions of this presentation are threefold. First, we show how cloud based synchronization solutions in general, and Dropbox in particular, can be used as a vector for delivering malware to an internal network. We do this by examining a case study from a penetration test. Second, we show how specially developed malware can use the synchronization service as a Command and Control (C2) channel. Given an active C2 channel via Dropbox, an attacker can determine how to establish a more traditional C2 channel out of a network on the compromised host. Finally, we demonstrate functioning malware that uses Dropbox to exfiltrate data en-masse from the network. While the idea of using cloud synchronization technologies for data exfiltration is not new, we are not aware of any functioning tools designed to exfiltrate data from a network via Dropbox.

In our experience, people tend to take potential vulnerabilities more seriously *after* proof of concept code is publicly available. By releasing this tool, we hope to stir up some real conversation about whether synchronization software is appropriate for all corporate environments (and if so, under what controls).

Giant Honking Disclaimer:

We **<u>don't</u>** think Dropbox is insecure. Don't sue us. We're not defaming you. The fact is that it provides an awesome service that just happens to offer a C2 and exfiltration channel ***by design***.

To our knowledge, installing Dropbox on your machine doesn't introduce any new vulnerabilities. But like **<u>any</u>** software you install, it increases the attack surface. Dropbox is no exception (sorry, no free pass here). Any software you install that has established network communication channels doubles down on this surface, making it more attractive to an intruder.

We could do the same thing any web based email service. The problem with that approach though is that we don't know ahead of time which mail services and browsers the user is allowed to use. If we pick wrong, we lose (FAIL). Dropbox on the other hand is pretty obvious. If it's installed, it's allowed through the firewall (WIN).

Finally, we would like to give mad props to Dropbox. In our investigations into client side security of cloud synchronization services, **nothing else comes close**. Most services implement security that is **just embarrassing**. My (black) hat is off to the Dropbox team for making my job that much harder.

History:

Cloud synchronization clients have been plagued with poor design choices almost since their inception. Possibly the best publicized vulnerability was the discovery that the authentication mechanism used by Dropbox was broken by design. The implementation allowed anyone with access to the user's Dropbox database files (in their profile directory) to authenticate as that user from an arbitrary location. In other words, Dropbox did not key the databases on a per-device basis. While this vulnerability requires an attacker to have access to a Dropbox user's Windows profile directory, this doesn't diminish the attack's implications. In a case where an attacker had compromised a server storing roaming profiles for users, they could use these databases to access any data that had been synchronized to Dropbox from a remote location. This would allow wholesale 'hoovering' of data from a corporate network without accompanying telltale bandwidth spikes.

Dropbox has since corrected this fault in its authentication mechanisms and encrypts its client side configuration databases. Some reverse engineering work was recently (2012) performed on the Dropbox code by Ruff and Ledoux. They discovered that Dropbox uses a well known SQLite encryption library that can be readily decrypted with the appropriate information. Dropbox is likely to change its implementation again in light of this revelation. However, users must understand that if they don't have to provide a password to log into Dropbox, a suitably funded attacker with access to their files doesn't either.

Dropbox and other cloud synchronization software have been studied by forensic professionals as well. After forensic practitioners openly discussed the format of Dropbox databases, tools were built to automatically parse information from the databases. This allowed forensic professionals to discover hashes and filenames of files that were synchronized to the cloud, as well as a timestamp of when the activity was performed. Given that most users believe a file (and information about the file) is gone when it is deleted from their machine, this represents a data disclosure. When presented with the news that their profile directory was compromised, most average users would not include information about data long ago synchronized to the cloud in their mental loss inventory. Dropbox again partially addressed this issue by encrypting the databases on the host. The same problems highlighted above in the authentication section continue to persist here (e.g. well-funded attackers may reverse engineer the encryption scheme and access the data).

Note that while it seems we may be attacking Dropbox exclusively, several other cloud synchronization providers have not done as much as Dropbox to secure data stored on the end-host. Many cloud synchronization providers continue to store client configuration information in plain-text databases stored on the end-host. Several do not key the database to the specific end-host, enabling the trivial 'copy the databases to authenticate from anywhere' attack.

Dropbox has experienced other problems with security. In one case, an internal software upgrade left the service without any authentication for up to four hours (or less depending on who you trust). Of course Dropbox has reviewed internal policies to ensure this sort of thing never happens again. We sincerely believe that. Or at least we believe that they believe that (say that five times fast). But because we're pretty sure Dropbox employs humans, we think someone will screw up again at some point in the future. Whether the error is discovered first internally, by a black hat, or by a white hat is anybody's guess.

Another gem in the history of Dropbox came when it was discovered that Dropbox mobile was sending files metadata in the clear. So the files were encrypted, but the metadata (including file name and hash) were not. Yeah, that could sting. This was discovered after the issue with non-keyed (and unencrypted) authentication databases, so Dropbox doesn't get a pass here.

Problem:

Practically everywhere we go on penetration testing and incident response engagements, we see the Dropbox LanSync protocol in packet captures. This leads us to the conclusion that cloud synchronization program installations are nearly ubiquitous. Whether these installations are approved by IT staff is another question entirely.

Many of these applications "install" into the user's profile directory and the synchronization processes are placed in the user's registry hive (HKCU). Users without administrative privileges can use these applications without so much as popping a UAC dialog. This freedom makes illicit installations of these applications all the more likely.

No matter how good the security in a network is, we usually find that the network rather resembles a piece of candy. It has a hard outer shell and a soft-gooey inside. Once a foothold is established in the network, game on. Security personnel should strive to eliminate programs that increase the attack surface. For instance, an infected PDF file might be caught by a spam filter if emailed to a victim. However, the same file can be transferred onto the victim's machine through Dropbox, completely bypassing network defenses.

In the case study below, we'll illustrate just how easily a single installation of Dropbox is more than enough to compromise the security of the entire network. That firewall you bought: worthless. Your DLP solution: that was worthless before we came along. Your IDS: you guessed it, it's not going to stop this attack. The best you can hope for is whitelisting software on the end-host (and as we all know, even that can be defeated).

Case Study:

Scenario:

The client (a large defense industrial firm) contacts us and says they want a **real** black box penetration test. They have an in house security assessment team, but they want to know what a no holds barred attack by a persistent adversary would look like. Really, they want to know their exposure to this type of attack.

Start with standard methods:

We started the same way every penetration test starts – recon.
We looked for vulnerable web servers and inventoried publicly available information on C-level employees and IT admins.

Sure, you can go after other employees (thousands of them) but the IT admins usually get you access. C-level employees get you the golden nuggets that are so interesting. We'll stick with those.

Social Engineering:

We try to social engineer some employees, but no luck. This company has quite an aggressive security policy. We find out that they are running some type of McAfee antivirus before the employee becomes suspicious. Over the next couple of days, we get no love from social engineering calls. One employee kindly tells us he knows what is going on because he got a company-wide email that the company is under attack from an active social engineering campaign. That's useful information, but it means that further social engineering efforts will likely be unsuccessful. I hate it when that happens…

Spamming:

Next up: start spamming. We figure we can leverage the company wide social engineering email to offer some links containing "training" on how to avoid social engineering attacks. We get several hits on our web server, and even get a couple of unpatched browsers, but we never get any command and control on the victim machines. We try sending infected documents containing the old standbys –

Confidential: Workforce reduction plan 2012Q3
Attention: Health insurance benefits change – Action Required

Between these two, we usually get good coverage. No go. I know people are opening these documents (they always do). We have a good mix of PDF with Javascript (and patched vulnerabilities) and Excel spreadsheets with macros. We should be getting callbacks from the first stage payload, but we don't have a single hit.

Social Network Analysis:

We've run into this before. The company security is great, they have proxying firewalls with protocol inspection, and maybe they are filtering mail at the gateway. In the past, we've been successful owning an employee's private computer (away from the corporate network defenses) and then getting into the corporate network over the employee provided VPN. Once inside the defenses, it's usually fairly easy to figure a way out (since we are no longer flying blind). You don't want to rely on a single VPN connection in if you can help it (since it won't always be connected).

We try to get close to C-level types and managers on LinkedIn and to a lesser degree Facebook who are likely to take work home (and need a VPN connection). Google searches help locate some email addresses too. We even use some of the awesome tools from Black Hills Infosec (such as recon-ng) to find links to our execs. Our most promising lead comes when we identify the personal email address of the CIO, who is active with the PTA for his kid's school. Nothing like leveraging a kid to gain access…

Social Engineering (redux):

We don't send the CIO an attachment right away. First, we just work him over a little talking to him about PTA fundraising options where we are offering nearly unbelievable returns (that's right buddy, they were too good to be true). We offer to send him some data detailing the opportunities. We send an .xls file with return calculators that don't work without macros enabled. Essentially, the user is forced to be owned to even read our document…

Great, we are on the work-from-home computer of the CIO – this took us almost two weeks. Now to get that VPN connected and bridge over to the corporate network. Let's see, Cisco AnyConnect – nope, OpenVPN – nope, where the f%$# is the VPN software???? You don't get to be a CIO at a Fortune 500 without working seven days a week. We do a quick survey of the software on the machine and don't find any VPN software. We do find company documents spread out across a number of folders. Maybe he is moving files via USB drive? He can't be stupid enough to send them over his Gmail account.

Confidential Files, a good start:

Then it hits us: he has Dropbox installed. We check the Dropbox folder and find that many of the company documents we found are being sync'd to his computer via Dropbox. Pay dirt! Or is it… Sure we can steal some work documents from this guy, but so far this isn't exactly the "golden nugget" we like to put in the final report. I want to own some internal servers and really show the company how they can be hurt.

We've read some research on the potential to use Dropbox and the like to exfiltrate data from a corporate network, but we've never actually seen it done. To make matters worse, we aren't actually in the network yet. We still need to get in there. First, we grab all the Dropbox databases. Some people have had success reading these in the past and using them for various purposes. Damn. Dropbox started encrypting them. This could be a serious reverse engineering effort and "ain't nobody got time for that" (not on this engagement at least). We grab all the confidential corporate documents we can find, install a beaconing implant on the CIO's home machine to maintain access, and retreat for a six pack and some brainstorming.

What we have so far:

A way to send files over Dropbox to any device the CIO uses.

What we want:

A running implant (with command and control) in the corporate network.

Brainstorming (and beer):

With a blood alcohol level of .12 and a small dose of inspiration, it hits me. We can write some custom software that communicates over Dropbox and use that as a C2 channel. We already have evidence that Dropbox is being used to send confidential corporate documents. It stands to reason that it is installed on the internal corporate network. Let's work from that assumption. We place files on the CIO's laptop (via our malware) and they will automagically end up on his corporate machine. We don't need to know anything about the corporate firewall, proxies, or anything else. He's already configured our malware delivery channel to talk through all of that. Bonus: everything we send is encrypted to the desktop, so no IDS will have a chance to inspect it.

So we can deliver an executable to the CIO at work. Great. But nothing we sent before was able to call out of the network. Clearly there's some crazy network filtering going on that we just haven't yet accounted for. Just delivering a standalone Meterpreter isn't going to get us far…

DropSmack FTW:

What if we built some new malware to receive tasking and send exfil over the Dropbox file sharing service? Yeah, that sounds like a plan. We'll call it DropSmack. We code up a very quick implant that supports a few basic commands (PUT, GET, DELETE, and EXECUTE). We considered adding more commands (and may in the future), but that subset of commands gets you everywhere you need to go. Everything else is just gravy.

We embed DropSmack in one of the recently modified confidential spreadsheets that we've retrieved from the laptop and add some new macro goodness. When we regain access to the CIO's laptop, we place the spreadsheet in his Dropbox folder. It automagically synchronizes to the cloud, and shortly thereafter down to his corporate machine.

We could wait for the CIO to open the file at work, but "ain't nobody got time for that." Instead, we socially engineer him to do so with a strategically placed phone call requesting information we already know to be in the spreadsheet. The next time we see the laptop on, we place a tasking file in the Dropbox folder. DropSmack gets the tasking (a set of commands to survey the machine) and within 5 minutes, we have our data back. DropSmack writes the output data back to a file in the Dropbox folder on the corporate machine where it automatically synchronizes back to cloud (and the laptop).

From here, the sky is the limit. We have bi-directional command and control over an implant running in the corporate network. Ideally, we don't want to use this forever since it is slow and kludgy. However, we can map the network and most importantly determine the settings that will allow us to get a quicker connection to the corporate network.

Note again that we no longer care about the corporate firewall and barely care about the IDS. It's a heck of a lot easier to figure out how to get out of a network than to get in.

Detection:

Detection is a real problem. All of the workstation class cloud backup solutions we've investigated transfer files of HTTPS. We think that's swell. If they didn't use HTTPS, that would open up a whole host of new problems. Other encrypted protocols would likely require new firewall rules. Experience tells us that new firewall rules == increased attack surface == increased likelihood that someone misconfigures something (Win).

In the case that a client applications don't use an encrypted protocol (like HTTP), your files can be read while syncing too and from the cloud. Of course this is bad for a whole number of reasons. Of course, that hasn't stopped some really low-density cloud providers from doing it in the past.

Dropbox uses Amazon S3 for back-end storage (as do many other cloud backup solutions). One detection scheme involves blacklisting (or at least alerting) on communications with S3 addresses. As more and more applications migrate to Amazon EC2 and S3, we expect this approach to become increasingly fragile. There will simply be too many alerts to detect actual cloud backup usage.

A related, but more effective detection strategy is to implement monitoring at the corporate DNS server. If DNS traffic is allowed past the perimeter firewall from machines other than the DNS server, packet inspection at the perimeter may be required. In either case, the goal is to detect the DNS requests made to the specific cloud provider's servers for authentication purposes.

Some cloud based providers, such as Dropbox, use proprietary protocols to synchronize data between clients on the LAN. These protocols can be easily detected by auditing network traffic and identifying those hosts involved. The cloud synchronization providers who utilize these protocols usually have them enabled by default. There is usually an option for the end user to disable these features however, so the lack of protocol traffic does not necessarily mean that no synchronization programs are in use.

The LanSync protocol used by Dropbox communicates on TCP and UDP ports 17500. The client opens these ports on machines where the software is installed. Port scanning may be effective in identifying machines with LanSync listeners enabled. Host based firewalls may limit the effectiveness of this approach however. As discussed previously, non-administrative personnel can typically install these synchronization clients. However, they may lack the privileges to update the firewall configuration. In other words, the client may listen on a port but be blocked by a host-based firewall. This approach would render the port scanning approach ineffective.

The most effective way we've found to detect insiders running unauthorized cloud synchronization software is application whitelisting. That helps a lot. Another possible approach is to scan user profiles for related files since most synchronization clients install there. Roaming user profiles help a lot with this since profile information is synchronized back to a central server in the office. Roaming profiles introduce new risks however, so you need to think hard before implementing them (if you haven't already accepted the inherent risk).

Future Work:

This work is still in its infancy (but growing every day). We're picking on Dropbox because it is the clear market leader, but we've examined other cloud synchronization services and Dropbox is the clear leader when it comes to security. Anything that can be done to Dropbox can be done to other services with near impunity. Note to service providers: encrypt your client side databases. It doesn't stop a reverse engineer, but it does at least slow us down.

The fact is that for this to be useful, one of a user's devices has to already be compromised (or we have to be able to log into the user's account at the web service). This obviously isn't a zero-day vulnerability (though we're sure the media will portray it that way the day after Blackhat).

Right now, we don't try to read the Dropbox client databases. We could in previous versions, but Dropbox has upped their game and changed the way they bundle their code. We anticipate that by the time we present this, we'll be reading them again (work schedules permitting). We'd like to be able to cut out the laptop in the CIO scenario, and reading the login credentials from the client side databases will allow that.

Another problem is that we currently assume that the Dropbox folder is in the default location. If we deliver DropSmack via a compromised document and the user has disabled the default sync folder, we won't have command and control. We have two plans to overcome this. First, we're planning to read the databases to gain configuration information (yeah, those again). Second, we've noted that explorer.exe changes the icons for Dropbox synchronized files. We're considering exploiting that to locate synchronization locations. We like the second approach better since it's a feature Dropbox and unlikely to change.

Dropbox likes to issue a popup notification when files are updated, new files are added, or files are deleted from the synchronization folder remotely. This isn't very good from an operational security standpoint. We don't really want the user noticing every time one of these files updates (since we send new tasking files and receive exfiltration files). We should turn this off, but again, we need database access. To steal a technique from other malware, we may opt to listen for the window notification and dismiss the window before the user can see it.

Finally, we really should support other synchronization services. These would make way cooler demo's since we can read (and modify) their configurations at will. But they just aren't the market leaders and somehow that makes it feel like cheating. Box.net and SpiderOak are on the short list for future support.

DropSmack Usage:

Folder locations:

DropSmack looks in the user's profile directory for a Dropbox folder. If it doesn't find the Dropbox folder there, it looks for a Dropbox folder under the My Documents folder. If it doesn't find a Dropbox folder in either location, it exits. The first location is the default location for the Dropbox folder in Windows Vista/7 and the second location is the default on Windows XP installations.

If DropSmack finds the default Dropbox synch directory, it copies itself to %APPDATA%\DropSynch.exe. It then creates an autostart entry in the user's Run key (in HKCU). Not only is this good for not popping a UAC dialog on the user's machine, but you also benefit from executing wherever the user's profile is loaded. This may get you execution on several different machines in an enterprise environment.

This presents a potential problem when it comes to tasking. After verifying that a file exists (perhaps by running a dir command), you might issue a GET command. However, the GET command might be executed on a different machine where the file doesn't exist. In a future version of DropSmack, we will implement a system of host IDs so that commands are guaranteed to run only on the desired host or hosts.

Tasking files:

By default DropSmack looks in the Dropbox folder for files starting with the string 'DROPSMACK_*'. This is far from subtle and you should change it before deploying it. When files are found, commands are parsed by line. Each line is checked for a verb at the beginning and one or two arguments (depending on the command). This means you could send a .doc or a .dat file that a normal user won't inspect the contents of. The file header can even be set so tools checking the file signature will incorrectly identify the file. Yeah, we're evil like that. DropSmack will happily ignore any line that doesn't start with a valid verb (listed below). It will also ignore any line that contains the incorrect number of arguments for that verb.

Logging:

As currently implemented, DropSmack is just fire and forget. You don't get any confirmation back as to whether or not the commands executed. We implemented a version that wrote status messages back to the synchronization folder, but it was super noisy and we didn't like it. So we changed it.

Command Verbs:
PUT <local>|<remote>
The 'PUT' command takes the filename specified in the 'local' argument and moves it to the filename specified by the 'remote' argument. The 'local' filename should be placed in the Dropbox directory before the tasking file. The 'local' filename should not contain a path, as DropSmack knows where to locate it (the synch folder). The 'remote' filename should have a path starting with a drive letter. The file is moved, rather than copied so you don't need to do any additional clean-up.

GET <remote>|<local>
The 'GET' command takes the filename specified in the 'remote' argument and copies it to the 'local' argument. The 'remote' file requires a complete path while the 'local' file requires only a filename (no path). The file will be placed in the Dropbox synchronization folder with the 'local' name where it is synchronized to the cloud. It is your responsibility to collect and delete the file at your leisure.

MOVE <source>|<dest>
The 'MOVE' verb moves a file already on the target machine to another location on the target machine. I found myself wanting to use move a fair amount in some work I was doing and grew tired of spawning command prompts to do it. Both fi

DEL <filename>
The 'DEL' command deletes the file specified in filename. The filename should be a complete path.

EXEC <command>
The 'EXEC' verb executes a command on the remote machine. Note that the command is executed in the context of the user's account. In an enterprise environment this account probably doesn't have administrative privileges (if it did, you probably wouldn't need DropSmack to get in). This means that if you try to do something that requires admin privileges, you will pop UAC. Yeah, that isn't subtle. Don't do that.

SLEEP <milliseconds>
The 'SLEEP' verb causes DropSmack to sleep for the number of milliseconds in the first argument. This is normally used when some number of commands are EXEC'd and you want to exfil the output of those commands using the same tasking file. Just place a SLEEP in the tasking file after the EXEC command and before the GET command for the output file.

About the author:

Jake Williams, a principal consultant at CSRgroup Computer Security Consultants, has over a decade of experience in secure network design, penetration testing, incident response, forensics, and malware reverse engineering. Prior to joining CSRgroup, he worked with various government agencies in information security roles.

Jake has twice won the annual DC3 Digital Forensics Challenge and has spoken at several regional ISSA meetings, Shmoocon, and the DC3 Conference, as well as numerous US government conferences.

Jake is currently pursuing a PhD in Computer Science where he is researching new techniques for botnet detection. His research interests include protocol analysis, binary analysis, malware RE methods, subverting the security of cloud technologies, and methods for identifying malware Command and Control (C2) techniques.

Bibliography:

Mulazzani, M., Schrittwieser, S., Leithner, M., Huber, M., & Weippl, E. (2011). Dark clouds on the horizon: Using cloud storage as attack vector and online slack space. Proceedings of the 20th Annual USENIX Security Symposium.

Ruff, N. & Ledoux, F. (2012). A Critical Analysis of Dropbox Software Security. Proceedings of 2012 Hack.lu Security Conference.

Newton, D. (2011). Dropbox authentication: insecure by design. Retrieved from: http://dereknewton.com/2011/04/dropbox-authentication-static-host-ids/

Newton, D. (2011). Forensic artifacts: Dropbox. Retrieved from: http://dereknewton.com/2011/04/forensic-artifacts-dropbox/

McClain, Frank (2011). Digital forensics: Dropbox. Retrieved from: http://computer-forensics.sans.org/blog/2011/06/17/digital-forensics-rain-drop-keeps-falling-on-my-box

Ferdowsi, A. (2011). Yesterday's authentication bug. Retrieved from: https://blog.dropbox.com/2011/06/yesterdays-authentication-bug/

Unknown. (2011). New security issue at Dropbox. Retrieved from: http://pastebin.com/yBKwDY6T

Cardwell, M. (2011) Dropbox Mobile: Less secure than Dropbox Desktop. Retrieved from: https://grepular.com/Dropbox_Mobile_Less_Secure_Than_Dropbox_Desktop