

# Post Exploitation Operations with Cloud Synchronization Services

Jake Williams  
CSRgroup Computer Security Consultants  
[jwilliams@csr-group.com](mailto:jwilliams@csr-group.com)

BlackHat USA 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 on every device. 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.

We released the DropSmack tool at Blackhat EU. This showed enterprise defenders the risks posed by cloud synchronization software and gave pen testers a new toy to play with (you can bet that pen testers weren’t the only ones who noticed). In response to feedback from the original presentation, DropSmack has been improved to deal with some of the unique operational challenges posed by synchronization environments. In particular, we added the ability to work with more synchronization services automatically.

In this talk, we’ll demonstrate how DropSmack v2 works and explain how to deploy it in an operational environment. We’ll look at some of the countermeasures to these attacks, including the encryption of synchronized files by third party software. Additionally, we’ll investigate the potential of using so-called “next generation firewalls” to defeat DropSmack.

You’ll also learn about the issues of credential storage in the context of cloud synchronization services. Several synchronization applications also use insecure authentication methods. We’ll highlight these applications so you know what works, what doesn’t, and what you should run (not walk) away from. You’ll learn about post-exploitation activities you can accomplish when your freshly compromised target is running a cloud synchronization product.

Finally, we’ll demonstrate the steps you need to follow to steal credentials for the products that store them. Why would you want to steal stored credentials for a cloud synchronization service you ask? After all, any files that have been synchronized to the cloud must already on the machine you just compromised, right? Not necessarily. You’ll learn a variety of nasty things you can do with the cloud synchronization service portals that you may never have considered.

If you’re a network defender, you’ll leave this talk with a new appreciation of the risks posed by cloud synchronization services (and a nauseous feeling if you have them in

your environment). If you are a penetration tester, you'll leave with a new bag of tricks. Either way, a fun time is sure to be had by all.

## **Giant Honking Disclaimer:**

We don't think Dropbox, SkyDrive, Box.net, or any other vendor that DropSmack operates against is insecure. Don't sue us. We're not defaming you. The fact is that you provide an awesome service that just happens to offer a C2 and exfiltration channel *by design*. You are no more or less insecure by installing one of these services in your network than you were installing an email server in the early 90's. The difference is that nowadays, every email product on the planet is being monitored by antivirus or other security software. File synchronization services? Not so much. Our hope is that by highlighting these issues, we can help fix that.

To our knowledge, installing Dropbox or any other cloud synchronization provider on your machine doesn't introduce any new vulnerability. But like any software you install, it increases the attack surface. Cloud synch providers are 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).

## **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.

The DropSmack tool was released at Blackhat Europe in early 2013. Since then we've entertained numerous requests for demonstrations of the tool as well as impact assessments on the use of cloud synchronization services in enterprise networks. Some people have mistakenly assumed that because they were not using Dropbox, they were immune to the problem. This couldn't be farther from the truth. To prove this, we've updated DropSmack to use other services.

[As a note to readers, this problem statement and case study are taken from the original DropSmack presentation. Given the larger audience of Blackhat USA, we opted to quickly review what DropSmack is and what problem it solves rather than assume everyone has seen the original presentation. Also, the Blackhat EU archives have copies of the white paper in PDF format and downloading a PDF from blackhat.com just seems kind of wrong somehow ☺]

## **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 automatically 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.

### **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.

### **New Commands**

POLL <milliseconds> Change the polling interval. The polling interval used to be defined at compile time. It controls how often DropSmack checks for and processes tasking files. There may be times when you want this to be especially long (to avoid consuming resources). In other cases, you may want it to be fast. It was easy enough to fix, so we fixed it. You can now dynamically update the polling interval. Note that when the process restarts (usually at reboot), the polling interval resets to the value DropSmack was originally compiled with.

### **DropSmack Updates**

We realized that DropSmack needed to operate on more than one synchronization program. To that end, we updated DropSmack to work with several synchronization programs in their default locations, including:

- Dropbox (of course) - %USERPROFILE%\Dropbox
- Box - %USERPROFILE%\Documents\My Box Files
- SugarSync - %USERPROFILE%\Documents\My SugarSync
- SpiderOak - %USERPROFILE%\Documents\SpiderOak Hive
- Amazon Cloud Drive - %USERPROFILE%\Cloud Drive\Documents
- Google Drive - %USERPROFILE%\Documents\Google Drive
- SkyDrive - %USERPROFILE%\SkyDrive
- JustCloud (Just because we can) - %USERPROFILE%\SyncFolder

### **Future Work:**

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.

Another issue for future work is better handling of tasking files when we're sending to a box with multiple synchronization services. Currently the support here is hack-ish at best. We can send a tasking file to all services and then sleep, but otherwise we may re-process the tasking file (ending in a resource intensive loop). A related problem is when we only want to process a task on one machine connected to a service synchronizing to many machines. Both of these can problems can be resolved assigning some sort of address to each DropSmack executable. Each DropSmack executable would then only process the tasking files with an identical address.

## **LAN Sync**

The basic functionality behind LAN sync is to allow only one machine on a LAN to download an updated copy of a file. From there, all other machines on the same LAN will retrieve the file from the machine on the local LAN. This saves Internet bandwidth, which may be considerable if many machines on the same LAN need to request copies of the same file.

### **Attack Surface**

LAN sync is an additional attack surface that can be exploited in two primary ways. First, they may open ports on the firewall that you may not have access to open as a regular user. In these cases, an attacker might stop the process listening on the LAN sync port and use that for his own communications. Such an attack would help the attacker blend in if the target network were monitoring communications. They would likely already have rules in place to ignore any communications using the LAN sync ports.

The second use case for LAN sync involves attacking the synchronization service itself. We have not spent significant time reversing the LAN sync protocols or fuzzing the services, but these seem to be areas that are ripe for exploitation. Microsoft has had numerous exploitable conditions in SMB (MS 08-067, MS 10-020, MS 11-020, etc.) over the years. Microsoft doubtlessly had a larger development team with more eyes on the vulnerable code than the LAN sync providers. This makes it almost certain that there are vulnerabilities in these products that are yet to be discovered.

### **Dropbox LAN sync**

The Dropbox LAN sync protocol requires access to TCP port 17500. According to Dropbox documentation (<https://www.dropbox.com/help/137/en>), machines using LAN sync must be connected to the Internet as they receive syncing orders from the Dropbox servers. Dropbox is very noisy in its implementation of LAN sync, regularly broadcasting to the LAN to notify potential clients of its presence.

### **SpiderOak LAN sync**

The SpiderOak client listens on an RPC port for LAN sync. The RPC port is dynamically assigned but could be programmatically discovered by contacting the RPC port mapper service (TCP port 135). The SpiderOak LAN sync service also broadcasts messages to the network. It uses a UDP source port of 21327 and broadcasts to UDP destination ports 21327 and 21328 every 30 seconds.

### **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 doesn't use an encrypted protocol, your files could be read while syncing to 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.

### **Are NGFWs the answer?**

Since the first presentation at Blackhat EU, we've been told that next-gen firewalls (NGFW) will prevent DropSmack from operating. Intrigued by this possibility, we examined what specifically an NGFW would bring to the table. One of the many touted benefits of an NGFW is the possibility of SSL proxy and inspection. We think this may cause as many problems as it solves, but is still worthy of consideration.

We do concede that NGFW deployments are useful in restricting illicit deployments of cloud synchronization programs. Because they can inspect the contents of an SSL stream, NGFWs are able to differentiate between SSL connections destined for Content Distribution Networks (CDN) and online storage, both of which are often stored in Amazon S3. This allows the network operator easily determine which applications are communicating on the network and block specific applications as they see fit.

However, an NGFW solution falls short in two specific cases. First, we do not see how they can stop DropSmack from communicating. DropSmack communicates by placing commands inside files that are automatically synchronized. We fail to see how an NGFW would detect which files contained commands and block only those file. Of course, NGFW operators could create signatures on the list of DropSmack commands. However, because the source of DropSmack is freely available, the commands could be easily changed, yielding this approach impotent.

The second shortcoming of an NGFW for this solution involves detecting exfiltration. It is possible that an NGFW might detect an encrypted RAR file being synchronized and flag on that. However, we don't think this is a sure thing. Even still, how will an NGFW determine which normal office documents a user meant to synchronize to the cloud and which ones are being exfiltrated by the attacker? Organizations could couple an NGFW with a DLP system to achieve some greater fidelity in this area, though we have never observed such a deployment in our consulting practice (and don't get me started on the weaknesses of many DLP systems).

### **Whitelisting**










This isn't at all cloud synchronization specific. However, the most effective way we've found to detect insiders running unauthorized cloud synchronization software (or any software for that matter) is application whitelisting. It 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).

## Insecure Authentication

It seems insane, but some synchronization services actually implement very insecure authentication. While searching for SkyDrive market share or some such on Google, we came across "news for SkyDrive." This led us to a "news" site where there was a supposed news article in the "Community Events" section of an online newspaper. Here, we were told that we should drop Dropbox, Google Drive, and SkyDrive in favor of another provider. The provider's name? JustCloud.

We'd never heard of JustCloud, but a quick search turned up several other faux news items about how awesome they are. We even found that they are rated #1 at [www.thetop10bestonlinebackup.com](http://www.thetop10bestonlinebackup.com).

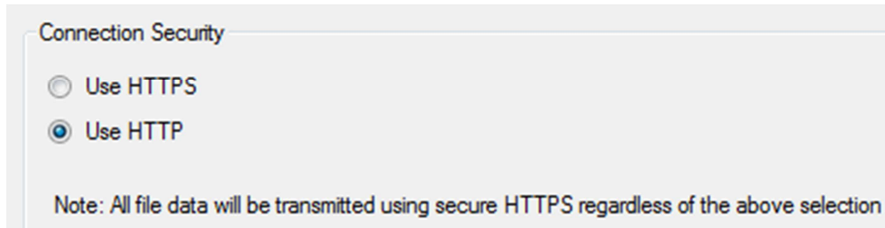
Trend	Company	Price	Storage	Score	Review
↑		Free (Limited Time)	Unlimited	98% Rate	<a href="#">Read Review</a> <input type="checkbox"/> Compare
→		\$4.95	250GB	97% Rate	<a href="#">Read Review</a> <input type="checkbox"/> Compare
↑		\$2.95	Unlimited	96% Rate	<a href="#">Read Review</a> <input type="checkbox"/> Compare
↓		\$6.66	50GB	93% Rate	<a href="#">Read Review</a> <input type="checkbox"/> Compare
→		\$9.99	60GB	92% Rate	<a href="#">Read Review</a> <input type="checkbox"/> Compare
→		\$7.99	125GB	91% Rate	<a href="#">Read Review</a> <input type="checkbox"/> Compare
→		\$4.95	250GB	91% Rate	<a href="#">Read Review</a> <input type="checkbox"/> Compare
↓		\$9.99	50GB	91% Rate	<a href="#">Read Review</a> <input type="checkbox"/> Compare
→		\$9.99	25GB	90% Rate	<a href="#">Read Review</a> <input type="checkbox"/> Compare

"So there you have it" I thought. My search for cloud synchronization providers is over. Like any self-respecting security researcher, I registered for an account with a throwaway email address and started researching.

While installing the application, I fired up wireshark and saw something I hadn't seen investigating any other cloud synchronization provider: HTTP. I know what you're thinking. It must be something like grabbing an image, or a stock file, or something. This can't possibly be authentication (or worse, file transfer). Well, not to disappoint, the folks at JustCloud were fine with authenticating over HTTP. Just look at the POST contents below:

```
credentials=
{"token":"[REDACTED]bcd9e81e]d77e6ed3_gjnokbajb515d
89b2[REDACTED]4","secret":"[REDACTED]","app_time":607812,"app_v
ersion":"1.4.0.14","app_key":"idi_dotnet"}
```

I checked the configuration options to see if possibly I had unwittingly misconfigured by clicking 'Next' repeatedly on the installer. It turns out that you can enable HTTPS for authentication, but for some reason this is not the default. Also, for some reason, they find it necessary to specify that all "file data" will be send using HTTPS. We can't figure out why this really matters if authentication is done using only HTTP.



Further examining the client, I checked my inbox to find my welcome email. This one was another winner. In a world where far too many people send me my password in plaintext, I was delighted to see that JustCloud doesn't send passwords in email. All they include is a link used to validate your email address, right?

Not quite. It turns out that the link **LOGS YOU INTO THE SERVICE**. What's better is that the link doesn't expire. It's good for multiple authentication attempts. It's like free beer that never goes stale.

**JustCloud** <support@justcloud.com> 1:  
to joe ▾  
Hi joe,  
Congratulations on making the smart decision to backup your computer.  
Once installed, JustCloud will automatically start to backup your files and store them securely online.  
To view, open, download, or share your files just login to your online control panel.  
Control Panel: [http://www.justcloud.com/li?11\\_14706111\\_unkown](http://www.justcloud.com/li?11_14706111_unkown) [REDACTED]  
Login Details:  
Email: [REDACTED]  
Password: \*\*\* Chosen By You \*\*\*




### General Post Exploitation Ideas

- Steal Backup Files
- Upload Infected Files to the Cloud
- Find additional connected devices
- Cancel the account, deleting all stored files
- Find new contact information
- Discover deleted files that were previously synchronized

So how is buffoonery like this useful for post exploitation? Well, imagine you compromise an email server in a pen test. One of the first things I'm looking for now are

emails from JustCloud. This is a real win since you can access all files a user has backed up using this link that **never expires**. This is desirable even if you have network access to the victim machine from which the files were archived. If you download 20 gigabytes of data from a victim network, anyone running network monitoring software (such as the awesome Security Onion distro) would have to be blind to miss it. However, the victim network's monitoring software won't see you downloading from the service provider and that's a win for you!

Also, the user may have put a phone number in the client in an attempt to get more free space or something similar. Here we were able to find phone numbers of a victim just by clicking the link from a compromised web server. Configured email addresses are also a possibility, but in this case we gained access to the portal by stealing an email link (so we presume you already have that).

 My Account	 Transaction History	 Health Check
Name	Bilbo Baggins	
Email	[REDACTED]	
Phone	706-555-1234	
Cell Phone	706-555-6789	

Transaction history, connected devices, and connection logs are also useful when profiling a victim. On most systems, once you own the web portal account, you own the user's account.

What about sending files to a victim machine? That's a possibility too. Most providers allow you to send files from the web portal directly to the synchronized devices. The upload dialog shown below is from JustCloud, but it is representative of other service offerings. After compromising the web portal accounts, you could easily upload infected files (or replace already present files with infected copies) that a user may later open.



## Upload to 'Sync Folder'

Choose a file to upload to your sync folder. You can upload multiple files at once.

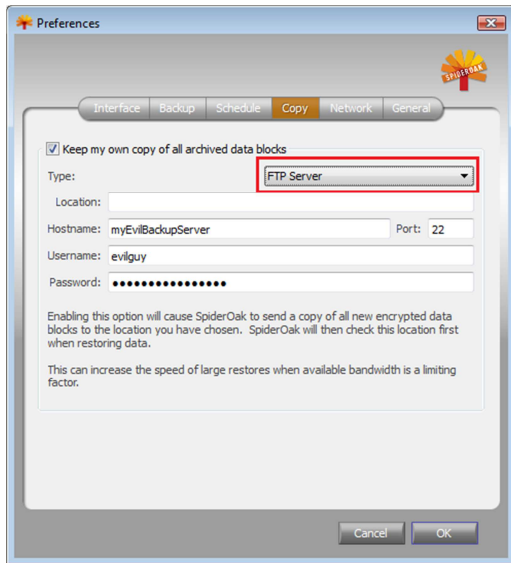


### **RSS Anyone?**

Dropbox has a useful post-exploitation feature we didn't find in any other service offering. Dropbox allows you to configure RSS updates. This is tremendously useful if you want to CyberStalk your victims from afar. Dropbox does implement some security with this RSS feed. The links provided in the RSS feed do not allow direct downloading of the files, you'll need to re-authenticate to the service to download files. However, in some cases filenames are enough to derive context. Best of all, your monitoring of the RSS feed isn't logged anywhere that's visible to the victim.

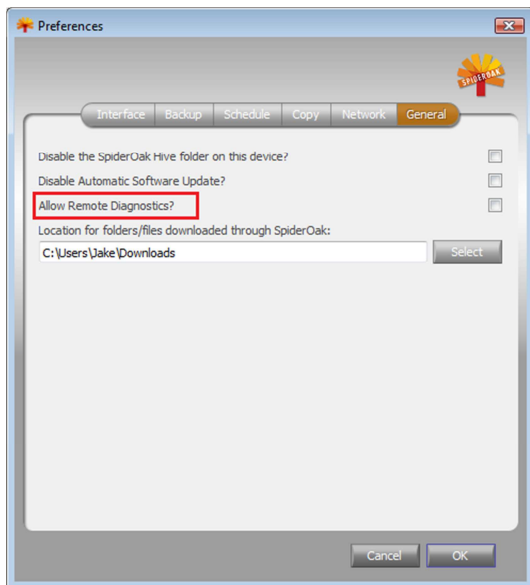
### **Network Backup**

We found an odd configuration option in SpiderOak. Unlike most providers, SpiderOak encrypts all data on the user's machine before uploading to the cloud. They claim to never store the key, so you're data is completely safe. Based on our testing, we tend to believe them. However, we were very interested in a configuration option that allows you to backup encrypted blocks (the same things being sent to the cloud) on a networked machine of your choosing. Note that you can send the data to an FTP server. Since the credentials are sent in the clear, this provides an easy vector for stealing credentials (useful when pivoting). Open the configuration, change the server to an address you own, fire up an FTP server there, start a packet capture and win. This technique works even if the user has configured sFTP as the protocol of choice. Just change to FTP and you get the user's credentials on that server. This allows us to pivot to that server to see what else we can see. Also, we all know how common credential reuse is ☺.



## Remote Diagnostics

We ran out of time before the submission deadline to run this one to ground, but we found something called “Remote Diagnostics” that can be enabled. Whatever this is, we’re betting it’s not good for your security. Luckily the default is disabled. However, when troubleshooting a problem, most users will enable anything and everything.



## Log files

In addition to log files in the cloud, many services also create log files on the host. These are useful because some services leave these folders in the user’s profile even after they are uninstalled. Many providers write the log files to the user’s roaming profile. This has significant implications if you have compromised a server with roaming profile backups. Note that many providers do not remove the logs when the product is

uninstalled. They are left on the machine along with other configuration settings (presumably in case a user later re-installs the product?).

```
filedb.cpp:2954!FileDB::GetNewDBEntryPos (DETAIL): FILEDB: putting entry at end of db, pos 2068
filedb.cpp:1892!FileDB::RecordItem (DETAIL): FILEDB: writing file F60DD78EAC95B915!104 to pos 2068, flush: 0
drive.cpp:2643!sendEventToUI (DETAIL): sending a event 3 for 'regshot-skydrive.txt', caused by user 0 on install 0
filestatusnotifier.cpp:562!FileStatusNotifier::LogItemStatusNotification (DETAIL): path: C:\Users\Jake\SkyDrive\regshot-skydrive.txt
apiLoop.cpp:175!ApiLoop::ActivityEventHandler (DETAIL): ActivityEventHandler called
```

## SQLite Databases

Many providers use SQLite databases. While Dropbox encrypts their databases, most providers do not. This allows you to find file names, file sizes, and in some cases even hashes of files that have been synchronized to the cloud. This is particularly useful for finding files that once existed on a victim machine but are no longer present.

path	fileName	hash	size
{syncfolder}	{syncfolder}		0
{syncfolder}\	JustCloud Quick Start Guide.pdf	9c92b30d88e47418651552e040561f76	991271
{source:11210377}\	C:		0
{source:11210377}\C:\	Users		0
{source:11210377}\C:\Users\	Jake		0
{source:11210377}\C:\Users\Jake\	Desktop		0
{source:11210377}\C:\Users\Jake\Desktop\	desktop.ini	9e36cc3537ee9ee1e3b10fa4e761045b	282
{source:11210377}\C:\Users\Jake\Desktop\	regshot.exe	aaa8ffbcace9c4999a77d63e0fa80f85	73728

**About the author:**

Jake Williams, the Chief Scientist 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, Shmocon, 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 20<sup>th</sup> 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](https://grepular.com/Dropbox_Mobile_Less_Secure_Than_Dropbox_Desktop)