

DC207 HARDWARE HONEYPOT WORKSHOP

Hi, and welcome to the DC207 Hardware HoneyPot workshop. In this workshop, you'll be learning how to flash and customize code on microcontrollers. We'll be downloading and installing the Arduino IDE and loading, customizing a couple of different miniature honeypots for you to experiment with within this workshop. Finally, we'll also look at another implementation of using MicroPython to emulate a telnet server.

In your kit, you have an ESP8266/WeMos D1 Mini Pro clone. It's a minimal development board that can operate as a wifi router, hotspot, and client. You'll also have a 3D printed case (excuse the irregularities) and a USB cable.

During the workshop, we'll have USB flash disks with all of the software you'll need. In addition, all of the necessary software for this workshop can be found in this GitHub repository and other links included in the repository.

https://github.com/BallinBallen/dc207_hpworkshop

One note: Your microcontroller comes packaged with headers. We won't be using them in this workshop but hang on to them. They can be used to add shield modules that can extend the use of your honeypot by adding small LED screens, SD card readers, and various sensors. Digging into the hardware is a little out of our scope, but I've included a helpful video that introduces the hardware much more in-depth. Feel free to explore as you see fit. 😊

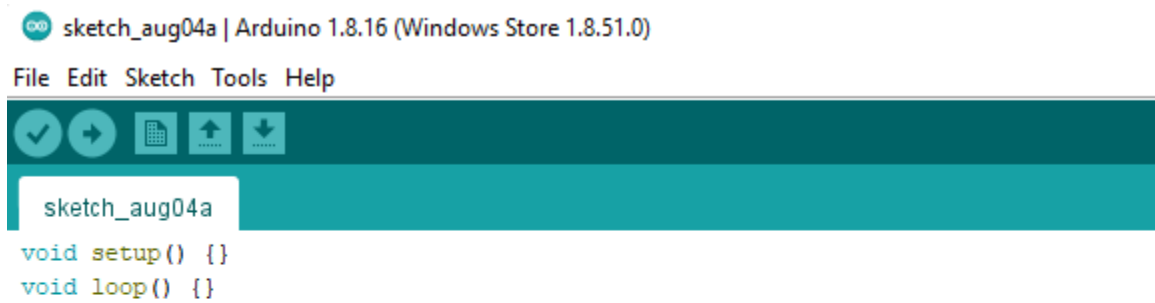
Let's jump into it.

Install and configure the Arduino IDE

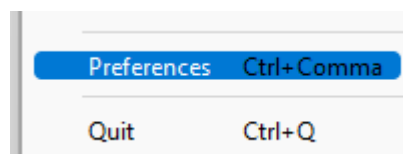
Let's install the Arduino IDE. You're welcome to download this off the internet, but if you've downloaded the repository or you're using a UFD in the workshop, the IDE installers can be found in the following folder:

3D Printed Case Files	10/20/2021 7:57 AM	File folder
Arduino IDE	10/20/2021 7:15 AM	File folder
Extending The Honeypot	10/20/2021 7:16 AM	File folder

We've got MacOS and Windows – Linux users; You're on your own. 😊 After installing the software, open up the IDE, you should see a screen like this;

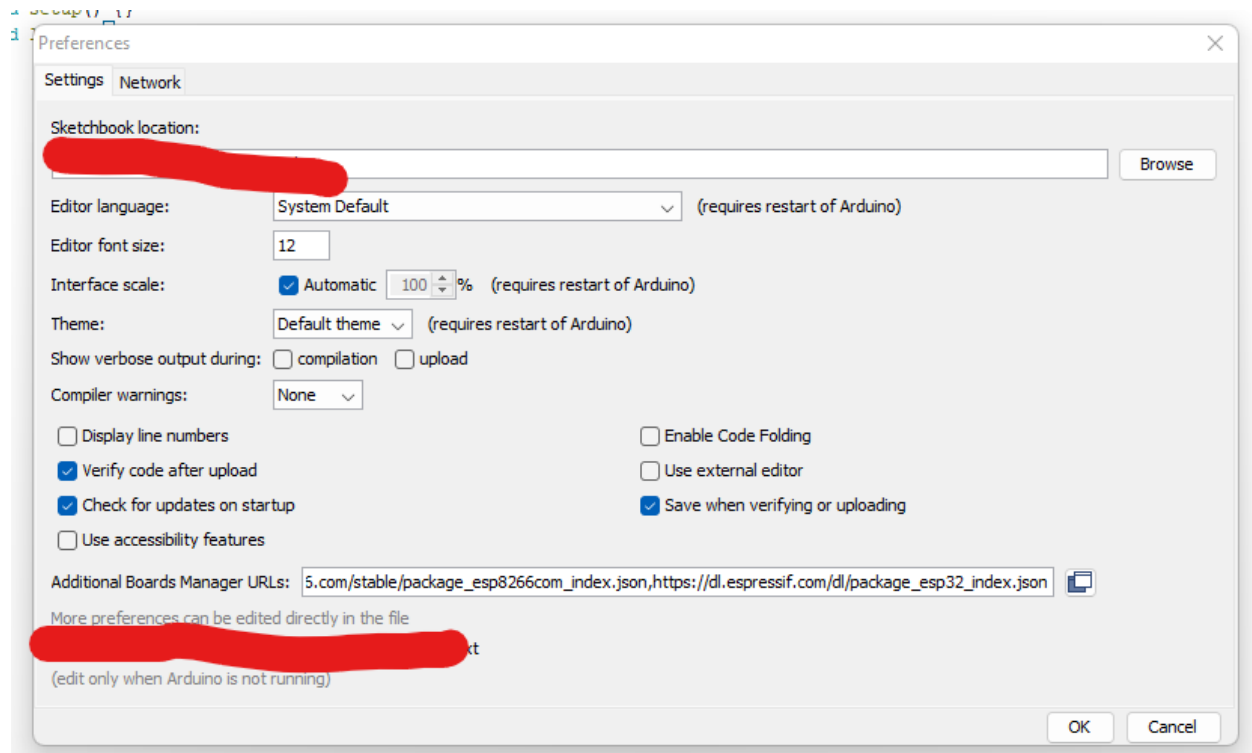


Cool. Unfortunately, the IDE isn't set up with the software we'll need for our workshop. We'll need to set that up ourselves. Start by adding ESP8266 to the board's manager URLs. To do this, open preferences

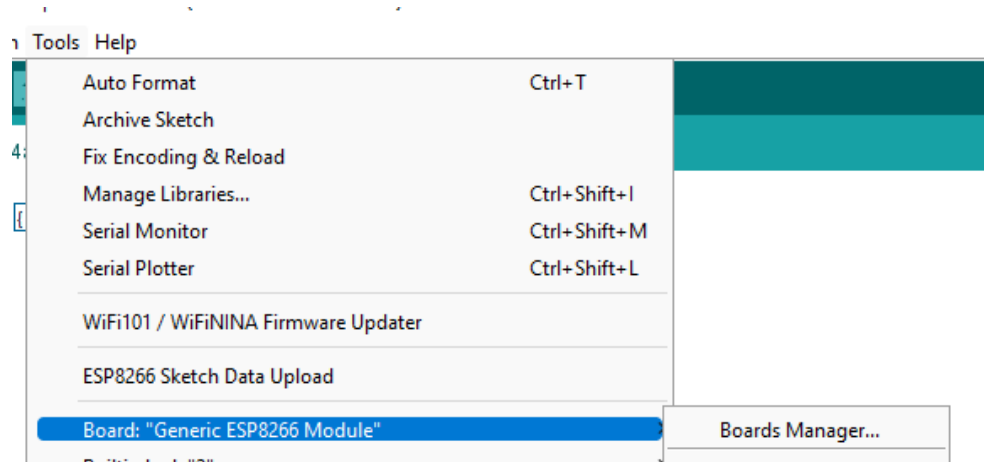


Under the "Additional Boards Manager URLs" text field, add the following text;

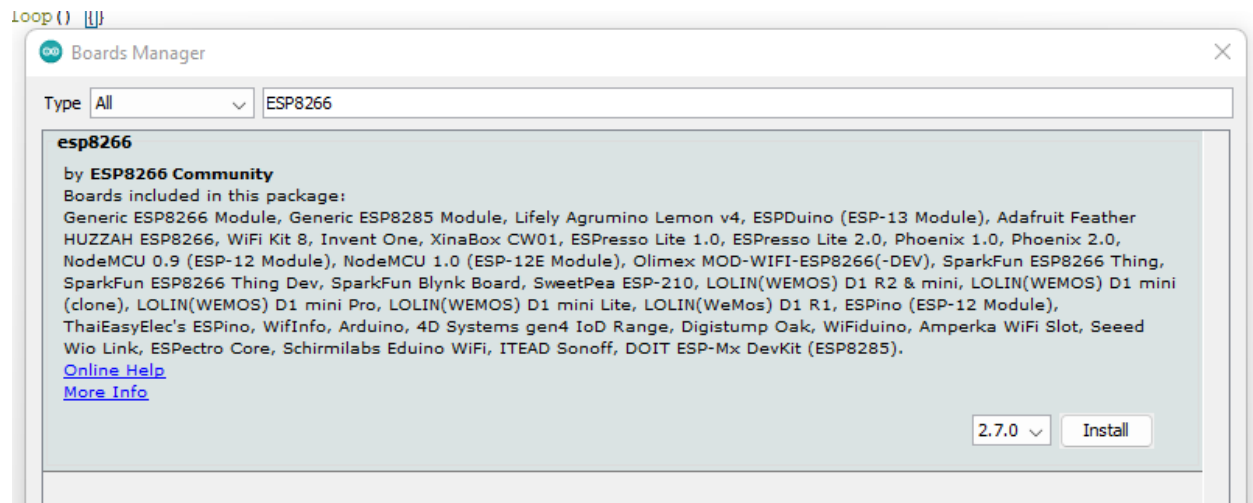
http://arduino.esp8266.com/stable/package_esp8266com_index.json



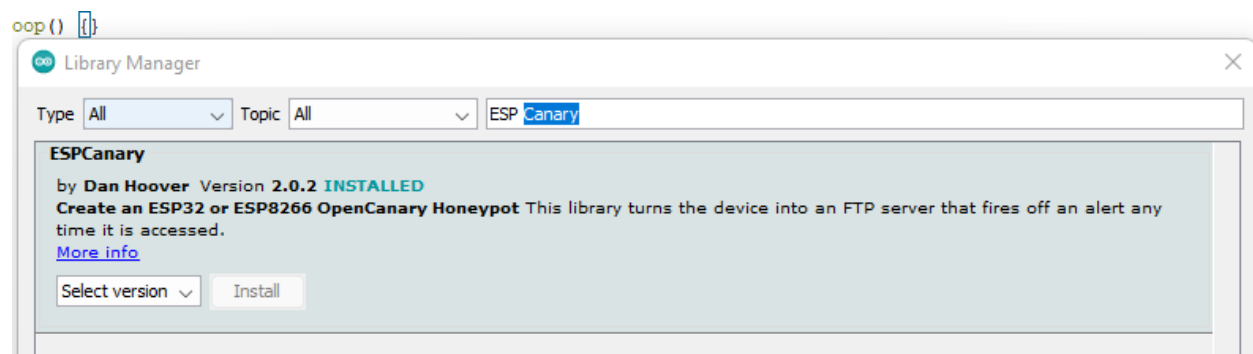
Next, we'll load the libraries—open tools → Boards → Board Manager.



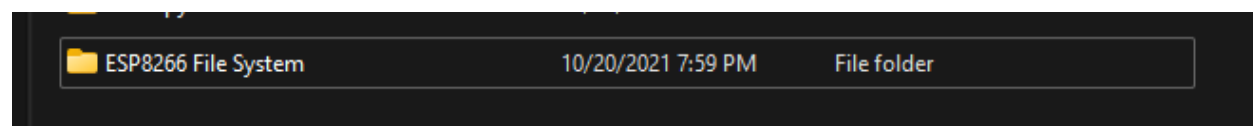
Once opened, type ESP8266 into the search box. You'll see the ESP8266 boards. **Select version 2.7.0 (newer ones are not compatible with our target libraries)**, and click install.



Next, we'll verify the libraries we want are installed and ready to go. Head over to Tools → Manage Libraries. Once open, search for 'ESP Canary.' If it's not already installed, install the latest version.



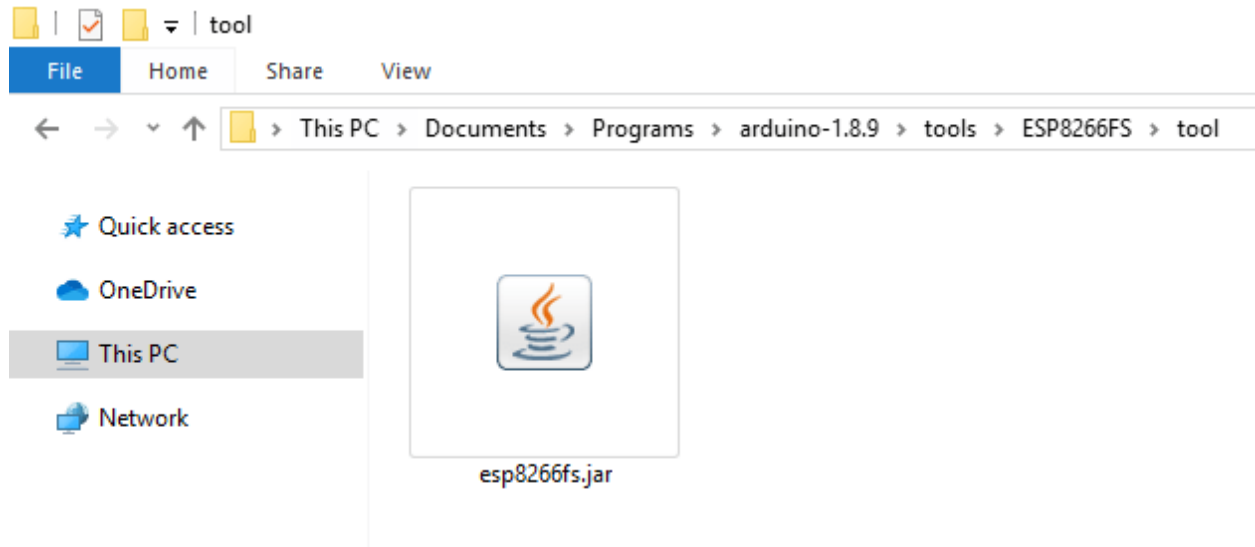
Finally, we'll want to install the ESP8266 Filesystem Uploader. It can be found in our folder here:



Or, you could download the following ZIP file:

<https://github.com/esp8266/arduino-esp8266fs-plugin/releases/download/0.5.0/ESP8266FS-0.5.0.zip>

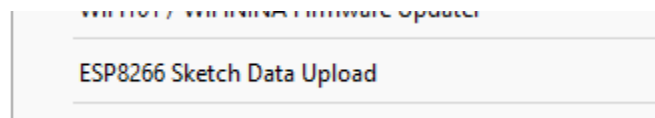
Once downloaded, unzip the file into the Arduino tools folder. This folder will usually be just inside where the IDE is installed.



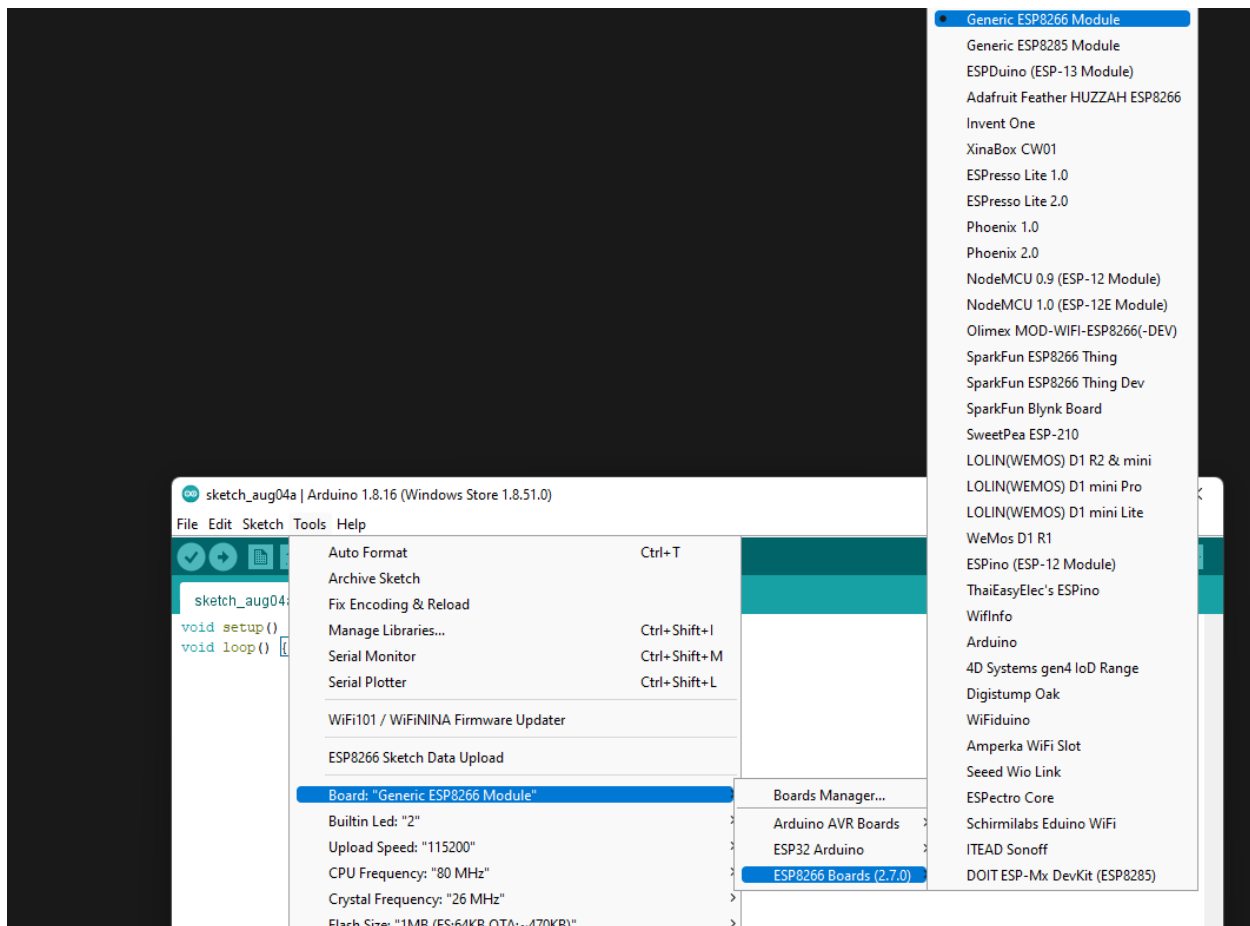
Your file system structure should look like the following:

`<home_dir>/Arduino-<version>/tools/ESP8266FS/tool/esp8266fs.jar`

Once you're finished, restart the Arduino IDE and check to see if the tool is correctly installed. Next, navigate to tools → and then look for the ESP8266 Sketch Data Upload menu option. If you see the following, you're good to go.



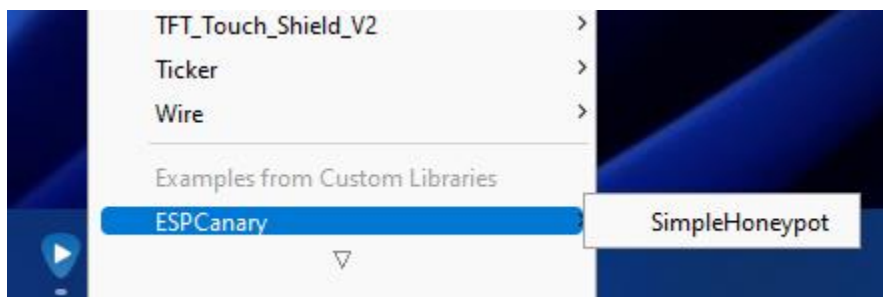
Finally, we'll need to select the board we're using. Navigate to Tools → Board Manager → ESP8266 Boards. Then, click on the Generic ESP8266 Module.



Whew, that's it. Let's get our first sketch (or code) loaded onto our device.

Configuring the Code

Most libraries come with example code, and the ESP Canary Library is no exception. We'll start by opening the example sketch. Head to File → Examples → ESPCanary → SimpleHoneyPot



This example allows you to set up a simple FTP honeypot. There are a few things we'll want to set up to start exploring this. First, let's start with the MAC address. A Media Access Control (MAC) address is a unique identifier assigned to a network interface controller as a network address in communications

within a network segment. It's a 12 digit hexadecimal number, and the first six characters in the address denote the manufacture of our device. Our honeypot examples appear as a network storage device by default, but you can set this to any valid MAC address. This is placed under the "newMACAddress" variable.

```
// by default we're using a Synology mac address (0x00, 0x11, 0x32)
// must be unique per network
uint8_t newMACAddress[] = {0x00, 0x11, 0x32, 0x07, 0x0D, 0x66};
```

MAC Addresses and the manufacturers they're associated with is public knowledge, and you can find a list of the most popular MAC addresses here:

<https://udger.com/resources/mac-address-vendor?v=top>

For my example, I'm going to set up the device as a Xiaomi device. Under Xiaomi's list for MAC address ranges we'll pick the first one:

vendor homepage	http://www.mi.com/
Assigned MAC range	format 1
	00:9E:C8:xx:xx:xx
	00:EC:0A:xx:xx:xx
	04:7A:0B:xx:xx:xx
	04:B1:67:xx:xx:xx

To add this address simply append the last two characters to the first three digits of the device. It should look something like this:

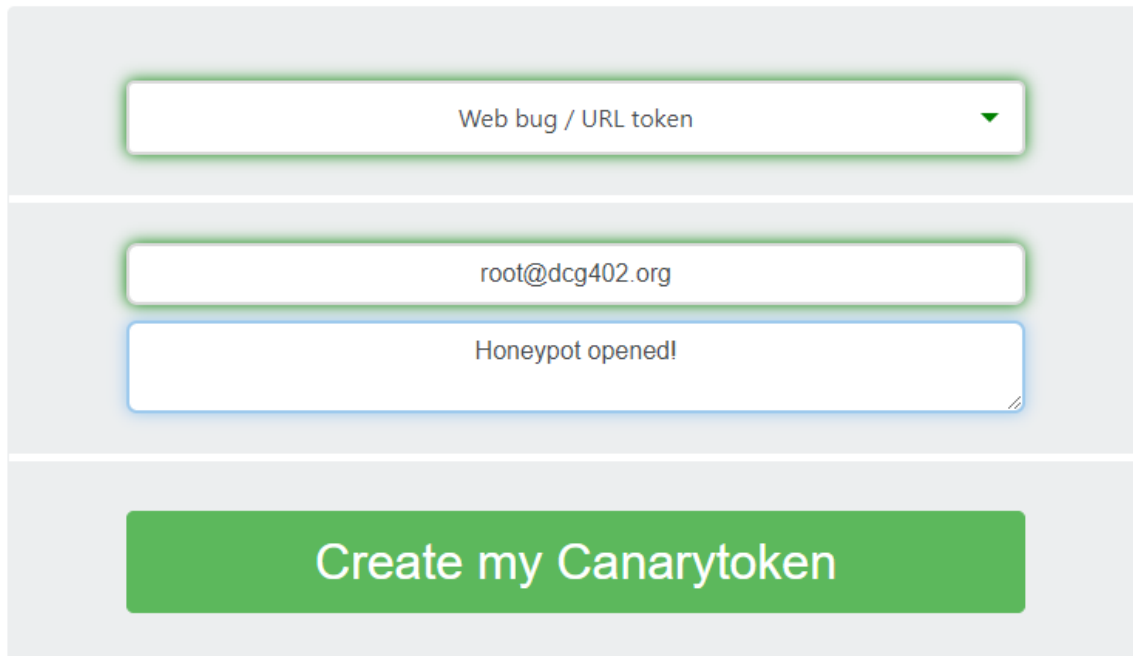
```
// if you would like to create your own mac address for your canary...
// by default we're using a Synology mac address (0x00, 0x11, 0x32)
// must be unique per network
uint8_t newMACAddress[] = {0x00, 0x9E, 0xC8, 0x07, 0x0D, 0x66};
```

You can (and should) set the remaining values in your MAC address to other, random values to prevent network collisions if you plan on deploying multiple honeypots. Next, set up your wireless network connection. For this in-person lab, we'll be using the following details to connect to a router – but make sure you set yours up to match the environment you'll be using the honeypot in.

```
const char* ssid = "internet_of_trash";
const char* password = "xiaomi_smart_broom_home_router";
```

The next element we'll configure is our canary token. Now this is optional, you can roll your own webhooks and customize this a great deal more, but setting up a free canary token is the easiest way to

get ourselves up and running. Head over to <http://canarytokens.org> and set up a URL token with your e-mail address and a note to remind.



Web bug / URL token ▼

root@dcg402.org

Honeypot opened!





Create my Canarytoken

If this is the first time you've used CanaryTokens, know that this is a simple service that allows you to receive a notification if someone (or thing) accesses a URL, document, or API key. Once you've created your token, copy and paste it into the URL placeholder.

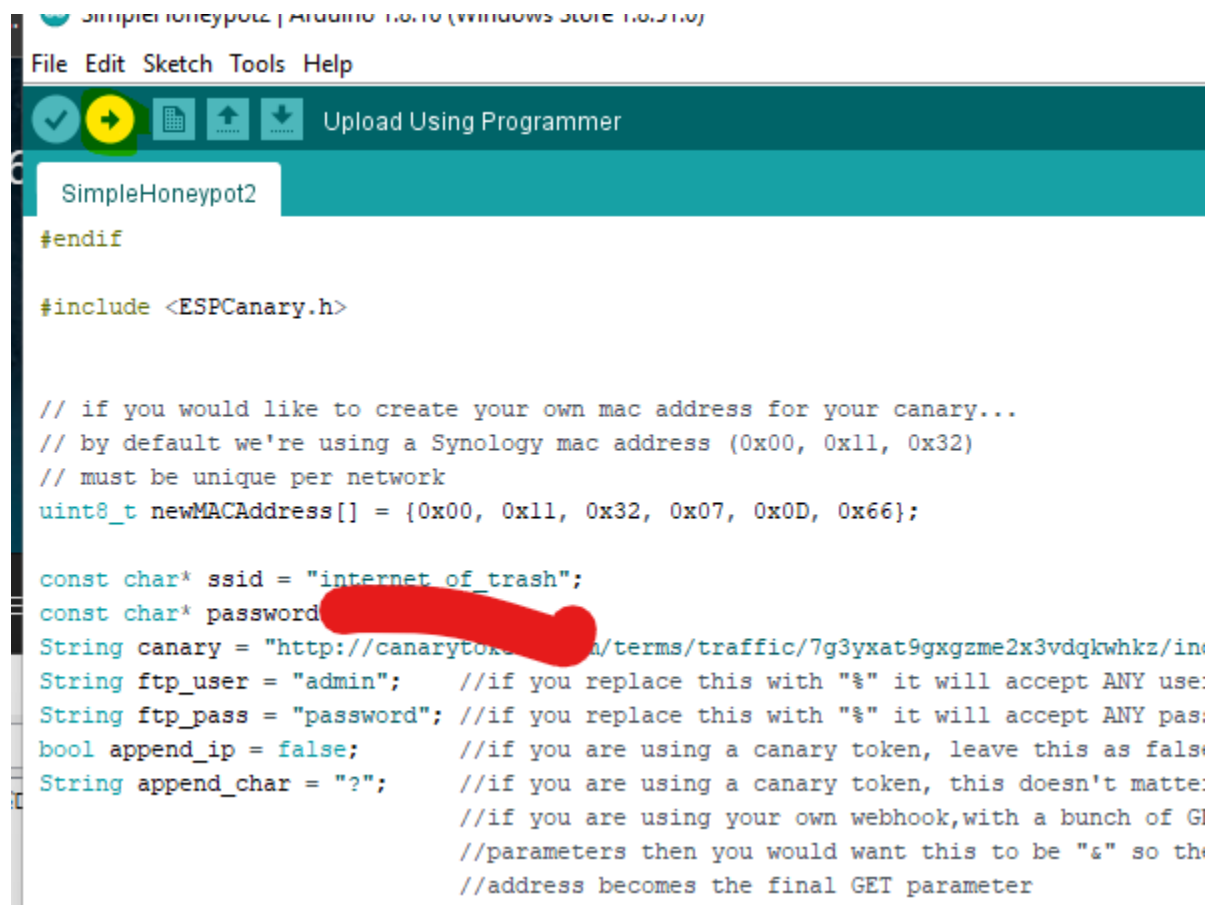
```
const char* ssid = "internet_of_trash";
const char* password = "xiaomi_smart_broom_home_router";
String canary = "http://canarytokens.com/traffic/terms/nibtyu6lz7qikr9fbdf6a5783/contact.php";
String ftp_user = "admin";    //if you replace this with "%" it will accept ANY username
```

We can configure a few more things, such as a username and password on your FTP server and additional settings if you don't plan on using canary tokens. Once you've set up the options you'd like to use, let's flash the code to our device.

Plug in the D1 Mini into your computer and make sure it's correctly installed. You can head to the device manager and look for COM and LPT ports if you're on Windows. If the device is working correctly, you should see a USB to Serial device here.

- >  Portable Devices
- ▼  Ports (COM & LPT)
 -  USB-SERIAL CH340 (COM5)
- >  Print queues

Next, we'll upload our code to the D1 Mini. To do that, click the upload button on the IDE, which is an arrow pointing left.



The screenshot shows the Arduino IDE interface. The title bar reads "SimpleHoneypot2 | Arduino 1.8.10 (Windows Store 1.8.10)". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains several icons, with the "Upload" icon (a green circle with a white left-pointing arrow) highlighted by a green circle. To the right of the toolbar is a button labeled "Upload Using Programmer". Below the toolbar, the sketch name "SimpleHoneypot2" is displayed in a teal box. The main editor area shows C++ code for a SimpleHoneypot2 sketch. The code includes a header file, defines MAC and SSID, and sets up various parameters for a canary token and FTP access.

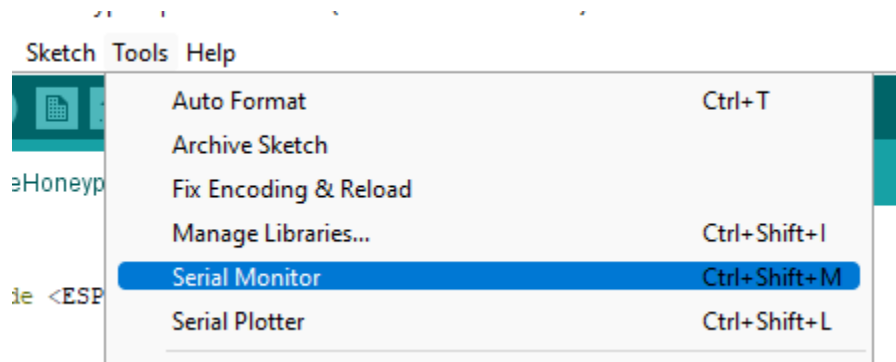
```
#endif

#include <ESPCanary.h>

// if you would like to create your own mac address for your canary...
// by default we're using a Synology mac address (0x00, 0x11, 0x32)
// must be unique per network
uint8_t newMACAddress[] = {0x00, 0x11, 0x32, 0x07, 0x0D, 0x66};

const char* ssid = "internet_of_trash";
const char* password = "password";
String canary = "http://canarytoken.com/terms/traffic/7g3yxat9gxgzme2x3vdqkwhkz/in";
String ftp_user = "admin"; //if you replace this with "%" it will accept ANY user
String ftp_pass = "password"; //if you replace this with "%" it will accept ANY password
bool append_ip = false; //if you are using a canary token, leave this as false
String append_char = "?"; //if you are using a canary token, this doesn't matter
//if you are using your own webhook, with a bunch of GET parameters then you would want this to be "&" so the
//address becomes the final GET parameter
```

While the code is being flashed, you can monitor the serial port with the serial monitor tool. This basically lets us watch the console output and read messages. Really helpful if we need to troubleshoot this or if something goes wrong. You can find this in the Tools menu under "Serial Monitor".



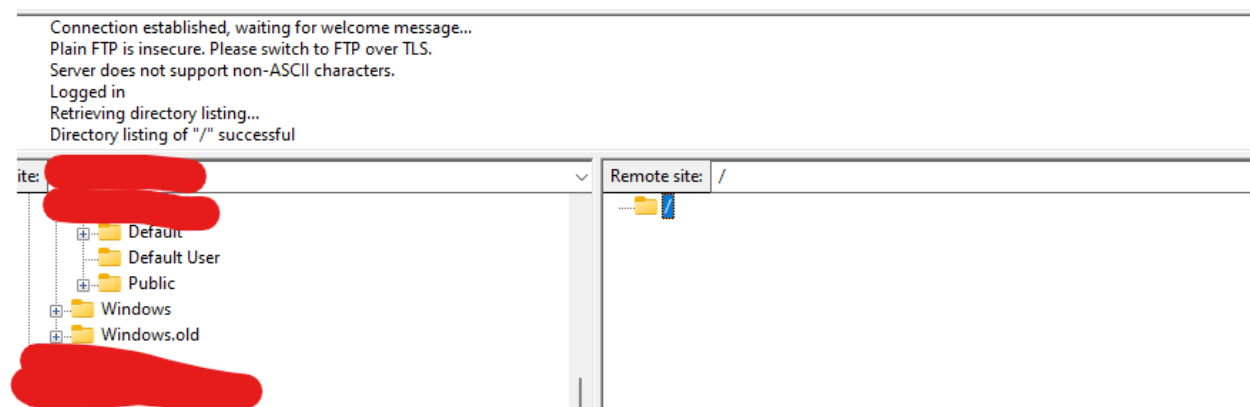
You should see scrolling output here, and if the upload was successful, you'd see the following:

```
11:38:21.275 -> .....
11:38:25.310 -> IP address: 192.168.68.123
11:38:25.310 -> MAC address: 00:11:32:07:0D:66
11:38:25.310 -> SPIFFS opened!
```

Ok great. With a FTP client, try to connect to your FTP server. You'll be able to log in with the username and password you may have set earlier. Once logged in, return to the serial monitor.

```
12:00:39.856 -> Connection made from 192.168.68.118
12:00:39.856 -> Attempting Canary
12:00:39.856 -> Connecting to http://canarytokens.com/terms/traffic/7g3yxat9gxgzme2x3vdqkwhkz/index.html
12:00:39.856 -> POSTing JSON
12:00:39.856 -> {"ip":"192.168.68.118"}
```

If you see this, you've got a little working canary that will alert you to if someone logs into your FTP server. If you set a valid e-mail address, you'll even get an alert. Very cool... but there are no files on the FTP server.



Not very exciting. You can create some exciting files to make it look more like a proper FTP server. We'll do this again with another Canarytoken. Head back to canarytokens.org and this time, create a word document.

Microsoft Word Document ▼

|

dc208@gmail.com

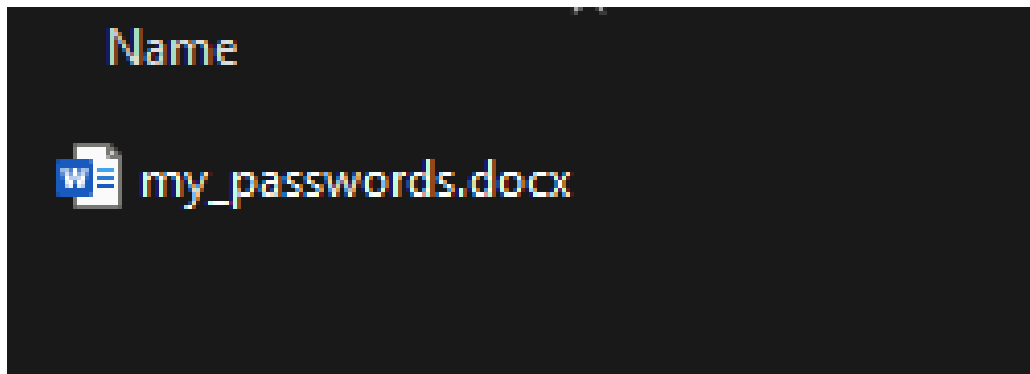
My password document alert

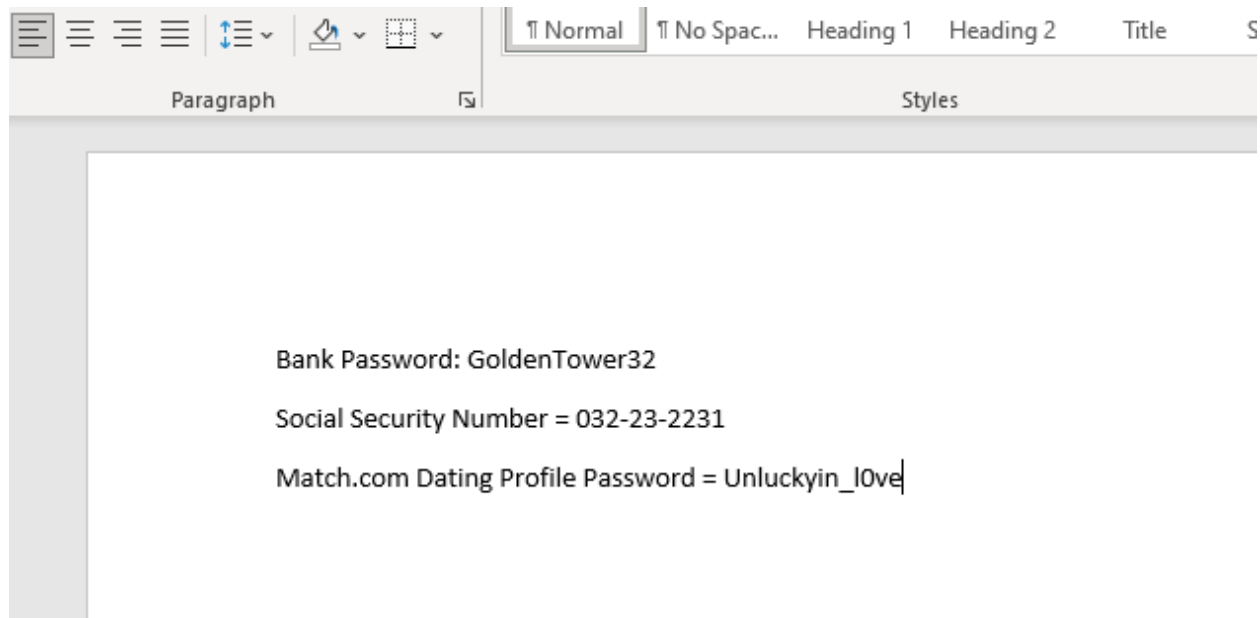
Create my Canarytoken

Brought to you by [Thinkst Canary](#), our insanely easy-to-use honeypot solution that deploys in just four minutes. **Know. When it matters.**

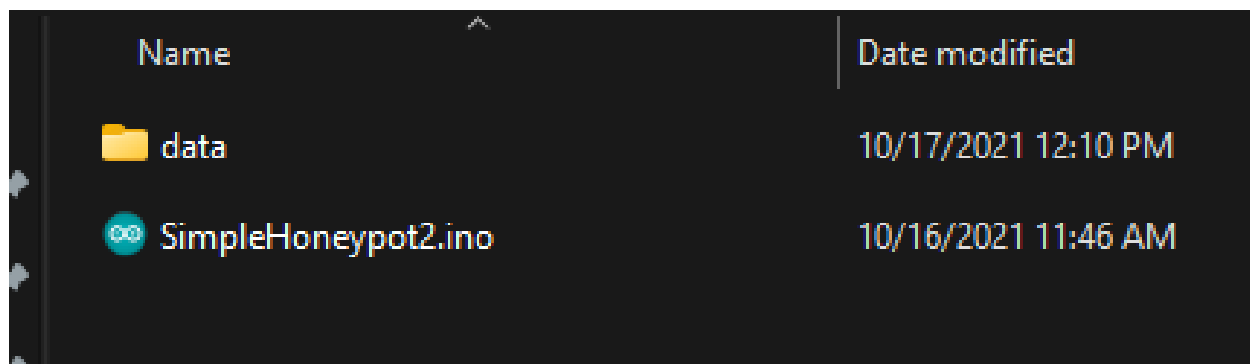
© [Thinkst Canary](#) 2015–2021

Download the file and rename it to something interesting an attacker might want to open and feel free to populate the document with exciting entries.





Great. Now let's upload the file. To do this, we'll need to create a folder where our Arduino sketch is kept. To open it where it's kept, select Sketch → Show Sketch Folder. Once you have the folder open, create a new folder called 'data'.



Now copy your word document into the data folder. Ensure the Arduino console monitor is closed while you do this; otherwise, you'll get a permission denied error. Now you're all set to add your document. In the Arduino IDE, click tools → ESP8266 Sketch Data Upload. It'll upload your document for you in just a few moments.



```
esptool.py v2.8
Serial port COM5
Connecting....
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: a8:48:fa:c0:40:10
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 65536 bytes to 16960...
Wrote 65536 bytes (16960 compressed) at 0x000eb000 in 1.9 seconds (effective 278.6 kbit/s)...
Hash of data verified.


Leaving...
Hard resetting via RTS pin...
```

Nice. Check your FTP server again, and this time you should be able to see the document we uploaded.

new site - admin@192.168.0.123

Remote site: /



Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group
..					
 my_passwords.docx	19,155	Microsoft ...	1/1/2000 11:06:...		

Extending The Honeypot

Hopefully, this has given you some ideas and gotten you a bit familiar with the Arduino IDE. As we mentioned in the introduction, this device can run in a few different ways, and one of those ways is a router. An excellent example of this which modifies our example just a little bit is in the following example over at https://github.com/skickar/ESP8266_Router_Honeypot, also located on in the files under 'Extending the Honeypot,' we'll set up our honeypots to act as a wifi access point which will broadcast a SID and forward traffic to another network.

Check out the code; try setting it up. First, you'll need to set up a new sketch in the Arduino IDE. Note that there are a few additional settings we'll need to configure; check the comments in the code for more information.

Thinking about this, or talking to the person next to you... how else do you think you could extend this honeypot? Remember, our headers in the packaging allow for the stacking of shields (such as an LED screen).

Python Telnet Bank Honeypot

Did you know you could run MicroPython code on microcontrollers? MicroPython is a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library and is optimized to run on microcontrollers and in constrained environments. If you already know python, this might be an easier way to start developing your microcontroller honeypots. Check out the following repository: https://github.com/gbafana25/esp8266_honeypot. (Also included in the files). It's a MicroPython Telnet Honeypot that emulates a banking system.

```
# This is a module with all the fake system messages
# If you have any other messages to add, put them here, and then implement them in esp8266_honeypot.py

welcome = b'\r\n\r\nFDIC FRONTIER SAVINGS BANK\r\n\r\nWARNING: UNAUTHORIZED USE IS PROHIBITED BY LAW\r\n\r\n'
# array_number 0 1 2 3 4 5 6
commands = [b'SYS_STATUS', b'DB_LIST', b'ACCOUNT_MANAGER', b'DEVICE_MANAGER', b'NAS_STATUS', b'EXIT']

valid_commands = b'\r\n\r\nSYS_STATUS - DISPLAY SYSTEM INFO\r\n\r\nDB_LIST - SHOW CURRENT DATABASE LISTING\r\n\r\nACCOUNT_MANAGER - OPENS ACCOUNT MANAGER\r\n\r\nDEVICE_MANAGER - SHOW CUR

prompt = 'FSB_MNGR:---> '

sys_info = b'\r\n\r\nKRAMER MANAGEMENT SYSTEMS\r\n\r\nENHANCED DATABASE / DOMAIN CONTROLLER / TELNET SERVER SYSTEM V2\r\n\r\nOS: GNU/OS\r\n\r\nVERSION: 3.04R3\r\n\r\nCPU USAGE: 12%\r\n\r\nRAM U

sys_info_shortened = b'\r\n\r\nKRAMER MANAGEMENT SYSTEMS\r\n\r\nENHANCED DATABASE / DOMAIN CONTROLLER / TELNET SERVER SYSTEM V2\r\n\r\n\r\n'

database = b'\r\n\r\nACCOUNT LIST:\r\n\r\n-----\r\n\r\nBUSINESS: 45,903\r\n\r\nHOME: 73,459\r\n\r\nCURRENT TRANSFERS: 7,824\r\n\r\n\r\n'
```

Please take a few moments to review the code and understand how it operates, what you can customize, and how you can load the code with the install scripts. How is this different from our previous Arduino projects?

Closing Out the Workshop

I hope you enjoyed this little workshop exploring microcontroller devices as miniature honeypots. In addition, I hope it gives you some ideas on how to set up one of these for yourself. While these are certainly not as full-featured as most honeypots in the wild, these are inexpensive and can allow for some flexible implementations.