

# SkyNET: a 3G-enabled mobile attack drone and stealth botmaster

Theodore Reed  
*Stevens Institute of Technology*

Joseph Geis  
*Stevens Institute of Technology*

Sven Dietrich  
*Stevens Institute of Technology*

## Abstract

SkyNET is a stealth network that connects hosts to a botmaster through a mobile drone. The network is comprised of machines on home Wi-Fi networks in a proximal urban area, and one or more autonomous attack drones. The SkyNET is used by a botmaster to command their botnet(s) without using the Internet. The drones are programmed to scour an urban area and compromise wireless networks. Once compromised, the drone attacks the local hosts. When a host is compromised it joins both the Internet-facing botnet, and the sun-facing SkyNET. Subsequent drone flights are used to issue command and control without ever linking the botmaster to the botnet via the Internet. Reverse engineering the botnet, or enumerating the bots, does not reveal the identity of the botmaster. An analyst is forced to observe the autonomous attack drone to bridge the command and control gap. In this paper we present a working example, SkyNET complete with a prototype attack drone, discuss the reality of using such a command and control method, and provide insight on how to prevent against such attacks.

## 1 Introduction

A botnet is typically a network of compromised machines controlled by a botmaster. These networks, which can contain hundreds of thousands of hosts, are leveraged for cybercrime such as click fraud, identity theft, DDoS extortion, cyberwar, etc. [19]; botnets have become the leading threat to cyber security [16]. Botmasters engineer their networks to be resistant to analysis, detection, and disruption. Researchers have countered by developing techniques and tools which do just this [7, 10, 11]. This arms race, played by botmasters and researchers, has led to advanced botnets which use different forms of encryption, signaling, dormancy, and network formation such as peer-to-peer (P2P) based communication [8, 9].

Botmasters have another implicit goal when design-

ing their botnets, to remain hidden [14]. By using a P2P network, a botnet no longer relies on a central server for communication, and a botmaster no longer issues commands from a central location. The botmaster can issue a command to a peer bot, and the command will propagate through the network. The botmaster can use this feature to remain hidden by changing the injection point, the bot used to initiate the command propagation, with every message. But this method does not offer the controller complete isolation from the network [15].

Botnets have also enhanced their command and control (C&C) protocols by using social networking sites to communicate. By using external services a botmaster can issue commands by posting to an innocuous blogging website. These methods require analysts to obtain information from various blogging websites, which may never be used to deliver commands again. However, by analyzing bot behavior, trends and patterns may be identified which lead to command injection prediction [26]. By using Internet services to deliver commands, botmasters ultimately risk detection or the discovery of defensive measures against their botnet. Botmasters can use security measures to protect their networks by implementing encryption and integrity protection. However, traffic analysis can enumerate commands and identify the location or the identity of a botmaster [2]. These security controls may not hide patterns such as account names used on blogging websites, login time, or login location.

This paper introduces a technique to separate the botmaster from their bots. This technique strengthens the botnet by avoiding common network security mechanisms such as firewalls, intrusion detection systems, and event logging. C&C data enters the botnet via home Wi-Fi-enabled networks. Personal networks are the most unsecured networks on the Internet, as they often contain no security controls, unpatched machines, no logging or auditing, bad password management, and typically run wireless radio with poor security [12]. The technique we describe uses the insecurity of these networks and semi-

permanent C&C to protect both the botnet by reducing origin traceability, we call it SkyNET.

There are many ethical concerns with SkyNET, this paper serves as a waypoint for further research on related threats. We aim to preempt attacks that use out-of-band communication to control Internet hosts. The attacks, methods, and equipment described in this paper were tested in controlled environments; the protocols used to test were approved by the university’s Institutional Review Board (IRB).

The outline of the paper is as follows, we present the idea of using a SkyNET to enhance botnet C&C in section 2. In section 3 we briefly outline an attack framework used by a mobile attack drone; section 4 describes an example control protocol; section 5 provides a description of the design and feasibility of the prototype drone; section 7 provides possible defense measures against SkyNET; we identify how our work compares to related research in section 8; finally section 9 concludes the paper.

## 2 Approach

The goal of SkyNET is to create a botnet controlled by a stealth network, the network in the sky. When an attacker has a physical presence the number of possible attack vectors increases. SkyNET takes advantage of poorly configured wireless network security, and poor trust configurations on mobile devices, to join networks and access devices locally using a mobile attack drone. We call this a SkyNET drone, which is controllable via auto-pilot or via a mobile broadband (3G) connection.

Once network access is acquired, the drone utilizes an array of existing tools to compromise hosts, such as the Metasploit framework [22]. SkyNET then takes advantage of multiple assumptions made when implementing security measures to create a highly stealthy botnet. These assumptions include: assuming all network traffic must pass through network choke points [6], that attack traffic originates from the Internet, and that local network hosts can be trusted. The drone implements a 4-step attack procedure to enlist hosts into the network. We call this procedure PAAE (pilot, attack, attack, enlist).

**Pilot.** The drone’s first step is acquiring physical proximity. Using a client application, for a mobile phone or within a web browser, the drone controller (botmaster) pilots, and lands (landing saves power on the device, it is not a requirement) at an attack position. We developed a web client to control the drone, record Wi-Fi cells, and map each trip.

**Attack.** The pilot uses the web client to choose methods of attacking a network, this step usually includes gaining network access or creating a wireless network

(similar to a Caffe Latte attack [21]). This step may involve several trips used to scan and collect Wi-Fi statistics. The methods used for attacking or creating a network are detailed in the following section.

**Attack.** Once the drone joins a network with loyal hosts, it begins scanning and attacking. We use loyal to describe mobile and non-mobile hosts which consistently use the attacked network. We do not present any new attack vectors in this respect. We leverage existing network- and host-based attack frameworks by configuring and creating simple administrative mechanisms for the drone. An experienced controller can install and configure whatever security tools they desire.

**Enlist.** Once a host is compromised the drone attempts the enlist the host into SkyNET. Section 4 outlines an example two phase approach for enlisting hosts. One of SkyNET’s advantages is the ability avoid network-level detection by removing the requirement for Internet access. The methods used for enlisting a host should not pass through gateway routers, network choke-points, or the Internet.

## 3 Attack Framework

This section outlines some of the network and system level attacks used by the SkyNET drone. We assume these attacks are practical in a real-world urban area. The technical feasibility of the attacks was demonstrated by the drone in a controlled environment.

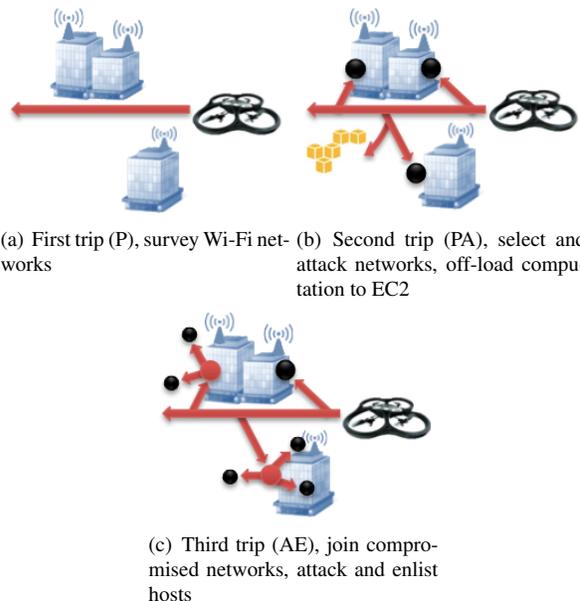


Figure 1: Diagrams showing the PAAE procedure used by the SkyNET drone. Black dots represent targets. In *b* the targets are networks. In *c* the targets are both networks and hosts.

### 3.1 Survey, Select & Attack

The SkyNET drone’s initial task is to survey local wireless networks in the area of interest. Information about the composition of local networks is gathered: BSSID, SSID, encryption type, channel, MAC address(es) of associated clients. Capturing handshakes and data across all channels at this point is not feasible as our monitoring wireless card has to rapidly cycle through channels to gather the access point and client association information efficiently. An attacker should structure channel selection based upon individual channels wireless network composition. This could be done through weighted metrics, including composition of encryption types, number of clients per network, and signal strength, allowing for more effective use of flight time. As open networks require no data collection or cracking to connect we can say that these are the easiest targets. It should be noted that wireless hotspots, although often open networks, may not provide loyal clients and may be ineffective for exploitation. Once SkyNET has determined the wireless network(s) to attack, it must crack encryption of the various wireless networks it wishes to access.

#### 3.1.1 Wired Equivalent Privacy (WEP)

WEP requires clients and access points to share up to four secret symmetric keys for communications. Most installations just use a single key called the root key. Using the improved PTW attack in Aircrack-ng [3], we can obtain a 95% success rate of cracking the key with between approximately 40,000 and 50,000 packets, depending on if the Initialization Vector is generated randomly or in counter mode respectively [27]. To put this into perspective, a common viral YouTube video<sup>1</sup> which is 228 seconds generates 30,208 packets at 320p (the default setting). With packet injection gathering the necessary amount of data takes a matter of minutes [17].

#### 3.1.2 Wi-Fi Protected Access (WPA)

SkyNET infiltrates WPA and WPA2 encrypted networks by attacking the Pair-wise Master Key. If a Pre-Shared Key (PSK) is used, then  $PSK=PMK$ . The PSK is 8 to 63 characters. This is the solution provided to home networks, and small enterprises that lack authentication servers. It is generated using a known algorithm:  $PSK=PMK=PBKDF2$  (password, SSID, SSID length, 4096, 256) where PBKDF2 is a method used in RSA’s Public Key Cryptography Standard #5, 4096 is the number of hashes, and 256 bit is the output length. The number of hashes required makes brute forcing computationally intensive and not suitable to be done on the drone [17].

The Pairwise Temporal Key each client session uses is derived from the PMK using the 4-way handshake. To attack the PMK, the attacker needs to capture the 4-way handshake messages. An attacker, the drone, can actively force a 4-way handshake to occur by deauthenticating an associated client, forcing the client to re-associate with the access point [17].

#### 3.1.3 Cracking

As the drone has limited computational power, cracking keys in wireless networks efficiently is not feasible. To compensate, the drone utilizes a 3G mobile data link to off-load computation to an Amazon Elastic Compute Cloud (EC2) GPU Cluster [1] instance running Aircrack-ng [3] and Pyrit [18]. The full utilization of the Aircrack-ng suite is available on the drone for packet capture and attacks requiring injection.

### 3.2 Attack & Enlist

Once the drone has access to a compromised network its second task is to attack hosts; preferring non-mobile hosts. The botmaster can deploy an array of attack scripts or frameworks. In our example we suggest using the open source Metasploit framework, scripted to run `autopwn` [22]. Once a host is compromised, the drone exchanges identification information, configures a callback mechanism, and secures the host as it is now a potential asset to SkyNET. A better outline of this exchange is described in the next section.

## 4 SkyNET Command and Control

In this section we describe an example protocol for controlling SkyNET. This protocol demonstrates a process of converting a compromised host into a bot controller (Enlist), and a process for commanding bot controllers. Controllers will receive command data delivered by a SkyNET drone, from the botmaster. In this example we demonstrate how the drone can be used as a secure and trusted channel (phase 1), and as an untrusted information relay (phase 2). Figure 2 shows these two phases for one bot controller (host/controller), separated by a star. The star represents a second flight of the SkyNET drone. We refer to this example protocol as a control protocol, but it also delivers botnet commands to bot controllers. An encryption or decryption key used between parties  $X$  and  $Y$  is denoted  $X_{enc,dec}^Y$  for communication from  $X$  to  $Y$  and  $Y_{enc,dec}^X$  for communication from  $Y$  to  $X$ . We call the botmaster  $M$ , a host or controller  $H$  and the drone  $D$ .

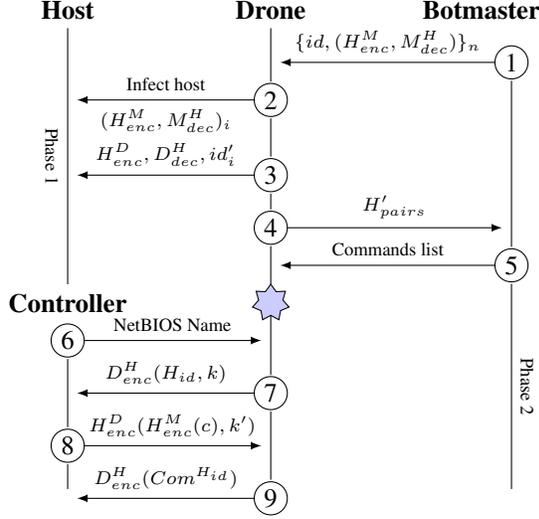


Figure 2: An example control protocol for SkyNET outlining two flight phases. Before the star the drone victims are nothing but compromised hosts; after the star, they are part of SkyNET.

#### 4.1 Phase 1 - Infection

The first phase of the example control protocol uses the drone as a secure channel to deliver keys to a potential botnet controller. This phase takes place before a host joins the SkyNET. To begin, a botmaster generates  $n$  pairs of encryption / decryption keys for  $n$  potential controllers and one botmaster ( $2n$  key pairs). Where  $n$  is an arbitrary value that remains constant, chosen by the botmaster. In this step the botmaster randomly generates  $n$  strings of length 15, and assigns each as an ID for each key pair. The chosen length 15 is the maximum length of a NetBIOS name, used later in the example. The botmaster records these ID and key pairs and sends the set to a drone, saved as set  $H_{pairs}$ . The second step is infection, this occurs when a drone compromises a loyal host on a compromised network.

We use infection as a general term for achieving system access, preparing persistence, and logging information about the host. The third step delivers five data items to the host, which will be used for potential identification and authentication. The drone generates two new pairs of encryption / decryption keys. One pair is for the host  $(H_{enc}^D, H_{dec}^D)$  and the other is for the drone  $(D_{enc}^H, D_{dec}^H)$ . These keys are used for subsequent communication between the host and drone. Then the drone selects a random ID and pair of keys as  $id_i, (H_{enc}^M, M_{dec}^H)_i \in H_{pairs}$ . These keys are used for potential communication between the host and botmaster.

This example does not implement PKI, every host has a unique key pair for a drone, and botmaster. The third

step ends when the drone sends the host two encryption keys  $(H_{enc}^D, H_{enc}^M)$ , two decryption keys  $(M_{dec}^H, D_{dec}^H)$ , and an ID. The drone does not send  $id_i$ , instead it generates a new random 15-character string  $id'_i$  and sends this. The drone then updates the  $H_{pairs}$  set for  $i$  by replacing  $id_i$  with  $id'_i$ .

The modified  $H'_{pairs}$  table, which includes the changed  $id$  values is sent to the botmaster in step 4 of Phase 1. The botmaster can identify which pairs were transferred to infected hosts as  $H'_{pairs} - (H'_{pairs} \cap H_{pairs})$ . When this occurs the botmaster verifies the integrity of the drone. In this phase it is safe for the botmaster to fly the drone and observe the streaming flight video. If the drone's flight is considered trusted then the set of infected hosts are considered joined to SkyNET. If the drone's flight is not trusted, then the botmaster saves no information and considers the hosts unusable.

The botmaster then prepares the drone for a subsequent flight which will be flown by auto-pilot. A commands list is generated and loaded on the drone by encrypting each command for a potential controller with that controller's encryption key  $(M_{enc}^H)$ . The list is stored as pairs of encrypted text (command) to controller identification ( $H_{id}$ ) with static size  $n$ . We describe this commands list in more detail later in this section. There is no key pair for botmaster to drone since the drone must be absolutely trusted by the botmaster on the first phase only. Subsequent flights to deliver commands need not be trusted.

#### 4.2 Phase 2 - Command Exchange

The second phase of the example control protocol demonstrates an untrusted use of the SkyNET drone. This phase loops as long as the botmaster or botnet controller need to communicate. This phase begins for a selected host, now called controller, or botnet controller, when the drone re-visits their compromised network. The controller advertises their ID, we suggest setting the NetBIOS Name as the  $H_{id}$ , and the drone looks up the corresponding key pair. In the seventh step the drone sends a challenge to the controller by encrypting  $H_{id}$  and a randomly chosen nonce  $k$  with the drone-controller encryption key  $(D_{enc}^H)$ . The controller decrypts the challenge using the drone-controller decryption key  $(D_{dec}^H)$ , and stores  $H'_{id}$  and  $k$ . If  $H'_{id}$  matches  $H_{id}$  then the drone has authenticated itself to the controller. In step 8 the controller responds by encrypting an optional message for the botmaster, encrypted with the controller-botmaster encryption key  $(H_{enc}^M)$ , and  $k$  as  $k'$  using the controller-drone encryption key  $(H_{enc}^D)$ . The drone decrypts the response using the controller-drone decryption key  $(H_{dec}^D)$  and stores the optional message  $m$ , and  $k'$ . If

$k'$  matches  $k$ , then the controller has authenticated itself to the drone, and the drone can deliver the corresponding command in step 9 by encrypting it with the drone-controller encryption key. The controller decrypts the command message with the drone-controller decryption key and then decrypts the command using the botmaster-controller decryption key ( $M_{dec}^H$ ).

### 4.3 Protections and Attacks

The SkyNET control protocol is designed to protect the botmaster and reveal as little information as possible about the network. We designed the protocol with 3 goals. 1) Controllers should not have a way of communicating to each other through SkyNET. Since SkyNET controllers are used as botnet controllers for an underlying botnet, it does not make sense for them to know of each other. If two SkyNET controllers are part of the same underlying botnet they should communicate through that botnet's C&C. 2) A SkyNET drone should appear unique to each controller. A controller  $A$  cannot determine if the drone communicates with any other controller  $B$ . 3) The integrity of the SkyNET does not depend on the integrity of the drone.

This third goal only applies to a SkyNET with one or more controllers, thus only during Phase 2. In Phase 1, step 3 delivers a unique identification, drone key pair, and botmaster decryption key to a potential controller. Step 4 then delivers all the modified, drone-generated identifications, to the botmaster. The botmaster must trust that the drone delivered these identifications to different infected hosts. Without verifying the integrity, a man-in-the-attack is possible. The integrity can typically be verified by observing the streaming video from the drone, as described in the next section.

#### 4.3.1 Compromising a Controller

In Phase 2 we assert each host, now turned controller, has no knowledge of other controllers. These controllers cannot communicate. If controller  $A$  wanted to send a message to controller  $B$ , the message would need to be federated through the drone. This is because controller  $A$  cannot encrypt messages for  $B$  since  $A$  knows only the encryption keys for the drone and botmaster. Furthermore, these encryption keys are unique to controller  $A$ 's respective communication. Federating messages through the drone only allows for impersonation of a drone,  $A$  must obtain  $B_{M_{enc}^H}$  to send commands to  $B$ . This information is only known to the botmaster, thus  $A$  must compromise or be the botmaster to communicate to  $B$ . If an arbitrary controller were overtaken the attacker would learn a minimal amount of information about SkyNET. The attacker could intercept com-

mands for that controller and the controller's botnet, and respond with commands as that controller. The attacker could not impersonate another controller, learn of the existence of other controllers, or the existence of drones which the controller does not trust.

#### 4.3.2 Compromising a Drone

If a drone is compromised or obtained by an attacker during Phase 1 the drone is forfeited. The hosts slated to become controllers, and the data contained on the drone are abandoned. The attacker will obtain the  $n$ -sized  $H_{pairs}$  but this data does not reveal any information about the botmaster or number of hosts already compromised. The drone is designed with multiple anti-tamper mechanisms to assure trust in a Phase 1 flight. If an attacker obtains a drone in Phase 2, the attacker gains a minimal amount of information about the SkyNET. The attacker will receive the  $n$ -sized set of ID, key, and encrypted text pairs ( $H_{pairs}$ ). Since there are no empty tuples, the attacker cannot determine which pairs are associated with controllers, thus the attacker cannot estimate the size  $s$  of the SkyNET except that  $s \leq n$ . The attacker can verify themselves to a controller, but cannot issue commands since they do not have  $M_{enc}^H$  keys. Preventing replay attacks can be solved by implementing a command counter and binding this information to the controller and botmaster's communication state [4]. Each time the botmaster encrypts a command for controller  $A$ , the botmaster will include a counter, then increment the counter. Controller  $A$  will know the count of the last command it received, and only accept commands for a higher-number count. A replay attack is now mitigated since this count is encrypted; a malicious drone cannot change the counter attached to a command, nor will controller  $A$  respond to a second occurrence of a counter.

The authentication of the drone is not required in step 7, but rather, is implicit of the authentication of the controller. Furthermore the authentication of the controller is an implicit solicit of a message for the botmaster, and an update to the controller's state to prepare it for a command from the botmaster. We imagine a scenario where a botmaster can use these implicit authentication steps to vary commands interpreted by a controller, based on the identification of the drone. These authentication steps help mitigate known-ciphertext attacks on the controller. Furthermore, these steps are not intended to provide assurance to the botmaster, since the drone's keys are susceptible to capture. We discuss physical attacks to the drone and SkyNET in the following sections.

## 5 SkyNET Drone

The prototype SkyNET drone is based on the AR.Drone [20] quadcopter. We added a lightweight

and low-cost single board computer (SBC) running Debian Lenny to the AR.Drone. In this section we describe the design decisions for the creation of the SkyNET drone, including the requirements to implement the SkyNET C&C protocol.

The AR.Drone is controlled via an Ad-Hoc Wi-Fi network connection. The quadcopter provides a controller API and SDK with example application code for Microsoft Windows, Ubuntu Linux, and Apple iOS. The AR.Drone includes a forward- and down-facing camera, sonic sensors for altitude control, an accelerometer for stability, and four independently controlled motors; the AR.Drone automatically stabilizes while hovering. The onboard computer is a 468MHz ARM9 processor performing video encoding, navigation, and telemetry, and running an embedded Linux operating system. The AR.Drone was selected for the base of the prototype because it supported the weight of our SBC payload, provided an easy to understand API. We augmented the AR.Drone with the following enhancements to create our SkyNET drone.

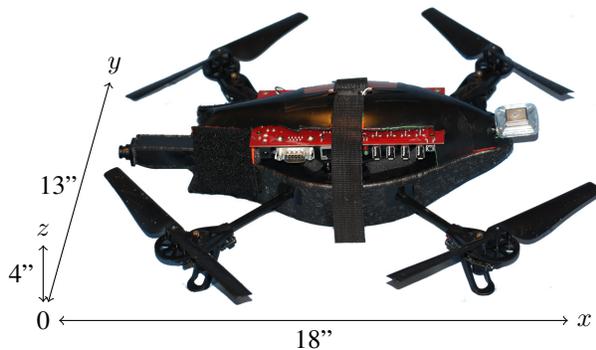


Figure 3: Our prototype SkyNET drone.

## 5.1 Drone Enhancements

**Linux Operating System.** The two attack steps performed by the drone require a large array of attacks, aimed at the network and system level. We wanted to implement the Metasploit framework and Aircrack-ng as examples, and needed a python environment to implement previously-written code for distributed WEP and WPA-PSK cracking. Therefore a Linux Operating System seemed most appropriate, with the flexibility to support hardware such as the 3G WWAN card and GPS receiver.

**3G Connection.** An Internet connection, deployed as a 3G mobile broadband data connection, is used to prepare a drone for use in the SkyNET. This connection is optional, but greatly reduces the amount of work for the botmaster, and potentially allows for greater anonymity

and range of the SkyNET. Anonymity can be preserved if the correct precautions are taken by the botmaster, which are described in section 5.2. During the survey and target procedure the drone utilizes a 3G connection to off-load cryptanalysis attacks. During the first phase of the SkyNET C&C protocol, which occurs during the enlist procedure, the botmaster uses the 3G connection to verify the trustworthiness of the drone’s flight. No connection is used during the second phase.

**GPS Receiver.** To implement the secondary phase of the C&C the drone needs to navigate autonomously. This protects the botmaster if the drone becomes compromised, since at this point the bots may have received botnet C&C details. Using a GPS receiver and sonar sensor the drone records latitude, longitude, and altitude. We used previous work in autonomous flight and navigation [23] to implement these features. This allows the botmaster to launch the drone and configure it to replay a previously recorded path.

**Wi-Fi Card.** This enhancement is specific to our choice of using the AR.Drone. Since the AR.Drone uses an Ad-Hoc Wi-Fi network to receive commands, we require a Wi-Fi card on the controller. This requirement can be replaced with any radio that implements navigation. Some example drones may be capable of running on a signal microcontroller.

**Injection-capable Wi-Fi card.** A secondary Wi-Fi card is required for network attacks and communication with bots. While the first card is connected to the drone’s Ad-Hoc network, the second may be scanning, recording, attacking, creating, or joining Wi-Fi networks. To implement the Caffe Latte attack, this card must support Master mode; to hasten the network attacks in section 3.1, this card must support injection. During the enlist procedure this card implements the communication between the drone and host or bot described in figure 2.

We had a more general requirement for modularity of attached radios. Since the drone functions as a mobile attack center, we can increase the attack capability with additional radios.

## 5.2 Implementation

Our prototype uses a off-the-shelf SBC with an ARM4 250MHz processor. We used Debian Lenny for the operating system. A quad-band Mini-PCI WWAN card was used for the 3G connection and GPS receiver. Two injection-capable Wi-Fi B/G cards were used to connect to the AR.Drone’s Ad-Hoc network and for attacks against Wi-Fi networks. The drone includes a secondary, stronger, GPS receiver. The modifications weigh 196 grams and the AR.Drone was measured and tested to hold a maximum payload of 278 grams. Using the default AR.Drone battery, a 3-cell 1000mAh lithium poly-

Drone	SBC	Flying	1Ah life	1.3Ah life	Amps
T	F	T	12m 31s	16m 06s	4.79A
T	T	F	1h 30m 44s	2h 01m 40s	0.64A
T	T	T	10m 05s	13m 14s	5.89A

Table 1: Battery life averages, including average amperage. Flying indicates whether the test was performed while the drone was in-flight.

mer, the drone can maintain attacks for an average of 2.5 hours. The drone can maintain attacks, while in motion, for an average of 20 minutes.

The drone is piloted through a web interface which shows the controller the output of the AR.Drone’s forward-facing camera, the location via Google Maps, and controls for launching attacks. Piloting the drone over a 3G network allows the controller to position the drone at great distances away from their location. This increases the anonymity of the pilot, but uses an Internet connection. Since the drone is connected to a private 3G network, an Internet proxy must be used for communication. The drone must first negotiate a connection with a proxy before the controller can connect. Additional steps can be taken to preserve anonymity when selecting this proxy. The pilot must make sure they purchase a contract-less network connection for the drone. Although this method of communication is suggested, we maintain that SkyNET can deliver C&C without the use of an Internet connection, simply by separating the duties of the pilot and botmaster.

## 6 Feasibility

The feasibility of this theoretical use of a drone to command botnets depends on multiple factors: the capability to remain unnoticed, the exposure to a significant number of Wi-Fi access points (AP), and a large enough time window to perform exploits and bot controller communication. We show that an average flight for the drone will expose a large amount of Wi-Fi APs, and provide time for exploitation. An average flight for the drone is determined based on average drone speed and maximum available flight time. We present multiple flight paths in an urban area using the speed, time, and power constraints of the prototype drone. Along with each path we include the number of APs that would have been exposed to the drone. The AP totals were collected using the drone’s Wi-Fi card, but not while the drone was in flight. We conclude the section with a brief example of how the drone may remain unnoticed during similar flights.

Table 1 shows the average battery life for three AR.Drone and SBC power configurations. These fig-



Figure 4: The three flight paths, shown on Google Maps, used to test the feasibility of SkyNET. Flight 3 (pointer with a dot) involves two separate paths around proximal parks.

Flight #	Distance	Flight time	Ground time	APs
1	2,756m	13m 14s	0s	1716
2	702m	3m 19s	1h 50s	1122
3a	876m	N/A	N/A	563
3b	74m	N/A	N/A	148

Table 2: Time used in-flight, time remaining for grounded operation, number of APs exposed to the drone; 3a is the larger park in flight 3, 3b is the smaller park.

ures are an average of three tests per-configuration. In all, the battery was allowed to run to 2% capacity. Note that the default setting for an AR.Drone is 10%. The AR.Drone advertises a speed of 5 meters per second, our tests yielded an average of 4.5, again in a controlled environment with low wind resistance. Adding the SBC reduced the average speed to 4m/s over a distance of 25m. Traveling at 4m/s for 794 seconds allows the drone to cover an average maximum theoretical distance of 3.18km. Flight 1 in figure 6 is an example of a scouting flight and covers roughly 2.8km. To demonstrate the feasibility of an exploit flight using the drone in an urban area we used the perimeter of a downtown block in Manhattan, NY, which is roughly 700m shown as flight 2. While attacking networks and hosts, the drone can land to conserve power. While grounded the drone and SBC consume an average of 641mA. If the drone uses one fourth of its battery to position itself, half of its power

to attack while not in flight, and the remaining fourth to return to a pick-up position, the drone will have approximately one hour to exploit.

The AR.Drone UAV is a great example prototype. The popularity and familiarity of the AR.Drone as a toy allows it to fly in parks and recreational areas without suspicion. To measure the feasibility of restricting the drone's flight to city parks we include flight 3a and 3b. We include all exposed Wi-Fi APs for each of the flights in table 2. The number of exposed access points for each of the three flights suggests that using the prototype SkyNET drone is feasible.

## 7 Defending against SkyNET

There are several obvious attacks against SkyNET that involve physical attacks on the drone. In this section we describe a few of these attacks along with mitigation strategies against the enhancement SkyNET adds to botnet C&C. An attack against the drone must begin with the detection of the use of a SkyNET drone. Thus it is less important to understand how to stop the drone, than it is to realize its involvement. We aim to provide a high-level overview of such detection strategies to prompt further research.

Detection of a SkyNET may be possible by observing the behavior of the underlying botnet and discovering the geolocation of the bots. By observing traffic [7], and correlating the location of hosts, analysis may reveal activity originating from a proximal location relative to multiple bots. Physical inspection of the environment nearby the hosts may reveal the use of a SkyNET drone. Defeating the prototype drone is trivial; under adverse weather conditions such as wind and rain, the drone is grounded. Tracing the path of the drone will reveal potential networks that contain bots. Tracing the drone to the point where it is retrieved by botmaster is a potential attack which may reveal the identity of the botmaster. Defense should focus on the detection of side channel C&C delivery. Defense could also take a bottom-up approach, not focusing on revealing the botmaster [19].

Detection of new hosts on home networks can prevent the SkyNET enhancement. It is seldom that home networks experience a new connection, authenticating new and returning hosts is a trivial task for routing equipment. Logging these connections and alerting home administrators may be the first step to mitigation [5].

## 8 Related Work

Some of our research builds on prior work related to home Wi-Fi botnets. Warkitting [28] describes using malicious router firmware and home Wi-Fi to propagate botnet infection by compromising other nearby Wi-Fi

networks. SkyNET uses a UAV drone as an alternative means of constructing this network. Using UAVs to break into wireless networks has been performed previously. The most notable UAVs, designed for 'warflying', are WASP [24] and the *Build-you-own Predator 2.0* [30]. The *Predator 1.0* was designed as a surveillance UAV and later enhanced with wireless penetration capabilities. Like our SkyNET drone, WASP and the *Predator* provide a mobile attack framework including multiple radios and a Linux environment for launching attacks.

Our SkyNET drone differs from these warflying UAVs in that it consumes less power for a comparable usage time, is almost one tenth the size, less than one tenth the weight, and can be operated by any Internet-connected device or pre-programmed before flight. The important improvements are auto-pilot capabilities and reduced footprint, measured by size, weight, and energy. WASP or the *Predator* could be used as SkyNET drones, but the pilot may be trackable during flight.

SkyNET uses local radios to deliver C&C to botnet controllers. This enhances the botnets by adding a layer of protection for the botmaster. Wi-Fi radio was used in the examples we presented. Singh et al. [25] has shown that weaknesses in Bluetooth can also be used as a method to create a botnet. We can apply the same methods to increase the attack capability available to SkyNET. These enhancements to protecting a botmaster are also related to more general enhancements to botnet C&C [13,29].

## 9 Conclusion

In this paper we described an enhancement to botnet C&C using a technique to separate botmaster from a botnet. This technique uses a flying drone to relay information. The drone assembles potential bot controllers by using vulnerabilities in wireless networks, wired hosts, and mobile devices. We created a prototype UAV drone which has the capability of implementing such attacks. We proposed a simple method for communicating from botmaster to bot controller using the drone, without revealing information about the botmaster or network size to an attacker.

This technique uses the insecurity of Wi-Fi networks, mobile devices, and home networks to evade analysis. SkyNET creates a stealth channel for C&C by evading popular network security mechanisms such as network firewalls, intrusion detection systems, and connection/event logging. Proper home network security and transparency of network events is an emerging requirement. By suggesting this enhancement, and demonstrating a prototype drone, we hope to motivate research on securing home networks and creating analysis techniques for detecting side-channel C&C communication.

## References

- [1] Amazon web services, 2011. Amazon Elastic Compute Cloud <http://aws.amazon.com/ec2/>.
- [2] ABU, M., ZARFOSS, R. J., MONROSE, F., AND TERZIS, A. My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging. In *Technical Conference* (2007).
- [3] AIRCRACK-NG. Main documentation, 2011. <http://www.aircrack-ng.org/documentation.html>.
- [4] AURA, T. Strategies against replay attacks. In *Computer Security Foundations Workshop. Proc., 10th* (1997), pp. 59–68.
- [5] CALVERT, K. L., EDWARDS, W. K., FEAMSTER, N., GRINTER, R. E., DENG, Y., AND ZHOU, X. Instrumenting home networks. *SIGCOMM Comput. Commun. Rev.* 41 (2011), 84–89.
- [6] CISCO. Combating botnets using the Cisco ASA botnet traffic filter, 2009. Technical Report.
- [7] COSKUN, B., DIETRICH, S., AND MEMON, N. Friends of an enemy: Identifying local members of peer-to-peer botnets using mutual contacts. In *Proc. of the 26th Annual Computer Security Applications Conference (ACSAC)* (2010).
- [8] DAGON, D., GU, G., LEE, C. P., AND LEE, W. A taxonomy of botnet structures. In *Proc. of the 24th Annual Computer Security Applications Conference (ACSAC)* (2007), pp. 325–339.
- [9] DITTRICH, D., AND DIETRICH, S. Command and control structures in malware: From handler/agent to P2P. In *login: The USENIX Magazine* (December 2008).
- [10] FREILING, F. C., HOLZ, T., AND WICHERSKI, G. Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks. In *Proc. of the 10th European Symposium On Research In Computer Security (ESORICS)* (2005).
- [11] GU, G., YEGNESWARAN, V., PORRAS, P., STOLL, J., AND LEE, W. Active botnet probing to identify obscure command and control channels. In *Proc. of the 25th Annual Computer Security Applications Conference (ACSAC)* (2009).
- [12] HOUSLEY, R., AND ARBAUGH, W. Security problems in 802.11-based networks. *Commun. ACM* 46 (2003), 31–34.
- [13] HUND, R., HAMANN, M., AND HOLZ, T. Towards next-generation botnets. In *Proc. of the 4th European Conference on Computer Network Defense (EC2ND)* (2008), pp. 33–40.
- [14] JUSTICE.GOV. "Botherder" dealt record prison sentence for selling and spreading malicious computer code, 2006. <http://www.justice.gov/criminal/cybercrime/anchetaSent.htm>.
- [15] KARTALTEPE, E. J., MORALES, J. A., XU, S., AND SANDHU, R. Social network-based botnet command-and-control: emerging threats and countermeasures. In *Proc. of the 8th international conference on Applied cryptography and network security (ACNS)* (2010), pp. 511–528.
- [16] KSHETRI, N. The simple economics of cybercrimes. *Security Privacy, IEEE* 4, 1 (2006), 33–39.
- [17] LEHEMBRE, G. Wi-Fi security - WEP, WPA and WPA2. *Hackin9* (January 2006).
- [18] LUEG, L. Pyrit, 2011. <https://code.google.com/p/pyrit/>.
- [19] ORMEROD, T., WANG, L., DEBBABI, M., YOUSSEF, A., BINSALLEEH, H., BOUKHTOUTA, A., AND SINHA, P. Defaming botnet toolkits: A bottom-up approach to mitigating the threat. In *Proc. of the 4th International Conference on Emerging Security Information Systems and Technologies (SECURWARE)* (2010).
- [20] PARROT. AR.Drone the flying video game, 2011. <http://ardrone.parrot.com>.
- [21] RAMACHANDRAN, V., AND AHMAD, M. S. Caffe latte with a free topping of cracked WEP, 2007.
- [22] RAPID7. Metasploit Framework, 2011. <http://www.metasploit.com/>.
- [23] RAY, A., BEHERA, L., AND JAMSHIDI, M. GPS and sonar-based area mapping and navigation by mobile robots. In *Proc. of the 7th IEEE International Conference on Industrial Informatics (INDIN)* (2009).
- [24] REDQUEEN, AND WHITEQUEEN. Wi-Fi aerial surveillance platform W.A.S.P., 2011. <http://rabbit-hole.org/>.
- [25] SINGH, K., SANGAL, S., JAIN, N., TRAYNOR, P., AND LEE, W. Evaluating bluetooth as a medium for botnet command and control. In *Proc. of the 7th Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)* (2010), pp. 61–80.
- [26] STONE-GROSS, B., COVA, M., CAVALLARO, L., GILBERT, B., SZYDLOWSKI, M., KEMMERER, R., KRUEGEL, C., AND VIGNA, G. Your botnet is my botnet: analysis of a botnet takeover. In *Proc. of the 16th ACM conference on Computer and communications security (CCS)* (2009), pp. 635–647.
- [27] TEWS, E., AND BECK, M. Practical attacks against WEP and WPA. In *Proc. of the 2nd ACM conference on Wireless network security (WiSec)* (2009), pp. 79–86.
- [28] TSOW, A., JAKOBSSON, M., YANG, L., AND WETZEL, S. Warkitting: the drive-by subversion of wireless home routers, 2006.
- [29] WANG, P., SPARKS, S., AND ZOU, C. C. An advanced hybrid peer-to-peer botnet. In *Proc. of the 1st conference on Hot Topics in Understanding Botnets (HOTBOTS)* (2007).
- [30] WEIGAND, M., HAINES, B., AND KERSHAW, M. Build your own UAV 2.0 - wireless mayhem from the heavens!, 2010. Defcon 18, <https://www.defcon.org/html/defcon-18/dc-18-speakers.html#Weigand>.

## Notes

<sup>1</sup>We selected the most popular YouTube video during the month of March 2011, <http://www.youtube.com/watch?v=CD2LRR0pph0>