Roving Bugnet:
Distributed Surveillance Threat and Mitigation

Ryan Farley and Xinyuan Wang

{rfarley3,xwangc}@gmu.edu

GEORGE MASON UNIVERSITY

Department of Computer Science

Fairfax, Virginia, USA

●First ●Prev ●Next ●Last ●Go Back ●Full Screen ●Close ●Quit

Abstract

- Mobile devices make an ideal surveillance network
  - Increasingly always-on
  - Contain built-in microphones and cameras
- To explore this we present a modernized mic hijacker called **bugbot**
  - Controlled over a botnet called **bugnet**
  - Compatible with Windows and Mac OS X
- No surveillance-resource control mechanisms exist, so we introduce
  - A way to detect and protect against such an attack
  - A way to deceive the attacker and facilitate traceback

1. Introduction

- Surveillance spyware protection is a missing segment of privacy control
  - Most devices do not have physical kill switches
  - Leaves highly personal data vulnerable
  - Is a growing concern as exploitable devices become more pervasive
- Unanswered high consequence and universal threat
  - A microphone in every house is not that useful
  - Surveillance attacks will probably involve pre-specified targets
  - We're all capable of gaining an unwanted stalker or jealous spouse

1.1. Roving Bug

- Most plausible use is to create a **roving bug**
  - Surveillance that follows individual, not device
  - Implies a coordinated group of devices
  - The more devices, then the greater the capacity to monitor the victim
- To demonstrate, we have developed the **roving bugnet**
  - IRC botnet with microphone surveillance bots as nodes
  - Runs on Windows (95–Vista) and Mac OS X
  - Can seize control without interaction from the victim

Roving Bug, at Home

Roving Bug, on the Move

Roving Bug, at Work

Roving Bug, Coffee Shop Example

Roving Bug, Conference Room Example

1.2. Protection

- No existing defense
- To resolve this we present a preliminary mitigation mechanism
  - Can detect active use of the microphone
  - Includes a novel method to deceive a remote attacker after detection

2. Bugnet Design

- Two functional components
  - Microphone hijacker, the bug
  - Remote control, the bot
- Features:
  - Infect Internet connected hosts without victim's interaction
  - Bug can be turned on at arbitrary times or at predefined system conditions
  - Can record or stream indefinitely or for a specified duration

2.1. Bug Program

- Divided into two threads
  - Control thread starts and stops recording
    - Can use stdio, UDP server, or a more covert channel
    - Can detect if network dies and record to file until connection is restored
  - Data thread handles recorded audio from sound card
    - Creates a cyclical array of static length buffers
    - Sound card driver uses buffers to store audio data
    - Driver sends the data thread a message once a buffer is filled
    - Data thread outputs data and reinserts buffer into array

2.2. Bugbot Node

- Standard IRC bot
  - Connects to predetermined IRC server and channel
  - Waits for botmaster nick, and requires a password
- Windows version is in C, OS X version in Perl to support PPC and Intel
- Has basic set of commands
  - Self installation routine
  - Kill itself and erase existence
  - DCC file transfer handling
  - Run an arbitrary command at an arbitrary time

Threat Demonstration

play video of 1) bot logging on and 2) forwarding the mic audio to the bt4 vm

3. Detection and Mitigation

- Could use physical kill switch or cover
  - Difficult after-market option
- Fortunately, software based mic access filters are a low burden to users
  - Unlike network access requests or prompts for privilege escalation
  - Frequency of legitimate requests should be very low
    - Harder to hide in a cluster of legit requests
  - Even low tech users understand what a mic does and when it should be on or off

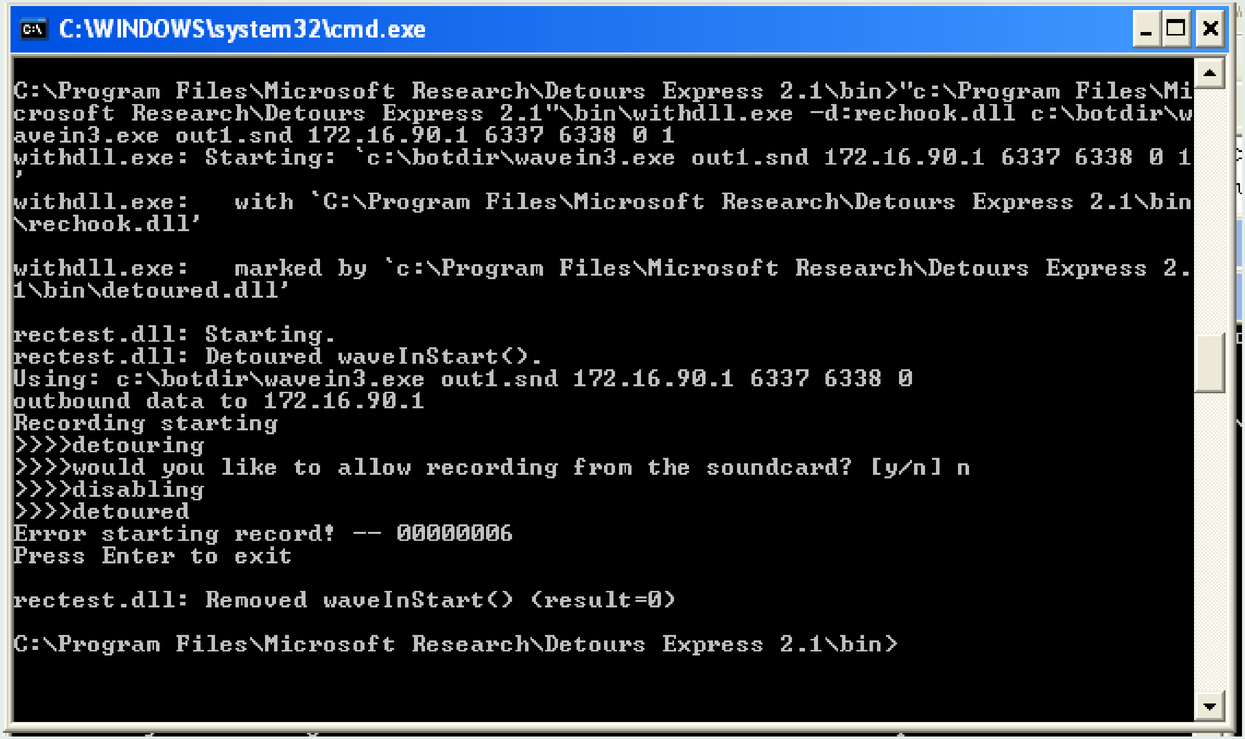
3.1. Detours

- Our method uses API call monitoring with Detours
  - Transparent access to all arguments and return values
  - Provides specification based intrusion detection
  - User can set access controls or be prompted at each request
  - Can completely deny a request or meddle with data passed back to a blocked application

3.2. Deploying the Protection Mechanism

- Catches call to initialize sound recording buffers
  - Happens before call to start recording
  - Bug fails to reach state capable of gathering data
- Automating process of deciding if a process should be trusted or untrusted is difficult
  - Best method is to prompt the user
- This method would be obvious to the attacker, more effective method may be through misinformation

Protection Mechanism in Action



```
C:\Program Files\Microsoft Research\Detours Express 2.1\bin>"c:\Program Files\Mi
crosoft Research\Detours Express 2.1"\bin\withdll.exe -d:rechook.dll c:\botdir\w
avein3.exe out1.snd 172.16.90.1 6337 6338 0 1
withdll.exe: Starting: `c:\botdir\wavein3.exe out1.snd 172.16.90.1 6337 6338 0 1
'
withdll.exe:   with `C:\Program Files\Microsoft Research\Detours Express 2.1\bin
\rechook.dll'

withdll.exe:   marked by `c:\Program Files\Microsoft Research\Detours Express 2.
1\bin\detoured.dll'

rectest.dll: Starting.
rectest.dll: Detoured waveInStart().
Using: c:\botdir\wavein3.exe out1.snd 172.16.90.1 6337 6338 0
outbound data to 172.16.90.1
Recording starting
>>>>detouring
>>>>would you like to allow recording from the soundcard? [y/n] n
>>>>disabling
>>>>detoured
Error starting record! -- 00000006
Press Enter to exit

rectest.dll: Removed waveInStart() (result=0)

C:\Program Files\Microsoft Research\Detours Express 2.1\bin>
```

3.3. Deception by Decoy Audio

- Feeding the bug crafted data can extend attacker's connection time
  - Provides better audit trail while still protecting microphone
  - Works even if data is exported through some yet undiscovered covert channel
  - Control of data streamed to attacker facilitates traceback
- Sound should be believable
  - Background chatter or keyboard clacking
- How it works
  - Signal from sound card that indicates a full buffer is intercepted
  - Decoy data is written over buffer pointed to by signal
  - Bug receives signal and is unaware of modified buffer

4. Discussion

- Current systems allow multiple users to access microphone simultaneously, regardless of physical presence
  - Threat increased if untrusted users on same system
  - Imagine a jealous spouse on a shared home computer
- Bugnet produces large data set (compared to keystrokes or online credentials)
  - Still scalable, could be integrated into existing wiretapping techniques
- Remote surveillance is more personal of an attack than identity theft
  - Which is worse?

5. Conclusion

- Universal threat with rapid growth of potential
- To demonstrate the viability we developed a modernized stealthy mic hijack threat
  - Features closely match in-the-wild exploits
  - Uses a botnet framework
- We then presented a way to mitigate the threat, giving user unobtrusive control
  - Provides allow, deny, or deceive behavior
  - Facilitates forensic analysis

As awareness of this problem increases, the potential threat to privacy may lead consumers and businesses to lessen their dependence on vulnerable devices

6. That's All. . .

- Unanswered questions? Comments?
- Afterwards feel free to
  - Contact me at `rfarley3@gmu.edu`
  - Find me at `ryanfarley.net`
- Thank you for your time