BugBox

A corpus, a testbed, a framework, an aid to research

Gary Nilson, Kent Wills, Jeff Stuckman, Jim Purtilo
Vulnerability data facilitating empirical security research

• Studying past vulnerabilities may provide insight toward detecting or preventing future ones

• Our current research requires well-structured vulnerability data
  • Metrics characterizing aspects of code that are uniquely associated with vulnerabilities
  • New types of vulnerability signatures
  • Examining relationship between features and vulnerability introduction

• Need: A collection of vulnerable applications with clearly identified vulnerabilities and metadata, running in a test environment facilitating reproducible exploits
What is BugBox?

BugBox is a corpus of web application vulnerabilities and associated exploits within a testbed, allowing for repeatable experiments and automated data collection.

- Application and Exploit Modules
- Sandbox for controlled experiments
- A scriptable framework for running security experiments
Application Modules

- WordPress
  - Exploit 1
  - Exploit 2
  - ...

- cute:flow
  - Exploit 1
  - Exploit 2
  - ...

BugBox

- Experimental Instrument
- Run Exploit

- Recorded data

- Reusable modules
- Automated data collection
- Easily developed exploits
Corpus Structure

• How is BugBox a corpus?
• Benefits of a public corpus.
• What data does the corpus contain?
  • Exploit module
  • PHP web application module
  • Instrumented Data
Essential qualities of our corpus

• Linguistics
  • A large structured collection used to perform statistical analysis and software testing.

• Facilitates replications of experiments
• Representative examples of the most common exploit classes
• Quality controlled data
• Metadata
A testbed for experiments

- Large-scale automated security experiments
- Query and run exploits by metadata
- A fresh application environment is created for each exploit
- The testing is contained in a sandbox for exploit - application interaction
- Exploit script
A framework for quick module development

• Selenium Browser automation framework
  • Javascript support
  • Live display capability

• Engine
  • Automatic application installation and configuration
  • Old applications with dependencies

• Application Module
  • Reuse

• Exploit Module
def exploit(self):
    payload = "<script>prompt("Attack!!\")</script>"
    driver = self.create_selenium_driver()
    driver.get("http://localhost/wordpress")
    self.logger.info("Page opened successfully: %s", driver.title)
    self.logger.info("Uploading payload: %s", payload)
    driver.get_element(by_xpath="//input[contains(@id,'sk_alias')]").clear()
    driver.get_element(by_xpath="//input[contains(@id,'sk_alias')]").send_keys("John Doe")
    driver.get_element(by_xpath="//input[contains(@id,'sk_email')]").clear()
    driver.get_element(by_xpath="//input[contains(@id,'sk_email')]").send_keys("johndoe@aol.com")
    driver.get_element(by_xpath="//input[contains(@id,'sk_text')]").clear()
    driver.get_element(by_xpath="//input[contains(@id,'sk_text')]").send_keys(payload)
    driver.get_element(by_xpath="//input[contains(@id,'sk_text')]").send_keys("\n")
    driver.get_element(by_class="sk-userdata-user")
time.sleep(10)
    driver.cleanup()
    return
BugBox applications

• Simulating malicious activities
  • Test
  • Intrusion Detection System (IDS)
  • Intrusion Prevention System (IPS)

• Gauging effectiveness of software hardening techniques

• Testing and training static analysis tools and vulnerability predictive models

• Education and Demonstration
This exploit utilizes xterm to login as an administrator, extract the \_file \_include file, and then use it to write the payload. The payload changes the title for the first past.

```bash

```
Lessons Learned

• Summer Undergraduate Research
• Developing exploit scripts accelerated the corpus development process
  • Vulnerabilities became self documenting
  • Structured development tasks
• Quality control methods are helpful
  • Verification
  • Validation
  • Testing
• Live playback of exploits speeds up development
Conclusion

BugBox is a corpus of web application vulnerabilities and associated exploits within a testbed, allowing for repeatable experiments and automated data collection.

• 71 Exploits available and development is continuing
• Download @ http://seam.cs.umd.edu/projects/BugBox
  • Debian package
  • GitHub repository
• Contribution of BugBox modules are welcome!
Questions?

Availability:
http://seam.cs.umd.edu/projects/BugBox

A corpus, a testbed, a framework, an aid to research.