

# Zero Day Zen Garden: Windows Exploit Development – Part 0 [Dev Setup & Advice]

Aug 11, 2017 • Steven Patterson



Welcome to the first post in a series of tutorials on exploit development and vulnerability discovery! I've spent a great deal of time studying this area of computer security and I'm eager to share with others what I have learned. In doing so, I hope that I can gain a better understanding of these subjects while also helping others who are new to the wild world of exploit development. This post will go through the development environment setup you'll need to perform to follow along in Part 1 next week and general tips for newbies. So without further ado, let's get started!

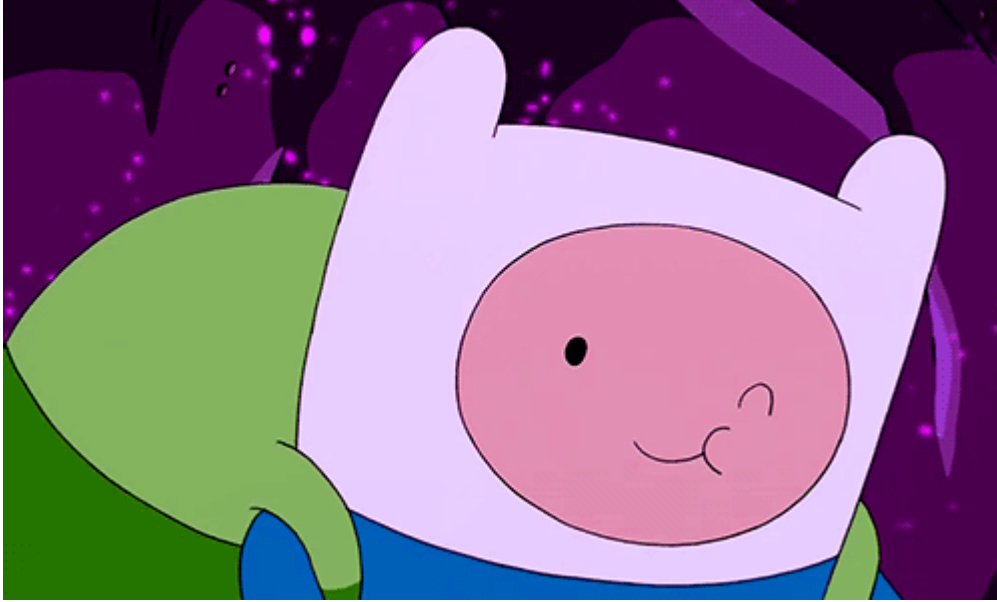


## Getting things setup

The first set of tutorial posts will be dealing with exploit development on the Windows platform. We want to start small and slowly work our way up towards more complex exploits on systems with greater degrees of mitigations, so we'll begin with simple buffer overflow exploits on Windows XP (32-bit). To get prepared, you'll need to setup a virtual environment for testing our exploits:

1. [Get Windows XP SP3 32-bit running on a virtual machine \(VirtualBox\)](#) with a Windows XP ISO file.
2. [Install Immunity Debugger](#) on the virtual machine
3. [Install a code editor \(like Sublime Text 3\)](#) on the virtual machine

After you've completed those 3 steps, you will be ready to tackle the upcoming tutorials with a glorious virtual environment. Awesome job!



## General tips and advice for beginners

Before starting, some brief advice for you that I wish I had when I began exploit development.

- **Understand the basics first before moving on to more advanced subjects.**
  - The old adage of “walk before you run” applies in exploit development to a great degree. Recognize that you will need to understand the basics of assembly language, computer architecture, operating systems and memory management if you want to fully comprehend computer exploitation.
  - Even after gaining this understanding, don't skip straight to ROP chains and heap overflows in modern browsers. Start with simple exploits on toy examples before moving on. Your primary goal at this stage is to establish a strong foundation of understanding from which you will build upon.
  - If you ignore this advice and presume you know how everything works already, you will be in for a rude awakening and suffer through many frustrating hours of exploits that refuse to run. Be warned.
- **Don't mistake knowing for doing**
  - Early on, you will likely be seeking out many varied resources to learn from. This is good and I strongly encourage reading books on exploit development as well as reverse engineering. However, be careful that you don't pour all of your time into reading. It is tempting to spend entire days researching exploit development while you forget about DOING exploit development. Knowledge that isn't applied is wasted, maintain balance between studying and practice.
  - A few examples of things that allow you to apply your knowledge include doing **CTFs**, **exploiting toy programs**, **reversing challenges**, **analyzing malware**, writing exploit development tools or scripts, exploit development writeups or tutorials (like this one!) and **finding exploits in the wild**.
- **Take breaks and stick your head out into life**
  - You will be infinitely more productive if you schedule regular breaks into your work schedule. Do something that gets your mind away from work on the computer so you can digest what you are trying to master. Personally, I like to lie on the couch for 10-15 minutes listening to

music on headphones for short breaks or taking walks around my neighborhood for long breaks.

- Also, sleep. Regular sleep will turn a 2 hour problem into a 30 minute one.

- **Divide and conquer**

- Much of the work involved in exploit development is about dividing a problem up and slowly obtaining a solution with each minor problem solved. When faced with a large, complex piece of software with hundreds of moving parts, ease into development by defining the problem and breaking it down into smaller chunks. I like to remember the following “Q: How do you eat an elephant? A: One bite at a time”.

## Closing thoughts & Part 1 coming soon

I hope you are as excited as I am to learn about exploit development and vulnerability discovery. I’m looking forward to giving back to the community that has been so kind to me. Since beginning my own journey into this field, I’ve discovered how interesting and challenging the work can be. I sincerely enjoy the time I spend researching vulnerabilities and crafting an exploit, each piece of software is like a tiny universe with secrets that haven’t been uncovered yet. Perhaps some of you will discover a similar enjoyment.



Fair warning though, I’ll be presuming you have some level of basic computer science knowledge. I won’t be explaining in detail assembly language or how the stack works, I’m going to assume you understand these things already. If you currently lack this knowledge, no worries! You can see a list of resources at the end of this post to read up on/watch before coming back to this set of tutorials. See you for **Part 1** next week, you can subscribe to the **RSS feed** if you want to be notified right away.

頑張って!

**UPDATE: Part 1 is posted [here](#).**

## Fundamental resources & topics

### x86 Assembly Language

- <https://www.cs.virginia.edu/~evans/cs216/guides/x86.html>

### C Function Call Conventions and the Stack

- <https://www.csee.umbc.edu/~chang/cs313.s02/stack.shtml>

## Intro to x86: Architecture, Assembly, Applications

- <https://www.youtube.com/playlist?list=PL038BE01D3BAEFDB0>



## Good books to read

- [Assembly Language Step-by-Step: Programming with Linux](#)
- [Reversing: Secrets of Reverse Engineering](#)
- [The Shellcoder's Handbook: Discovering and Exploiting Security Holes](#)
- [Hacking : The Art of Exploitation](#)

---

## Shogun Lab | 将軍ラボ

Shogun Lab | 将軍ラボ  
[steven@shogunlab.com](mailto:steven@shogunlab.com)

 [shogunlab](#)  
 [shogunlab](#)  
 [shogun\\_lab](#)

Shogun Lab does application vulnerability research to help organizations identify flaws in their software before malicious hackers do.

The Shogun Lab logo is under a [CC Attribution-NonCommercial-NoDerivatives 4.0 International License](#) by Steven Patterson and is a derivative of "Samurai" by Simon Child, under a [CC Attribution 3.0 U.S. License](#).