Redefining Attack Surface Measurement

A LangSec Perspective

F. Maldonado[1] M. Levy[1]

[1]Scientist
Naval Information Warfare Center, Pacific

LangSec, May 2024

Definition of Attack Surface

- The sum of vulnerabilities, pathways, or methods—sometimes called attack vectors—that hackers can use to gain unauthorized access to the network or sensitive data or to carry out a cyberattack. [2]
- The set of points on the boundary of a system, a system component, or an environment where an attacker can try to enter, cause an effect on, or extract data from that system, component, or environment. [6]

The Good

Framing an attack surface in terms of what is made explicit, e.g. entry and exit points, contributes to discourse on making systems safer.

- What can I do to reduce my attack surface, and thus, reduce risk.
- Why is this particular system configuration unsafe?

The Bad

- This illustrates the difficulty in understanding attack surfaces to measure them accurately. This might also explain why attack surface measurement research is reduced to entry and exit points [3, 5, 4].
- Given the myriad ways an attacker can enter, affect, escalate privileges, and extract data from a system or environment, we contend an attack surface is much more than what is described above, and accordingly, new forms of thinking are needed to measure it.

Gaps

While the framework works for measuring flaws at the system level of an application it lacks language to describe security flaws within an application's boundary.

- How can we describe ROP attacks?
- How can we measure the attack surface reduction due to hardening?
- How can we measure the complexity of exploitative inputs?

Weird States

Weird States in a program are defined to be states that are not present in the IFSM but are in the implementation. We will be using Q^{cpu}_{weird}. These states are reachable through the set of exploitative strings.

Exploitative Strings

Exploitative strings are defined to be the set of strings in the input language of a program that take a normal state to a weird one. The set will be referenced as E. Note that E is a subset of the set of input strings L.

Redefining Attack Surface

Let's define the attack surface of an application to be the tuple of weird states and the set of exploitative strings, (E, Q^{cpu}_{weird}).

Redefining Attack Surface

Having the tuple (E, Q^{cpu}_{weird}) leads to some interesting avenues,

- Can compare the size of weird states between programs.
- Can compare the "size" of exploitative strings.
- Gives a framework to proof security properties of applications.

Example

Program of interest,

```
int main(int argc, char **argv)
{
 printf("Hello! What's your name?\n")
 char usr_input[8];
 gets(usr_input);
 printf("Hello %s!\n", usr_input)
}
```

Now assume we compile it twice,

- Binary n
- Binary m with DEP/NX enabled

Sanity Check

We know that the attack surface of binary m is lower than n, so it would make sense if the exploit string constructed to be more "complex". Let's work on that...

Injecting code into n

In order to inject code into n, we have to feed the following input $s_n = w_p + w_{rip} + w_{addr} + w_{nop} + w_{sh}$,

- w_p, padding required to initiate buffer overflow
- w_{rip}, padding to overwrite rip
- w_{addr}, address we want to change control flow to
- w_{nop}, nop slide
- w_{sh}, shell-script to run exploit

Injecting code into m

Since the stack is now marked as non-executable space, we need to disable that feature by calling VirtualAlloc(),

$s_m = w_p + w_{rip} + w_{VA} + w_{addr} + w_{nop} + w_{sh}$

- w_p, padding required to initiate buffer overflow
- w_{rip}, padding to overwrite rip
- w_{VA}, ROP gadget collection to disable VirtualAlloc
- w_{addr}, ROP gadget collection to change control flow
- w_{nop}, nop slide
- w_{sh}, shell-script to run exploit

That's neat...

While the example is rather trivial, it shows how to write exploits in this sort of format to express the difficulty in achieving an advantageous state.

- Lots of blog post explaining exploits are shown like this [1]
- Beginnings of being able to compare "complexity" of attack vectors
- Potential to expand framework outside of binaries
- Easy to understand
- topological definition of attack surface

Contact

- Froylan Maldonado : froylan.g.maldonado.civ@us.navy.mil
- Matthew Levy : matthew.l.levy.civ@us.navy.mil

bibliography I

[1] Ahmed (0xRick) Hesham. *Buffer Overflow Examples, Code execution by shellcode injection - protostar stack5*. https://0xrick.github.io/binary-exploitation/bof5/. Accessed: 2024-01-10. 2019.

[2] Inc. IBM Systems. *What is an attack surface?* https://www.ibm.com/topics/attack-surface. Accessed: 2024-01-10. 2014.

[3] Pratyusa Manadhata and Jeannette Marie Wing. *Measuring a system's attack surface*. Pittsburgh, PA, USA: School of Computer Science, Carnegie Mellon University, 2004.

[4] Pratyusa K Manadhata and Jeannette M Wing. "An Attack Surface Metric". In: *IEEE Transactions on Software Engineering* 37.3 (2010), pp. 371–386.

bibliography II

[5] Pratyusa K Manadhata et al. *An approach to measuring a system's attack surface*. Tech. rep. Carnegie Mellon University, 2007.

[6] National Institute of Standards and Technology. *Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations*. Tech. rep. NIST Special Publication 800-171, Revision 2. Washington, D.C.: U.S. Department of Commerce, 2020. DOI: 10.6028/NIST.SP.800-171r2.