Using Modular Arithmetic to Reliably Encode Authentication Information Within Messages Consisting of Port Connection Sequences

Patrick F. Wilbur - *wilburpf@clarkson.edu*

Department of Mathematics
& Computer Science,

Clarkson University

Background

- Even firms operating “secured” networked services on systems/servers are not immune from attacks, including those that authenticate users and encrypt communications.

  Services may include: web servers, file servers, terminal servers, etc.

  Certain **software, system,** and **social weaknesses** can sometimes still be exploited when such vulnerabilities exist and are discovered by an attacker.

- A **pre-authenticating firewall** can be established that protects important services from outside access until some form of pre-authentication occurs successfully.

  We define **pre-authentication** as authorization checking that takes place prior to giving a connection attempt access to a service and that service's existing authentication mechanism(s).

- By having no access to services—even protected ones—whatsoever, an attacker cannot exploit **security flaws in service design**.

  Also, by lacking information about which services are installed and running, an attacker cannot use technical knowledge about a firm and its systems for an advantage during **social engineering**.

  Social engineering is an act of tricking individuals, where such knowledge would be used to exploit flaws in company information policies and cause employees to make risky mistakes, possibly leaking even further information.

  (For as much fun as social engineering might sound,
  please do not try this at home!)

Problem

- Even a firewall implementing pre-authentication might have weaknesses like any other service!

  If the firewall simply opens a visible control port where individuals connect, pass authentication information (user name, password, etc.) and receive approval and are given access to a service and its conventional authentication mechanism, is the service really any more secure?

- By opening up a visible port for individuals to connect to and become pre-authenticated, the firewall's pre-authentication mechanism is operating the same exact way an authentication mechanism of a typical service operates.

  We already defined this to be problematic in the case of a service, and this problem similarly extends to the firewall itself.

  - The firewall implementation may contain vulnerabilities in its pre-authentication/access-control mechanism that could be exploited

  - Having a visible, open port divulges the existence of the pre-authentication mechanism and its specific implementation, which will allow an attacker to search for known vulnerabilities and direct attacks specifically at that implementation

Solution

- It is possible for the pre-authenticating firewall to not open a control port and instead invisibly spy on invalid connection attempts to closed ports.

  The sequence of closed port numbers (in chronological order) that an individual attempted to make connections to can be treated as a secret message being passed to the server discretely. This is a new concept known as **port knocking**.

  This message may be:

  - A complicated string containing information such as a user name and the service the user is requesting access to, and the entire string may be encrypted using public-private key encryption for added security

  - A simple sequence that acts as a combination lock
    (which we will examine for simplicity)

- Unfortunately, the protocols used to route and transfer such connection attempts over networks and the Internet **do not guarantee us that a sequence of multiple connection attempts will be received in the order it was sent!**

  This is a cutting-edge solution—and, quite frankly, the Internet's various protocols were never designed for what we are proposing.

Preserving Order

An off-beat knock will be interpreted as an incorrect combination—how could we possibly accept (1,3,2,4) if the combination is actually (1,2,3,4)?

- We ensure order preservation of the elements of our code by segmenting specific ranges of ports for certain digits.

  On the client:

  Let d = *total number of digits possible within the language of our combination*
  Let C = *the sequence representing our combination message*
  Let P = *the set of ports representing our combination*

  The idea is that we want P to contain our sequence encoded in such a way that the firewall can infer the order of its elements.

  For every i where $0 < i < length(C)$:

  $$P_i = C_i + (d * i)$$

- For example, if we allow our combination to consist of the digits 0 through 9 and our combination is 1-2-3-4, then we have:

  $$d = 10, \quad C = (1,2,3,4)$$

  And P evaluates to:

  $$P = \{1,12,23,34\}$$

Inferring Order

- On the firewall side, given a set of ports P that contained incoming connection attempts, we use the following algorithm to decode the value of the digit element and the index of where it should be placed in our combination sequence C :

  For every j where $0 < j < length(P)$:

  $$C_{P_j \div d} = P_j \bmod d$$

  The idea is that, by reversing our encoding algorithm, we observe two things:

  - The location that the digit contained within P_j belongs inside of C is equal to $P_j \div d$ since we made **divisions** within the acceptable range of port numbers for each digit (where these divisions have a width of d).

  - The value of the digit at that location is represented by P_j when counting in *modulo d.* The values of the digits remain in this message in the form of **remainders** because of how we added them to the multiples of d that we computed in order to generate our port range divisions.

- Working with our previous example, the ports that incoming connection attempts were received on are represented by the set $P=\{1,12,23,34\}$, where $d=10$:

$$C_{P_0 \div 10} = P_0 \, mod \, 10 \Rightarrow C_{1 \div 10} = 1 \, mod \, 10 \Rightarrow C_0 = 1$$
$$C_{P_1 \div 10} = P_1 \, mod \, 10 \Rightarrow C_{12 \div 10} = 12 \, mod \, 10 \Rightarrow C_1 = 2$$
$$C_{P_2 \div 10} = P_2 \, mod \, 10 \Rightarrow C_{23 \div 10} = 23 \, mod \, 10 \Rightarrow C_2 = 3$$
$$C_{P_3 \div 10} = P_3 \, mod \, 10 \Rightarrow C_{34 \div 10} = 34 \, mod \, 10 \Rightarrow C_3 = 4$$

We have $C=(1,2,3,4)$ which was the combination intended to be sent by the individual to the firewall.

Similarly, if we reorder P :

$$P=\{23,1,34,12\}$$

$$C_{P_0 \div 10} = P_0 \, mod \, 10 \Rightarrow C_{23 \div 10} = 23 \, mod \, 10 \Rightarrow C_2 = 3$$
$$C_{P_1 \div 10} = P_1 \, mod \, 10 \Rightarrow C_{1 \div 10} = 1 \, mod \, 10 \Rightarrow C_0 = 1$$
$$C_{P_2 \div 10} = P_2 \, mod \, 10 \Rightarrow C_{34 \div 10} = 34 \, mod \, 10 \Rightarrow C_3 = 4$$
$$C_{P_3 \div 10} = P_3 \, mod \, 10 \Rightarrow C_{12 \div 10} = 12 \, mod \, 10 \Rightarrow C_1 = 2$$

Which gives us the exact same combination sequence $C=(1,2,3,4)$.
The combination's order was preserved.

Conclusion

- We were able to implement a scheme for passing messages from an individual to a pre-authenticating firewall in a form that allowed the firewall to recognize the digits in the combination and infer the intended order of the combination.
- Modular counting may very well be useful in other areas of networking as a reliable replacement protocol for ensuring that the recipient of a long-winded message will infer proper packet order.

For More Information

- These slides, port knocking project notes, and source code for the implementation discussed today:

  http://www.clarkson.edu/~wilburpf/portknocking/

- Port knocking, in general:

  http://www.portknocking.org