ZCSC790 Midterm Take Home Exam

Shuowen Wei

1.

A. “Instant messaging”, “distributed presence” and “transfer from one machine running WASTE to another”, those three features would be useful for collecting passwords typed in by the victims. The bot-master can exploit “distributed presence” function to monitor other compromised machines, and sniff the user inputs, as long as get some “interesting key sequences” it can transfer to that machines running WASTE and send “instant messages” to the bot-master, which is the center collection point.

B. If using SNORT, since the bot-master is the center collection point, and will frequently transfer to other compromised machines to send “instant message” back, thus SNORT should be identify those kinds of actions, especially when these actions are token from the same IP source (i.e, the center bot-master).

2.

When a policy’s accept sets is very big and the policy requires all the arrival data must meet be accepted by each rule in the accept sets, then function parallel firewall’s distributed rules will do no help to speedup, while an equivalent data parallel design will perform way better since each machines is only processing 1/m of arrival data.

3.

A. Since a class-C addressing scheme was used, thus and address space is relatively small. If the attacker is given intelligence of the defense, the attacker can keep track of its attacking history and make some predictions on which address might be changed at what time, thus the attacker can save some effort and be more efficient. Especially when the attacker is given full intelligence on the mechanism (experiment 7), the whole network is just as STATIC to the attacker.

B. No, I don't think so. Since during experiment 6 and7, the target and the network are still same with that of experiment 4 and 5, thus the attackers will exploit their acknowledge of the targets, those experiments are not completely independent mutually.

C. That is because not only the full intelligence on the mechanism is given to attacker, what’s more important is that the encryption on the plain texts is disabled, thus it saved the attacker of decryption, which I think is the most time-consuming during the attacking, and that’s also the purpose of experiment 8, to gain some insight on the importance of data encryption, hence experiment have the lowest attack time.

4.

A. He can use “Hitlist” method to develop the worm, with the Hitlist not including all the n ISC sensors’ IP address (since they are fixed), and the worm can always start his search on the first IP address on the hitlist, and half the hitlist whenever the worm compromise one more vulnerable machine, and pass the 2nd half of the hitlist to the affected machine, in the way we can assure that each machine in the network will be scanned only once, and at the same time and the worm is not be detected by the ISC.

B. We will develop a mimic “binary search” algorithm, it will work even though there is only one ISC sensor in the network. To make calculation simpler, we assume all the address in the network are available to hosts (including 255.255.255.255 and 0.0.0.0). Treat these many addresses as an interval . We can half this interval every day in the way below:

Day 1, use the bot army to send 100 scans to each address from , and send nothing to those addresses in . Watch the ISC’s daily report, if it reports 100 attempts, then we will use the interval for the next day, otherwise we will use the interval .

Day 2, WLOG, assume more than 50 attempts was reported from the previous day, now we will use the bot army to send 200 scans to each address from , and send nothing to addresses in , and watch the ISC’s daily report again and decide which interval to use in the next day.

…

We will keep doing this in the next 30 days and eventually we will find the only one ISC sensor essentially, when there are more sensors, it will take less than 32 days.

5.

The Confinker binary use the botmaster public key to encrypt all the “useful date” it collected, and only the botmaster, who is the only one has the private key, and decrypt the “fruit” and read the data.

6.

A. Since row 4, 5 are dominated by row 3, highlighted in yellow in the matrix low,

Thus we have

Since column 1 is dominated by column 2, and column 3 is dominated by column 4, highlighted in yellow in the matrix above, thus we have

Now, in the new matrix above, since row 2 is dominated by the last row, and at the same time, column 2 and 3 are dominated by the last column, both highlighted in yellow, thus we have

Now, in the new matrix above, since row 3 is dominated by the row 2, highlighted in yellow, thus we have the best strategy as follows:

B.

Both of their choices fall in the best strategy matrix, which shows both sides are well aware of war situations. Based on history we all know that most of Germen troops are not in Normandy, they were probably in disadvantage in defense, thus instead of risking 37 to the Allies, they choose second column to their benefit. While Allies had a lot of preparations before the invasion, instead of choose row 6 and risking getting the lowest score 22, they choose row 1, and they won.

7.

Offet: it allows the rule writer to specify where to start searching for a pattern within a packet;

Depth: it allows the rule writer to specify how far into a packet Snort should search for the specified pattern from a given offset;

Distance: it allows the rule writer to specify how far into a packet Snort should ignore before starting to search for the specified pattern relative to the end of the previous pattern match;

Within: it is a content modifier that makes sure that at most N bytes are between pattern matches using the content keyword

Nocase: it is for the content match specified, do not pay attention to case sensitivity

Noseholes: I didn’t find the definition for this modifier, not quite Internet friendly.

Example:

alert tcp any any -> any 80 (content:"abcde"; nocase; offset:5 depth:15; content:"fghi"; distance:3 within:9);

8.

Based on the Internet (<http://www.ossec.net/?page_id=165>), OSSEC has following features and I quote as follows:

* OSSEC helps customers meet specific compliance requirements such as PCI, HIPAA etc
* OSSEC lets customers implement a comprehensive host based intrusion detection system with fine grained application/server specific policies across multiple platforms such as Linux, Solaris, AIX, HP-UX, BSD, Windows, Mac and Vmware ESX
* OSSEC lets customers configure incidents they want to be alerted on which lets them focus on raising the priority of critical incidents over the regular noise on any system. Integration with smtp, sms and syslog allows customers to be on top of alerts by sending these on to e-mail and handheld devices such as cell phones and pagers. Active response options to block an attack immediately is also available.
* OSSEC will integrate with current investments from customers such as SIM/SEM (Security Incident Management/Security Events Management) products for centralized reporting and correlation of events.
* OSSEC provides a simplified centralized management server to manage policies across multiple operating systems. Additionally, it also lets customers define server specific overrides for finer grained policies.
* OSSEC offers the flexibility of agent based and agentless monitoring of systems and networking components such as routers and firewalls. It lets customers who have restrictions on software being installed on systems (such as FDA approved systems or appliances) meet security and compliance needs.

9.

A. My modification is highlighted in yellow as below, mainly based the random search, the worn in each station will keep a history of its hits, and try not to hit the same target twice in the future. In the way, the worms will be faster since it saving time on avoiding hitting the same targets again and again, in the meanwhile, it is also quieter.

public:

Station(int id = 0):id\_(id), vulnerable\_(false), infected\_(false),

timeInfected\_(0.0), numAttempt\_(0), attemptUpper\_(0),HistoryHit({0})

… …

do

{

x = rand()%MAX\_STATIONS;

y = rand()%MAX\_STATIONS;

toID = x\*MAX\_STATIONS + y;

}while( HistoryHit[toID] == 1 || toID == id\_);

HistoryHit[toID] = 1 ;

… …

private:

int id\_; // id of the station

bool vulnerable\_; // true if station can be infected

bool infected\_; // true if already infected

double timeInfected\_; // time when infected

int numAttempt\_; // number of infection attempts on this station

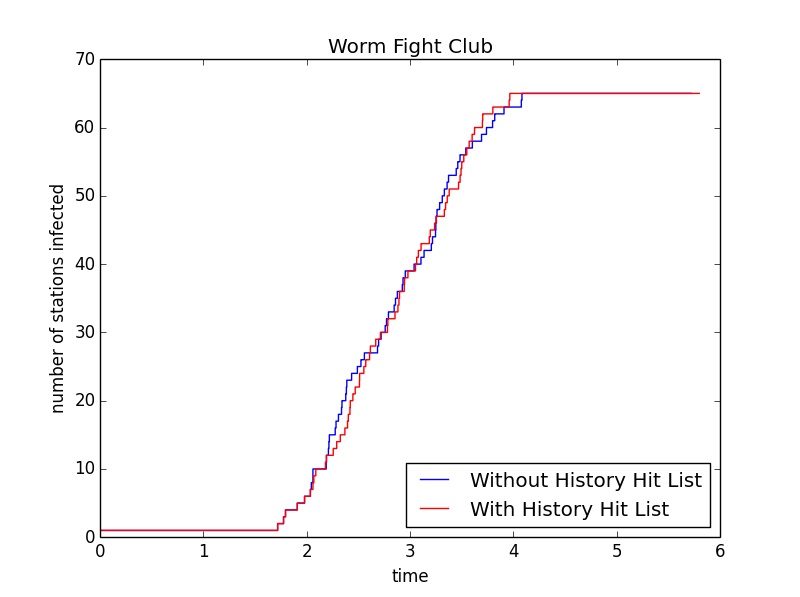
int attemptUpper\_;

int HistoryHit[MAX\_STATIONS\*MAX\_STATIONS];

};

B.

The plot of time to infect all vulnerable stations is as below:



And the average time to infect all stations of the random search is 4.08356 seconds, and its the average number of infection attempts per station is 5.6875; while for my method, the random search with a history hit list, that the average time is 3.96619 seconds, and the average number of infection attempts per station is 4.640625.

Since the network space is not very big, only 256 stations, thus the difference between these two methods is not significant, but still, we can see that worm with a memory of history hit list will save itself some time by avoiding hit the same targets more than once, hence it took less time to infect to the vulnerable machine.

10.

We assume , that the number of secure machines are greater than vulnerable machines

If , then

If , then

If , then

In all, under our assumption, we have

11.

This question has no difference from the last one, assume that we still have

12.

:(