Critical Infrastructure Security:

The Emerging Smart Grid

Final Take-Home Exam

Kirsten Fenske

George Gruber

May 3, 2012

1. Choose one real distributed application program that you are familiar with (or could become familiar with if you don't know any (!!!)). Write the following:

• A paragraph or two giving an overview of the application. Make sure to include which generic architecture (client-server, publish-subscribe, P2P, . . .) you believe best categorizes the application, and why it does so. Be sure also to describe what application-specific activities the various pieces in that architecture (clients, servers, whatever) are doing: what they are requesting, processing, etc.
• A paragraph or two describing whether you believe the following runtime issues are important to users of this application and why you believe it (e.g., what happens in its absence):
  – Low latency network connections
  – High bandwidth available
  – Perfectly "consistent" and "correct" replies/answers/service versus an inconsistent or approximate reply/answer/service

Make sure your answer indicates how you would desire to trade off the above runtime properties, and the "worst" for each it could reasonably tolerate.

The first distributed networking program I can think of is the MMO World of Warcraft. In this game, there is an overall subscriber base of over ten million users. The overall architecture is client-server, but with tens or hundreds of clients involved. In general there are several types of actions which can be taken in this game: communication, movement, single-target abilities, and area of affect abilities. Each of these has requires a different amount of communication between the client(s) and server:

- Communication: There are private and public forms of communication, though the overall method remains the same. When a player communicates, the message is sent to the server and echoed to one of several channels. Some examples are: General chat, Trade chat, Party chat (for groups of players), and private whispers. All members of these channels receive a copy of this message. If the user joins a channel after the message is sent, it isn't received.

- Movement: Each user's location is synced with the server, but the movement is handled on each client's machine. This is evidenced by cases where "lag" occurs. A user can continue to move and this will be reflected locally, but after a certain point (1-2 seconds) when the user has not been able to communicate with the server, the user's screen will freeze at a certain point, and they are still able to input commands (movement included). All of these commands will be tracked locally until client-server communications are

resumed. In any case, the server broadcasts each user's most currently known location to all users who are near them. The distance is defined by the broadcast recipient's location and their "draw distance", which is the radius at which their system will show other players. This viewing player receives information on location and movement direction. The movement direction is for animation purposes, the pinged location is what actually "moves" players on screen.

- Single-target abilities: These can affect the user, another user, or a non-player character (NPC). When an ability affects the player, the "activation" of the ability is communicated to the server, the server checks to see if that ability can be used (sometimes other effects block the use of abilities) an acknowledgement is returned, and the server lists it among active effects on the player. Lists of effects and changes in player health or energy level (the resource system which lets a user use abilities) are handled at the server side, as evidenced by the fact that changes to these do not happen when lag is occurring. Similarly, when an ability affects another user or NPC, checks are performed and the server decides whether the effect should be placed on that user or NPC.

- Area of effect abilities: These use a combination of movement/location data and effect management as with single-target abilities. Area of effect (AOE) abilities basically create a zone in which effects can be applied. The server checks to see if a player or NPC is in that zone. For any entities in the zone, the server checks to see if the effect applies (some abilities affect a limited number of entities in a region), then applies that to their status in the server.

For the three runtime concerns, the most important by far is low latency. Because the ideal concept is a near-realtime execution, the response time between the server and each client must be low. If not, lag occurs and users will be seen running in place, standing still, or skipping between several nearby locations. As long as the latency is under ~.5s, this generally evens out as overall location is controlled by the server. This depends on the activity, though. For solo player-versus-NPC activities, latencies up to 1 second may be acceptable. For player-versus-player activities, this must be as low as possible, closer to 0.2s.

The second most important is consistency and correctness of data. If a location packet is skipped, the difference in location will be small. Similarly, an erroneous location packet will quickly be overridden by a new, accurate location. If an action or status packet is incorrect, it can result in being unable to use an ability briefly, but this will eventually change. This is fairly easily managed as long as statuses are allowed to overwrite each other to correct data.

The least important (in general) is high bandwidth. If there are many users in one location (a major city or trade hub), most systems will have performance issues due to the rendering of so many dynamic characters. This is largely due to individual system performance, though. The players' locations are all brief packets, and the packets denoting their appearance are simply links to files on the client computer. It is very rare to see issues caused by bandwidth alone, as state and location information are fairly simple (location, movement direction, positive effects, negative effects, current health and energy levels) and can be represented in short information packets with no direct graphical data. Generally, performance degrades with connection rates below ~1Mbps, but this can be due to fluctuations in connection quality as much as anything else.

2. Explain in a paragraph or two in your own words the analogy “Middleware is to socket programming what high-level language programming is to assembler programming.”

At its basic level, the socket programming and assembler programming are using the low level, basic services that the hardware and operating system provide. In the case of assembly programming, this means access to the registers and basic functions provided by the hardware. In the case of socket programming, this means access to the basic network mechanisms and services provided by the networking hardware (Ethernet/wireless card). These are used to create lower-level tools, without levels of XYZ-ware between them and the actual system processes. This allows for a higher speed of execution and more direct control over those processes.

Middleware and high-level languages provide a buffer and in a lot of ways hide these low-level occurrences from the user or client. Generally speaking, these can have much more functionality because they are built on the backs of preexisting lower-level tools. However, each will have higher overheads because they have more dependencies and packages that may be required to run them. In the example of distributed or cloud computing, middleware can be used to hide the

fact that different systems are used, providing a framework for other software to build on, a unified front. However it uses the tools developed at the socket level to do so.

In general, if you want something to run very quickly and efficiently without a lot of overhead, you will want to use socket or assembler programming. If you want to design something broader in scope which may *use* these lower-level programs, you will be better suited using middleware and high-level programming languages.

3. Which of the following is false for a NASPInet-like critical infrastructure data delivery service for a power grid?
A. The scale is such that tracking per-flow state at a router (or similar forwarding device) is feasible.
B. The admission control perimeter can be complete.
C. Post-error recovery is sufficient.
D. The predictability of the delivery latency, even in the presence of a small (bounded) number of failures, is very good or better.

4. Which of the following is more burdensome for the application programmer?
A. Remote procedure call
B. Publish-subscribe
C. Remote method invocation
D. Request-reply protocols

5. Which of the following is space-uncoupled:
A. Remote procedure call
B. Publish-subscribe
C. Remote method invocation
D. Request-reply protocols

6. Which is not either a motivating observation or a property provided by P2P systems (i.e., what is false):
A. Compared to exploiting in-network logic, the same level of multicast efficiency and very low delivery latency is achievable (even in the presence of a few faults).
B. CPU/GPU and storage resources are often more abundant “at the edges” of a distributed system.
C. A much larger address space can be handled compared to IPv4 or even IPv6.
D. Using only edge resources means that deployment is much simpler than if in-network logic (e.g., GridStat forwarding engines) were required.

7. Which of the following challenges for the bulk power grid cannot be mitigated with much better wide-area communications (and sensors feeding it):
A. The different physics of some kinds of renewable power (e.g., no reactive power) and how that changes the dynamics of a grid at large.

B. Negligible storage of power in the grid.
C. Large numbers of retiring operators who are leaving with a lot of institutional knowledge—intuitive "seat of the pants" understanding—of how their grid operates in many contingencies or unusual situations (and combinations thereof).
D. Increasing stress on grid due to insufficient new transmission capacity compared to increases in both generation and load.

8. Tripwire is a software tool intended to assure integrity of system files by detecting un-expected modifications (such modifications are often a sign of rootkit activity). One version of Tripwire reads the names of the directories to be protected from a configuration file. For each file in the specified directories, Tripwire computes its hash value and stores it in a database. What property must this hash function have?
A. preimage resistance
B. 2nd-preimage resistance
C. collision resistance
D. all of the above

9. SYN cookies (http://cr.yp.to/syncookies.html) are used to mitigate SYN flooding attacks. Which of the following is not used in computing a SYN cookie?
A. Timestamp
B. TCP flags
C. Maximum segment size
D. Port number

10. Suppose you have an intrusion detection system detecting a computer virus on the network with 90% accuracy. Precisely, the IDS detects a connection transferring a virus as an attack with 90% probability and a benign connection as an attack with 10% probability. When 1% of the connections contain a virus, what is the probability that a connection flagged by the IDS as an attack is actually benign?
A. 0–20%
B. 21–50%
C. 51–90%
D. 91–100%

11. The following statements describe Kerberos authentication. Find an incorrect statement.
A. The KDC is a single point of failure.
B. KDC and the ticket-granting server can run on a single machine.
C. Kerberos uses symmetric encryption.
D. Every message in Kerberos authentication is encrypted.

12. Which statement is false?
A. Stuxnet exploited multiple zero-day vulnerabilities.
B. Stuxnet attacked systems from certain vendors only.
C. A network physically separated from the Internet can be infiltrated by Stuxnet.
D. The author of Stuxnet created two fake certificates to sign drivers.

13. The article “W32.Duqu: The precursor to the next Stuxnet” describes Duqu, a threat similar to Stuxnet. Compare Stuxnet and Duqu from the following aspects.
(a) Initial infection
(b) Propagation
(c) Command and control

Stuxnet and Duqu are two closely related attacks, with two very different goals. Where Stuxnet was a worm that propagated quickly through specific private networks, Duqu was more of a remote access Trojan out to harvest industrial information. It is suspected that these attacks were sent out by the same group, since they share almost identical source code. Both Duqu and Stuxnet masked themselves as legitimate code using a driver file signed by a valid digital certificate. These attacks also avoided detection by debuggers by tricking systems into accessing their components from the memory rather than the hard drive.

In terms of infection, Stuxnet exploited the naiveté of JMicron and Realtek employees by leaving seemingly harmless USB sticks in their parking lots. When the employees plugged these USB sticks into their computers, the Autorun program ran a piece of code at the end of the program, allowing attackers to steal the digital certificates necessary to legitimize their code. If the Autorun was disabled, users were prompted with two ‘open’ options in the command window, which also activated the worm. From here, Stuxnet exploited multiple zero day vulnerabilities as well as peer to peer communication and updates in order to install itself, propagate itself and activate malicious processes in private networks. Once it loaded itself into the computer’s memory, injected itself into trusted processes, tested the network connectivity, it would send basic information about the compromised machine to its command and control servers in Malaysia and Denmark. If the system met a specific set of criteria, the worm would disturb the operation of the SCADA systems.

Duqu used a more traditional mode of infection. By sending emails to specific addresses with an attached MS Word document, the attackers were able to exploit the MS1-073 zero day vulnerability to install the remote access Trojan without garnering suspicion. Once Duqu successfully infected a computer it communicated through HTTP and HTTPS with its command and control server from which it downloaded a keylogger. With this keylogger, data and passwords were collected, and used to move through the network. Data meeting a certain set of specifications are then recorded and shared with the command and control server. To propagate, Duqu would copy itself onto shared folder with the authentication of the infected computer. Once Duqu was on the next computer, it would schedule its own execution and begin the process all over again.

14. Read “21 Steps to Improve Cyber Security of SCADA Networks” (http://www.oe.netl.doe.gov/docs/prepare /21stepsbooklet.pdf) from DoE, and answer the following questions.

(a) If properly implemented, how would each step help preventing Stuxnet attacks?

1. Identify all connections to SCADA networks.

Since Stuxnet propagates from one computer to another within the same private network, knowing all the connections in a wide area network makes the worm easier to track down. Since Stuxnet was targeting SCADA networks, future attacks could be mitigated by knowing what connections the worm was targeting.

2. Disconnect unnecessary connections to the SCADA network.

By limiting the connections to the SCADA network there are fewer pathways in which a worm could propagate. Disconnecting these pathways also serve to limit the amount an attack like Stuxnet could communicate with its command and control server.

3. Evaluate and strengthen the security of any remaining connections to the SCADA network.

Strengthening security in the remaining pathways would help improve response time in the event of an attack, since the system would be monitored in concentrated locations.

4. Harden SCADA networks by removing or disabling unnecessary services.

Since Stuxnet initial infection was the result of an Autorun program on USB sticks, removing/disabling unnecessary services would limit what programs could automatically load without the user’s permission.

5. Do not rely on proprietary protocols to protect your system.

Proprietary security is good to have, but it should never be the primary security. Proprietary security is designed to be proficient in many applications, and for a more dependably secure system, it is best to expand to more application-specific security. If proprietary protections were the one means of security at the time, Stuxnet could have simply exploited a single zero day vulnerability in the system to gain access to the entire SCADA network.

6. Implement the security features provided by device and system vendors.

Since a worm like Stuxnet exploits zero day vulnerabilities, implementing security features such as product patches and upgrades will either work to limit the spread of an attack or put a stop to it entirely.

7. Establish strong controls over any medium that is used as a backdoor into the SCADA network.

By limiting access to back doors and vendor connections, potential infection can be limited. Stuxnet took advantage of a back door hard coded password server, which allowed it to use the credentials of other users to propagate. By strengthening authentication requirements for back door access, Stuxnet would have a hard time propagating throughout private networks.

8. Implement internal and external intrusion detection systems and establish 24-hour-a-day incident monitoring.

Attacks like Stuxnet rely on keeping personnel in the dark about their presence until it's too late to stop it. By implementing internal and external intrusion detection systems, better visibility in the system could be achieved, resulting in a faster reaction time.

9. Perform technical audits of SCADA devices and networks, and any other connected networks, to identify security concerns.

Knowing where the SCADA devices and networks are weakest or compromised allows for the paths of lease resistance to be attended to, before an attacker exploits them. Since a worm like Stuxnet preys on vulnerabilities, reducing them will thereby reduce the probability of infection.

10. Conduct physical security surveys and assess all remote sites connected to the SCADA network to evaluate their security.

Since any connection could be targeted by an attacker, it is important to check regularly for any security compromises. If connections like unknown USB sticks were stopped before they began, the spread of Stuxnet would have been profoundly delayed.

11. Establish SCADA "Red Teams" to identify and evaluate possible attack scenarios.

In the end, nobody knows more about the potential vulnerabilities in a system than the people who work on it. By identifying potential attack scenarios and evaluating their impact, protection strategies can be constantly updated and improved. If potential zero day vulnerabilities were identified in advance, Stuxnet would have had a harder time with its initial infection.

12. Clearly define cyber security roles, responsibilities, and authorities for managers, system administrators, and users.

Clear definitions remove ambiguity. Removing ambiguity of responsibility leaves personnel with only sufficient authority to do their job. By limiting one person's roles and responsibilities, you limit how much impact they have on the network. This way if Stuxnet were to steal one's

person's credentials, they would only get the worm so far in the network, rather than complete access.

13. Document network architecture and identify systems that serve critical functions or contain sensitive information that require additional levels of protection.

By documenting the security architecture and its components, the protection strategy can be better understood to identify a single point of failure. The sooner a point of failure can be identified, the sooner it can be patched to limit exploitation. Limiting the zero day vulnerabilities in the Microsoft machines would have reduced Stuxnet's impact on the network.

14. Establish a rigorous, ongoing risk management process.

Establishing a strong risk management process would enable personnel to make more informed decisions that may affect sensitive information. In the case of Stuxnet, if the employees of Jmicron or Realtek had known the risk of plugging in those abandoned USB sticks, they probably would have abstained from doing so.

15. Establish a network protection strategy based on the principle of defense-in-depth.

A good strategy allows for a more thorough security. By limiting the access of individuals to any specific resources, you limit failures to smaller sections rather than the entire network. These kinds of restrictions would have limited Stuxnet's propagation, since the credentials it stole within the network would only grant it limited access.

16. Clearly identify cyber security.

By eliminating the sole dependence of cyber security on individual initiative, security programs can be more structured and reliable. When people know that they will be held accountable for breaches in requirements, those people take those requirements more seriously. These requirements could have discouraged personnel from plugging in the abandoned USB sticks that contained Stuxnet.

17. Establish effective configuration management processes.

Configuration management ensures that any upgrade or new addition to the system won't introduce new vulnerabilities to the network. Since an attack like Stuxnet preys on these vulnerabilities, reducing their presence in the system would make initial infection difficult.

18. Conduct routine self-assessments.

Self assessment processes allows for the networks to systematically identify, analyze and correct errors that many occur. By performing these assessments on a regular basis, weaknesses and errors can be discovered by the system before a potential attacker discovers them. Since Stuxnet

infects through unknown vulnerabilities, having these vulnerabilities regularly tested and analyzed would limit Stuxnet's infection.

19. Establish system backups and disaster recovery plans.

In the event of an emergency, recovery of normal operation is of the utmost importance. Therefore, by exercising recovery plans to ensure their effectiveness, the systems reliability can be assured. When Stuxnet tampered with system frequencies, it disturbed the normal operation of motors connected to the network and created a state of emergency. With a more robust recovery plan the system could have been recovered, rather than shut down.

20. Senior organizational leadership should establish expectations for cyber security performance and hold individuals accountable for their performance.

Communicating expectations is key to implementing them towards cyber security. If these expectations are communicated and enforced at every level of personnel, standards will be better maintained and un-abiding personnel held accountable. Since an attack like Stuxnet relies on the naiveté of personnel, knowing the potential consequences would have limited Stuxnet's infection.

21. Establish policies and conduct training to minimize the likelihood that organizational personnel will inadvertently disclose sensitive information regarding SCADA system design, operations, or security controls.

By establishing strict policies about the release of potentially sensitive information, any system breaches can only apply to a small area of the network. When resources are shared based on a need to know basis and confirmed identity, this makes an attack harder to implement. Without user credentials that could give it access to all areas of the network, the Stuxnet attack would have had to have a different mode of propagation.

(a) Are those steps sufficient in preventing future attacks similar to Stuxnet or Duqu? If yes, justify it. If no, extend those steps to prevent such attacks.

These steps are sufficient in preventing future attacks like to Stuxnet or Duqu. Both of these attacks require someone from within the network to activate the initial infection. But if rules 12, 13, 16, 20, 21 were abided by, personnel would be more aware of the risks and potential consequences of plugging in anonymous USB sticks or opening attached MS Word documents from suspicious emails. Rules 1, 2 and 3 would ensure that all connections to the SCADA network were identified, trimmed to only the bare minimum and strongly secured. Rule 4 would prevent any unnecessary service from running without the user's permission, which would limit

the initial infection of both Stuxnet and Duqu if it were implemented properly. Rules such as 3, 5, 6, 7, 8, 9, 10, and 11 would ensure constant effort was being put into determining any weakness before an attacker exploited it. If the network is constantly being tested, updated, patched and tested some more, there are very few consistencies for an attacker to depend on.

The inherent problem in cyber security is that one can never know what an attacker might be thinking. Personnel can only improve the network based on past experience, while an attacker designs his attacks based on things the network overlooked. Although there is no sure way to prevent a Stuxnet or Duqu style attack, with awareness, accountability, enforcement, assessment and innovation the network can be well armed.