Supporting a Laptop Environment

and Thwarting the Wireless Anarchists

Presentation by
Erick Engelke
Engineering Computing
University of Waterloo
erick@uwaterloo.ca
CanHEIT Conference
June 2006

First Steps
Like many other universities, the University of Waterloo decided to deploy wireless for faculty, staff and student access a few years ago.

Our initial design requirements were relatively simple; we knew we wanted to check the identities of the users. We would test their userids and passwords against either of our two primary campus Active Directory LDAP stores, we would have a few exceptions allowing special access or denials, and then log usage.

These requirements are typically known as the "three A's": authentication, access control and auditing. There are now many solutions available to do such identity management for wireless environments.

Our particular system was written by Bruce Campbell, now Manager of Science Computing at the University of Waterloo. He created a captive portal using a FreeBSD based router. It is called NAA, or Network Authentication Appliance.

Beyond those basic capabilities, we had little idea of what else would be required.

Relatively quickly, the trends became obvious.

- Laptops now outsell desktops, and there is every reason to expect this trend will continue.
- Laptop growth presents incredible opportunities for teaching, learning and research, but also increased effort by IT support staff.
- Time spent chasing high bandwidth users and virus infected laptops is inefficient and not scalable. To plan for the future, we would need to improve our handling of these two problems.

Bandwidth Management
Shortly after we enabled laptop access, wireless clients started appearing in our high-bandwidth lists. There were several causes, including music and video file sharing, denial of service (DoS) attacks due to viruses and other non-academic applications.

Chasing wireless users about bandwidth usage is time consuming and frustrating. This raised the question: is it possible classify network traffic as good (academic) versus bad (recreational or the results of system compromise)? This is not to say that we want to eliminate recreational use of our Internet access, we just did not want it to choke our network pipes to the detriment of academic network traffic.

At first glance, it would appear that classification of traffic as good or bad should be possible. Kazaa (file sharing) and skype (free internet phone service) are recreational and often high bandwidth uses. So they would be considered bad. But secure shell (ssh) is used for terminal access to unix computers, so it would be good.

Unfortunately, life is never that simple. Skype is an excellent tool for collaboration with academics at other sites – which is good. And ssh can be used to tunnel other protocols which could include bad activities.

Our problem was that we were looking for the wrong type of classification. Protocols themselves are not good or bad. What we really wanted was to reduce or eliminate unnecessary bandwidth usage.

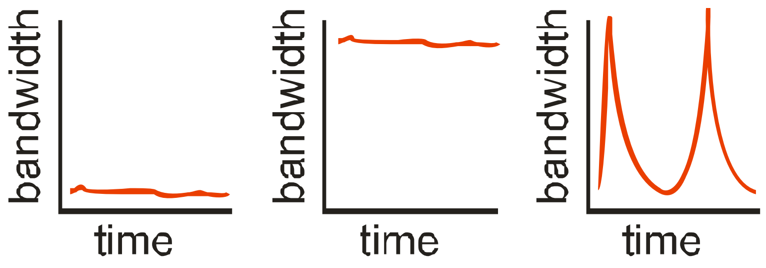
The obvious solution would be to consider traffic shaping, the concept of limiting the amount of bandwidth available for individual users so that they will not exceed some predefined limit.

In our case, we noticed that most high bandwidth users were often downloading multiples of 4GB per day. This number approximately corresponds to the data capacity of DVDs.

The most common method of traffic shaping is to use flat rates. If we were to limit users to a generous 2GB of traffic per day, approximately one half of a DVD, that would mean we would need a flat rate of 20kB/s. But that would be totally unacceptable, web pages would be slow to download, and a hypothetical 30MB PDF download, quite common for academics, would require twenty-five minutes.

Increasing the flat rate to 100kB/s is not a great improvement. Web access would still feel slow, the sample PDF would require a sluggish 30 seconds, and bittorrent could then download two DVDs per day.

Flat rate shaping is ineffective because it looks at the wrong problem. Consider the traffic patters of users.



There are really three patterns that we observe: relatively low bandwidth, consistently high bandwidth, and intermittent high bandwidth.

We need not worry about the low bandwidth clients, they are not causing us problems.

The sustained high bandwidth users are consuming a lot of network resources, and they typically represent activities like file sharing, DoS, spam, etc.

Finally, the intermittent high bandwidth uses are typical of web browsing, interactive graphical terminal sessions, etc. When you browse the web or work remotely, you typically get a burst of information and then you take time to read, think about or edit it. Bursty traffic usually represents good uses, but it is also the type of traffic which is most penalized by flat rate shaping.

Fancy vendor solutions like RED and WFQ are all very coarse tools for bandwidth management because they have the same problem as flat rate shaping: they only measure what *is* going through the network pipe, not what *has* gone through the pipe. A system to detect bursty traffic has to have a memory of what has happened before. We need some sort of feedback loop.

Bruce Campbell developed this insight and introduced a new feature to our wireless controller. He called it the Toilet Tank Traffic Shaper (TTTS).

The TTTS emulates a toilet. The new wireless user is presented a reservoir of unused bandwidth credits. The system has a high output flow and a low input flow.

Users can enjoy a burst of high bandwidth, but it slows to a trickle if the load is sustained – just like the toilet when you continuously hold down the lever. When the lever is released, the reservoir refills and is available for the next big download.

In his implementation, Bruce created parameters to control the mechanism. Those parameters and our current values are:

- tank size: 200 MB
- max bandwidth: unlimited
- min bandwidth: 40 kB/s
- min empty time: 5 minutes
- full percent: 80%
- separate upload/download queues

Internally, our system uses FreeBSD's flat rate traffic shaping on a per-client basis. A cron job looks at previous bandwidth credits and subtracts the bandwidth used in the last short period, then applies the other parameters to set the new pipe limits.

High volume traffic users see a gradual slowing of bandwidth. But when they stop that high bandwidth activity, the available bandwidth is slowly restored. Bruce uses the analogy, "Doctor, it hurts when I do this!" and the reply "Well, don't do that!" We frustrate those until they learn to not consume large amounts of sustained bandwidth.

In contrast, users who just need low bandwidth or whose bursts are smaller or less frequent never run out of credits and are never rate shaped. In fact, 95% of users typically see no rate shaping; it's as if they were unrestricted.

This system eliminates the need to chase or scold users, or even pass judgment on their activities. They simply cannot get in trouble for bandwidth issues.

Client Admission Control

Every year we see an increase in the number of security patches for operating systems and a growth in viruses. Laptops are a particular problem because they are usually client-managed (or perhaps not managed) and we are powerless to fix them.

Reg Quinton of our central IT department (IST) linked our NAA wireless authentication system with snort and other techniques to detect evidence of security threats, and to scan incoming clients for known compromises. Users of compromised or suspicious traffic are automatically Emailed warnings.

Windows clients seem to be the most targeted and the most susceptible to viruses, spyware and other malware. This statement is based on the overwhelming number of Email warnings sent regarding windows related compromises.

Much like our previous network bandwidth issue, the problem of tracking repeat warnings regarding compromised systems was becoming a staff resource issue and would not scale with increasing laptop usage. Something needed to be done.

Ideally, we wanted a strategy to encourage responsible laptop management. A bare minimum requirement for Windows system should be having an antivirus product and configuring the computer to receive security updates from the vendor. What was less clear was how we could convince users to act on this advice.

There are a number of solutions one could employ to spread the word. Education and rewards are the positive solutions. Negative solutions include nagging, embarrassing and punishing.

Education and nagging had been tried and failed miserably. Our next solution was to employ rewards and punishment, and do so in an automated way so that staff effort would be minimized.

It would seem ideal to identify only problem operating systems (currently just Windows) and subject those systems to a security scan before allowing them to use our network. A key goal was that all other operating systems would be totally immune to the limits placed on the problem operating systems.

This idea is not totally original. There are several client validation technologies available or becoming available.

- Cisco has Network Admission Control which does similar things, but it would require a significant capital investment to replace existing incompatible network hardware.
- Microsoft has Network Access Protection, a technology dependant upon Vista server, a future operating system

Neither option was appealing, so I created a new Windows program called MinUWet which validates Windows clients meeting our minimum campus standards for laptop management. Those clients who pass are granted premium network access to almost every type of protocol – the reward. Those who fail MinUWet's test, or choose not to run it, are relegated to just HTTP access. HTTP is sufficient technology to download Windows patches and to update antivirus definitions – which means it is sufficient access to bring a failing machine up to campus standards without any assistance.

The Cisco and Microsoft strategies are more advanced than MinUWet, but it has the advantages that it is available today on our existing hardware (any Ethernet or WiFi infrastructure) and it is compatible with every device our clients have brought to our network.

The MinUWet solution runs partly on our NAA authentication devices and partly on the client's computer. Technically the client portion is in the class of programs called agents, because it is the data collection part of a bigger program running on the NAA.

As the user connects to our wireless system and passes the authentication stage, the NAA uses various means to determine if the client's computer is running Windows. Those computers are flagged and assigned only HTTP access, and the user is prompted to run the MinUWet agent. All other types of computers are immediately assigned almost full network rights, only the Windows computers are restricted.

If the user chooses MinUWet, the agent downloads to the user's computer temporarily. It is a cryptographically signed application, so the user is assured by the browser that the program is officially sanctioned by the university and not a virus.

Then MinUWet checks and reports just two things: is the computer running an updated and enabled antivirus program, and is it receiving Windows updates? The results are automatically submitted back to the NAA, and it decides whether to grant premium network access, and it returns an HTML message to the user identifying its results and, if the client failed, advice on how to fix the system so it will pass the test in the future. In many ways it resembles a networked version of the Windows Security Center application.

Internally, MinUWet relies on Windows Management Instrumentation, the same technology Windows Security Center uses to detect third-party antivirus products. This allows MinUWet to be compatible with all antivirus products. It also uses cryptographic signing when conversing with the NAA to ensures users cannot easily create a MinUWet-compatible program to avoid compliance with our minimum standards.

A two week trial was conducted in the Faculty of Engineering. Over six thousand user sessions were monitored. About one quarter of them initially failed, but half of those failures were fixed by the users or the help desk and subsequently passed.

There were no observed snort security threats, a marked improvement over our previous multiple warnings per day.

Given the success, we decided to take the system campus-wide. Two weeks prior to deployment, we informed IT support staff, posted advance warning in the campus-wide daily newsletter and added a notification to the wireless login screen. This media blitz meant many users tried the system (with no impact on their access) in advance of it going into enforce mode.

Help desk staff reported an increase in the number of requests for laptop assistance to bring them up to compliance, but it was not overwhelming. In fact, the IT staff was very happy with the results, particularly since users didn't have to be nagged anymore.

Most users actually fixed the systems themselves. And in a large number of cases the only problem was that the antivirus definitions were not downloading automatically.

The test added about five or ten seconds delay to the login process. While this seemed like an acceptable delay during the design phase, it was a common source of complaints.

Our solution was to add "memory" at the NAA, so that users would only be prompted to run MinUWet about once per week. Now approximately two thirds of all wireless sessions are pre-approved and do not need to download or run the agent.

Running the agent once per week has been an acceptable solution for users while being frequent enough to catch laptops falling out-of-scope of system patches or antivirus definitions before they become a problem.

We concluded that client validation works very well and someday every school will use it. It makes client support more scalable, and it diagnoses security problems before they result in data loss or suspension of user access.

Some users know they will fail MinUWet, so they do not bother running it. They must live with inferior network access, but that also means they are less likely to be able to spread problems. There are tweaks we may consider to improve this situation. For example, we may require MinUWet before even HTTP access if the user has already received a security-related Email warning.

Conclusions

Effective traffic shaping and client validation are two very effective tools. They improve network and client system management in a scalable manner.

We encourage others to incorporate these ideas or even our code in their own deployments.

The problems associated with wireless mostly disappear as users are lead to appropriate use through the careful use of rewards and repercussions of their activities.

These two solutions are already in use for some wired ports. They would also be good candidates for our residences and some graduate student offices where our current system of nagging users has not been effective.

With our two most pressing wireless problems solved, we see the next challenges to be simplifying client access to resources, wireless data encryption and dynamic roaming. There are several possible solutions we can consider, but further thought will be necessary to balance client ease with the required functionality.