Slide #1:

Hi, my name is Cliff Zou. I will talk about our work on "monitoring and early warning for Internet worms".

Slide #2:

This paper tries to answer this question: how to detect an unknown worm at the early stage of its propagation? If we can early detect a worm, we may have time to set up efficient counteractions before it is too late. First, we have to set up a monitoring system, which will monitor and collect worm scan traffic, such as connections to nonexistent IP addresses. However, monitored traffic is very noisy: some old worms can probe the same port; some hackers can use port-scanning toolkits to scan our monitors, or monitored traffic can be caused by misconfigured routers or computers on the Internet.

For the detection part, for unknown worms, we have to rely on anomaly detection. Currently, most anomaly detection techniques are threshold-based. That is to say, they check monitored traffic burst, either short-term burst or long-term burst. If the burst is over their threshold, they will raise an alarm. However, threshold-based anomaly detection systems usually have high false alarm rate and their threshold is very hard to adjust.

Slide #3:

Therefore, in this paper, we propose a very different approach, which is a non-threshold-based worm detection method. We call it "trend detection", the detection principle is: detect the traffic trend, not burst.

We believe worm exponentially propagates at the beginning. So the trend means the exponential growth trend of a worm. For detection, we use on-line recursive estimation algorithm to estimate the exponential rate \alpha of this trend. If the estimated exponential rate is a positive, constant value, we believe we have detected a worm; otherwise, the monitored traffic is just some noise burst.

These two figures show the monitored illegitimate traffic in two situations. They can be the number of packets, or number of connections we observe at each unit time. They will cause threshold-based detection system to give alarms if the threshold is below this in this figure and below this value in this figure. However, we think that they are just noise, not caused by a worm, because they do not have the exponentially increasing trend. From our estimation point of view, the estimated value is either value 0, or is oscillating around 0.

(click) On the other hand, this figure shows the monitored traffic in another case. It has exponential growth trend; the estimated value is a positive constant value. So we believe this incident is caused by a worm.

You can think that the "trend detection" is a transformation that is similar to Fourier Transform. The trend detection transforms the original problem in this domain (the three figures above) to the "trend domain" (the three estimation figures in the bottom). In this trend domain, the worm detection will become much easier.

In these three scenarios, these first two will cause trouble or false alarms to traditional threshold-based detection systems, but they will not cause any trouble to our trend detection system.

Slide #4:

Now, many of you may think that: why worms will propagate exponentially at the beginning? Can't the author of a worm change the worm code to make it not propagate exponentially?

Here I want to say that exponential growth is the law of natural growth of any reproduction system. In fact, a reproduction system will have exponential growth when it satisfies these three conditions: (1). The interference between objects are negligible, this is suitable for the beginning phase of the system. (2). All objects have similar reproductive ability, not required to be same. (3). The system is a large-scale system, and then we can use law of large number and do not consider some random effects.

These conditions are very general. Thus many reproductive systems satisfy them and follow exponential growth at the beginning. For example, the rabbit population growth when we bring several rabbits into new grassland; the sales of a good product in a new market place.

Based on this natural phenomenon, we believe that fast spreading worms have to follow exponential growth at the beginning. The main reason is because of the attacker's incentive: attacker wants a worm to infect as many vulnerable hosts as possible before people can take counteractions. If an attacker wants to mess up our early detection system, then his worm will not propagate with its speed limit, will propagate slower. For slow spreading worm, we can rely on other ways to detect the worm, such as security staff's manual check. And slow spreading worm will be easier to control.

Slide #5:

Our trend detection relies on worm propagation model. We will first use simple epidemic model for it. The model assumes that any host is either infectious or susceptible. It also assumes that the contact between infectious group and susceptible group are proportional to the product of their sizes. Denote I(t) the number of infectious hosts, S(t) the number of susceptible hosts, N is the total population size. Then the simple epidemic is this. In this paper, we use discrete-time model, which is this one. \alpha is called infection rate. This figure shows the number of infected hosts as a function of time t. This curve is also called logistic curve, or simple "S" curve.

Our objective is to detect the presence of the worm as early as possible in this area. At early stage, a worm almost exponentially increases. If we only check this beginning part and show it with log-format on Y-axis, it's almost a straight line. These two lines show the time when the worm infects 1% and 2% of the overall population. Thus at beginning, the worm can be accurately modeled by this exponential equation.

Slide #6:

We use simple epidemic model for worm modeling because it can model most scan-based worms, such as Code Red, Slammer, even recent Blaster worm that did not uniformly scan the Internet. It can at least model the beginning part of a worm's propagation accurately, because at the beginning there is not much human intervention or network congestion. These two graphs are taken from a paper that analyzes Slammer and Code Red. They show the observed Code Red and Slammer propagation compared with the simple epidemic model. Code Red propagation matches well with the model. Because of congestion, Slammer quickly saturated network and slowed down its spreading speed, but at the beginning it still followed the model.

In addition, our trend detection method is not just useful for simple epidemic model. With minor modification, we can use our trend detection method on any worm models as well.

Slide #7:

In our trend detection, we use Kalman filter to estimate the infection rate \alpha at each time step. In the estimation case, Kalman filter is equivalent to recursive least square estimation. We choose Kalman filter because it can give estimated value at each discrete time, and it is robust to noise.

The system is modeled by the discrete-time simple epidemic model. From monitor, we can have two types of observation data: C_i is the cumulative number of infected hosts we have observed until the discrete time i; Z_i is the number of worm scans we observe during the discrete time interval i. They all have this relationship with I_t by replacing

y_t to C_i or Z_i. The estimation parameters are \alpha, the infection rate and \beta, the parameter in epidemic model.

Slide #8:

The system is described by this. X_t is the system state we want to estimate at each time step. The discrete epidemic model is presented in this y_t equation.

Slide #9:

We run Code Red simulation to check our trend detection system. Note that we use a normal distribution to give different infected hosts with different scan rates. We assume that we can collect worm scans to 2^{20} IP address space. It is about 16 Class B network space, which is not hard to achieve. We also consider background noise in the simulations.

The blue line shows the number of infected hosts on the Internet, the red curve shows what we observed. Based on the observation data, this figure shows the estimated infection rate \alpha as time goes on. In this simulation, the worm infects 2% of population at time 223 minutes. From this figure, we can see that at that time, the estimated value is already stabilized and oscillating a little bit around a positive constant value. The black dash line is the real value of \alpha. Thus we can see that we can detect the worm before 2% of vulnerable hosts are infected. We have run the simulation for many times; all of them have the similar results.

Slide #10:

We also run Slammer simulation. These are simulation parameters. This figure shows the estimated result. In the simulation run shown here, the worm infects 1% population at time 45 seconds. The estimation shows that at this time, the estimated value is already stabilized and oscillating a little bit around a positive constant value, although at this time the estimated value is higher than the real value. Therefore, for this fast spreading worm, we still can detect this worm when it infects only 1% of vulnerable hosts.

Slide #11:

We also studied early detection for Blaster worm. Blaster sequentially scans from a starting point in IP space. The starting IP address has 40% chance to be a local address, 60% chance to be a randomly picked address. Our simulations show that Blaster still follows simple epidemic model.

We use the same simulation parameters of Code Red for Blaster. We simulate Code Red and Blaster for 100 runs, respectively. This figure shows 95 percentile and 5 percentile of these two worms. The 95 percentile and 5 percentile means that in those 100 simulation runs, 90 runs of Blaster are between these two red curves. Thus they exhibit how the worm varies in different simulation runs.

Blaster will cause some trouble to our monitoring system. If we still monitor 2^{20} IP space and use 16 distributed Class B for monitoring, then the observation data is the red line. If we distribute the monitoring space into 1024 blocks of space, the observation data is the blue curve. You can see that the observation data is very noisy. How to deal with it? If you check this blue curve, you can still see the exponential trend at the beginning. From frequency domain, the exponential trend is low frequency information embedded in the data. On the other hand, these noises are high frequency information in the data. Thus it is suitable to use a simple low-pass filter to filter out those noises. (click) This figure shows the observation data after the low-pass filter. You can see the 1024-block monitoring can have much better data. By using this data, we can detect the worm when it infects about 2% population. But the 16-block monitoring is not good. It means that in order to early detect Blaster, our monitoring system must be well distributed, not just a bit chunk of IP space.

Slide #12:

Our monitor can only cover a very small percentage of IP address space. If a worm uniformly scans the Internet, it takes some time to send a scan into our monitoring space and be observed by us. Thus the observation data, the cumulative number of observed infected hosts, is biased and smaller than the real value of infected hosts at any time. For example, if we monitor only two Class B network space, the number of Code Red infected hosts is the blue curve in here and this black curve is the number of infected hosts that we observed.

Based on the uniformly scanning property of a worm, we derive this bias-correction formula. From it we can derive unbiased estimation of how many hosts are really infected based on monitored data. The red curve here is the estimated result. However, the bias correction will amplify the noise in observation data. We have analyzed this phenomenon in the paper. This figure shows that if we only monitor 2^{14} IP space, this is the observation data and the red curve is the estimated I_t after bias correction, just a little bit noisy. We have proved in the paper that bias correction can give us unbiased estimate of I_t.

Slide #13:

Based on observation data, we can also predict the number of overall vulnerable population on the Internet at the beginning of a worm's propagation. It's like a weather

forecast and it can give us some guideline on what we should do in face of a worm's attack.

One simple way to estimate the population N is to use this relationship. We can get the estimated value \alpha and \beta from our Kalman filter, thus N is equal to \alpha over \beta. However, N is a very big number, \beta is a very small value that cannot be estimated accurately by the Kalman filter. This approach will give very noisy estimation.

To overcome this, we find an alternative way to estimate the population size. Suppose we have egress monitors and we can observe most traffic sent out by individual infected hosts. Then we can observe and estimate the worm scan rate \eta, which is the number of scans sent out by an infected host in a unit time. Then we can use this equation to estimate N. At each time step, we estimate \alpha from previous Kalman filter, and then use it to estimate the population N. The red curve shows the estimated population from this equation; the blue curve is the estimated result directly from Kalman filter. We can see that the alternative way is much better. The worm infects 5% population after time 250. We can see that at this time, the error of estimation is less than 10%.

Slide #14:

This figure shows typical worm propagation. If we want to detect the worm during this beginning part, we can use the exponential model directly. If we only show the beginning part and use log on Y-axis, (click) this shows that the exponential model is quite accurate for the beginning part.

On average, the observation data y_i is proportional to the number of infected hosts. Y_I can either be the number of scans in each unit time, or the observed number of infected hosts after bias correction. In previous Kalman filter, we use ARMA model, so a simple linear equation for exponential model is this Autoregressive (AR) model. We can use this AR model to estimate \alpha.

After taking log, the exponential model naturally becomes a linear model of \alpha, thus we can use this third model, we call it transformed linear model, to estimate \alpha.

Slide #15:

For the same worm propagation, we use those three models and get these estimation results of worm infection rate \alpha. We have shown this figure in previous slides. It is based on epidemic model. This is based on AR model and this is based on transformed linear model. The transformed linear model is much better than the other two. If we use

this worm model, we can detect a worm when the worm infects less than 0.5% of vulnerable population in the Internet.

Slide #16:

Here we show some intuitive explanations of why these three models are so different. The major reason is the error introduced by these models. In epidemic model, there will be three errors introduced, here, here and here, while in AR exponential model, only two errors are introduced.

The error introduced in AR model has these two items. On the other hand, in the transformed linear model, there is only one error \nu_i and it is smaller than w_i.

Slide #17:

Summary. In this paper, we present a new non-threshold-based worm detection methodology. We call it trend detection. The basic principle is: detect traffic trend, not burst. We hope that it can give researchers a different view besides traditional threshold-based methods. Trend detection has the advantage that it is robust to background noise and will have small false alarm rate. Trend detection relies on the accuracy of the dynamic model, and how good the observation data can represent the real worm propagation.

In addition, we have derived these two formulas for uniformly scanning worms. One is bias correction, it can correct the bias in observed number of infected hosts to estimate the real infected population; another is the forecasting of population N at the early stage of worm propagation. Note that the 2^{32} here means that a worm scans the whole IPv4 space. When the worm scans different size of IP space, such as \Omega IP space, the equation should be this (blue equation). This equation is a very basic equation in worm analysis. Actually, from this equation, we have found a theoretical worm called routing worm. The routing worm can use BGP routing prefixes and can reduce the scanning space to be only 30% of Ipv4 space. In this way, this equation shows that the worm could have more than three times faster spreading speed. Why 30%? Because currently, from BGP routing table we know that only less than 30% of IP space is routable. For a worm, why bother to scan those 70% IP space that is empty and non-routable?