BGP Security via Enhancements of Existing Practices

Xiaoliang Zhao, David T. Kao *

Abstract— **Border Gateway Protocol (BGP) is the de-facto inter-domain routing protocol which logically connects different computer networks into the Internet. BGP is one of the critical Internet infrastructures, but sufficient security protections to BGP are lacking. In the past, false routing information have caused network problems such as prefix hijacking, BGP updates churns, even network melting down. Today BGP security still remains a great challenge which is exemplified by recent Youtube prefix hijacking incident. In this paper, we approached the problem from a different perspective by examining and enhancing the current operational practices in Telecom industry.**

Index Terms—**BGP, routing, security, self-learning, adaptive**

I. Introduction

THE internet consists of tens of thousands Autonomous Systems (AS). Each AS administrates its own networks autonomously. Border Gateway Protocol (BGP) is the de-facto inter-domain routing protocol used to exchange network reachability information between ASes. In the past, due to misconfiguration or software faults, there have been many cases where an AS falsely announced networks it didn't own, which in turn black-holed data traffic. For example, in Feb. 2008, Pakistan Telecom falsely announced a network owned by Youtube.com. Consequently, Youtube.com experienced hours of service disruption which affected millions of users around the world. This is just one example showing BGP is vulnerable and such vulnerability still exists today.

Many research proposals have been proposed to enhance BGP security. One way to protect BGP is to use cryptography algorithms for authority and authentication check [1]-[3]. Secure BGP (S-BGP) [1] is one example, which provides the most comprehensive security protection to routing information. However, it not only requires heavy modification to BGP protocol and implementation but also requires another global infrastructure, Public Key Infrastructure (PKI), to support it. Given its high implementation and deployment cost, for more than ten years, S-BGP has not achieved any significant deployment. The similar limitation is found in other cryptography-based proposals as well. The second approach is using a centralized database to manage the authority of routing information [4], [5]. Routing Assets Database (RADb) [4] is the most popular one. However, since it is a voluntary choice to use RADB, and given the constant change of the Internet, some organizations became reluctant to update the data. Along the time, the database contains more and more outdated and erroneous data. Consequently, fewer network operators are comfortable to use the database. Some research proposals suggested use public routing data to detect false routing information based on certain heuristic rules [6], [9]. The major problem for this kind of approach is lack of real-time protection. Some other work [10]-[14] exercised various idea such as exploiting visualization technique, or integrating data plane and control plane information, or many others. However, most of proposals are facing more or less similar deployment issue. As of today, it remains a remote opportunity to deploy them in a real production network.

As network practitioners, we approached the problem from a different angle. We re-examined the existing BGP security practices and their associated operational costs. The rational is that the existing practices are readily available but the associated operational cost may be too high to use for certain cases. If we can further lower the operational cost and prompt their use, we can improve BGP security to another level. In this paper, we proposed two algorithms based on the self-learning and adaptive concept to reduce BGP operation cost. The algorithms are verified with both public and internal BGP data. Some implementation and deployment considerations are discussed.

II. Current BGP Security Practices

For BGP security purpose, the most popular tools used today are prefix limit and route filter[1]. A prefix limit is a threshold set by operators to limit how many unique prefixes a BGP neighbor is allowed to advertise. Once the limit is exceeded, depending on the configuration, the overflowed prefixes may be discarded or the BGP session may be terminated. A route filter is a list of prefixes which a BGP neighbor is allowed to advertise. Anything not on the list will be discarded. Both prefix limit and route filter are simple techniques but quite effective to reduce the false routing information. However, manually maintaining

* Xiaoliang Zhao and David T. Kao are with Verizon Business Inc., Ashburn, VA 20147 USA (Email: xzhao@ieee.org, david.kao@verizonbusiness.com).

Disclaimer: the opinions expressed in this paper are authors' own personal opinions and do not represent their employer's view in any way.

[1] BGP MD5 is also a security mechanism widely used today but it provides security to BGP session, not the routing information.

such a limit or a list introduces extra cost to ISPs as well as their customers. For example, when a customer needs to advertise a new prefix, it is normally required to register the new prefix with its provider first. The whole registration process may take from hours to several days, which is not very efficient and sometimes causes customer dissatisfactions.

For those large ISPs with many BGP customers, tuning route filter and prefix limit is a non-trivial work. With the emergence of IPv6 deployment in the near future, the demand of changing the existing route filter and/or prefix limit will be even higher, so does the cost. Due to the high operational cost, some ISPs chose not to use these tools. In the Youtube case, the upstream provider of Pakistan Telecom apparently had no route filter in place, which "helped" the propagation of the hijacking routes. As a counterexample, during another less known network event [17], a service provider seemed having filter in place which did prevent false routing information leaking to the whole Internet. By reducing the operational cost, we hope to prompt the further adoption and deployment of proper tools, hence improving the overall BGP security. Our main approach to reduce the cost is to automate the process by using adaptive and self-learning algorithms.

III. AN ADAPTIVE PREFIX LIMIT

The existing prefix limit tool needs network operators manually set the limit for each customer. To set it properly, one largely depends on his/her empirical experiences. Sometimes the limit is set to an unrealistically high value to avoid late adjustments, which largely reduces the protection power of the prefix limit. An adaptive prefix-limit algorithm is proposed in Table I which will automatically change the limit based on the historical data. In addition, a high watermark is in place as the last defense line to prevent damage from the worst case. The algorithm works as the following.

For the first W days after a new customer's BGP session comes up, because there is no enough data yet, the prefix limit is set to a pre-defined value such as the one defined in existing configuration guidelines. At this stage, the algorithm behaves the same way as current practice. In the meantime, the daily count of unique prefixes advertised by the customer is recorded[2]. Then an Exponential Moving Average (EMA) value is computed based on the daily prefix counts. After W days, a new prefix limit will be computed everyday based on latest EMA value. To compute the new prefix limit, first an extra head room, 30% of the EMA value, is added to provide a buffer for unexpected increase. Then the result is rounded up to the closest 1000s to smooth the outputs, which is for the purpose to reduce the frequency of configuration changes to the router. EMA computation is the key component of the algorithm's adaptive capability.

A high watermark is used to cap the new prefix-limit, which provides the ultimate control over automatically generated numbers. This watermark can be set to a very high value which is unlikely to be exceeded under normal circumstances, hence which unlikely need to be changed frequently. Doing this way, the operation cost to maintain the high watermark is largely reduced comparing to maintain the traditional prefix limit.

TABLE I
AN ADAPTIVE PREFIX LIMITALGORITHM

peer:	an active BGP peer
count[i]:	total prefixes advertised by *peer* at i^{th} day
ema[i]:	Exponential Moving Average (EMA) of number of prefixes advertised by *peer*
***W*:**	Window of history data to compute EMA
***L*:**	computed prefix limit
M:	high watermark
L_0**:**	Initial value of ***L***

Event: at i^{th} day
  count[i] = count of prefixes advertised by *peer* at i^{th} day
  if (***i*** < ***W***)
    L = L_0
    ema[i] = average(count[1..i])

  if (*i* ≥ ***W***)
    if(*count[i]* > ***L***)
      count[i] = ***L***
    *a = 2 / (**W** + 1)*
    *ema[i] = a*count[i] + (1-a)*ema[i-1]*
    L = *ceiling(ema[i]* * 1.3 / 1000 + 0.5) * 1000
    If (*L* > *M*)
      L = ***M***
End

Event: number of prefixes advertised by *peer* exceeded ***L***
  discard future prefix advertisements by *peer*
End

Event: number of prefixes advertised by *peer* exceeded ***M***
  turn down BGP session with *peer*
End

When the number of advertised prefixes exceeds the adaptive prefix limit, the new prefix advertisement should be discarded as the traditional prefix limit does. Once the high watermark is crossed, which is a strong indicator of network problem, the BGP session should be shutdown as our last defense. Moreover, the outputs of the algorithm should be kept in a persistent storage to avoid loss of historical data in case of router crashes or BGP session flap.

IV. A SELF-LEARNING ROUTE FILTER

To reduce the operational cost associated with managing a number of route filters, we propose a self-learning algorithm which will build the route filter automatically over the time. It is

[2] Excessive counts will be adjusted to avoid biasing the late computation.

TABLE II
A SELF-LEARNING ROUTE FILTER ALGORITHM

peer: an external BGP peer
p: a prefix
count[peer,p]: number of days *p* has been advertised by *peer*, initial value is 0
D: threshold (in days)

Event: *p* is advertised by *peer*
  if (*count[peer,p]*=0)
    count[peer,p]++
  else
    if (*count[peer,p]*<0)
      count[peer,p] = -*count[peer,p]*,

  if (*count[peer,p]*<***D***)
    Delay(*peer,p*)
  else
    usual BGP processing of *p*
End

Event: *p* is withdrew by *peer* (explicitly or implicitly when session/router went down)
  count[peer,p] = -*count[peer,p]*
End

Event: daily timer
  for *count[peer,p]!=0*
    if (*count[peer,p]*≤2****D***)
      count[peer,p]++
End

based on a common practice that the longer a prefix we see in the routing table, the more we trust the prefix.

A self-learning route filter algorithm is proposed in Table II. The algorithm is counting the number of days[3] for a prefix advertised by a BGP customer. When a prefix is *newly advertised* (defined as in its first *D* days), the prefix will be delayed for certain time before it is processed by BGP process. Once the time passes *D* days, the future advertisement of the same prefix will be processed without delay. For legitimate prefix advertisement, the extra delay at first *D* days inevitably will have a negative impact on the data traffic. However, in reality, it is not unusual that when a new prefix is advertised to a provider, it is expected to have certain testing period before major traffic shifts. With customer's awareness, *D* can be arranged to match the length of the testing period to minimize the negative impact. As a last resort, a network operator can always override the default action by manually increase the count if necessary.

There are two options to implement a delay function. The first option is to exploit the existing route flap dampening function, which are available on most of today's routing platforms. To delay a prefix, we can manually set the prefix's penalty value to exceed the suppression threshold so that the prefix will be suppressed up to an hour [18].

[3] Or any form of time, but for clarity, we will use day as the time measurement throughout the paper.

TABLE III
THE DELAY FUNCTION

Function Delay(*peer*, *p*)
 Option 1:
  call Route Flap Damping (RFD) function to dampen *p*

 Option 2:
  if (advertisement of *p* caused a MOAS conflict)
    if(*peer* caused ***N*** MOAS conflicts in a short time)
      turn down BGP session with *peer*
    else
      queuing *p* for longer time (24 hours for example)
  else
    queuing *p* for shorter time (1 hour for example)
  usual BGP processing of *p*
End

Another option is to apply different delays to different prefixes in order to further minimize the negative impact introduced by delay function. For example, we can apply different delays based on if a prefix advertisement caused a Multiple Origin AS (MOAS) conflict [8] or not. If a newly advertised prefix which appeared to have different origin AS than what has been seen in the past, it is noted as a MOAS conflict, and we will delay it for longer time (24 hours for example). Karlin et. al. [7] demonstrated that such delay will effectively slow down the false routing information propagation, which gives operators more time to discover and fix the problem before Internet-wide damage occurred. For prefixes without MOAS conflicts, the delay is much shorter (1 hour for example). Moreover, observations have been made in [8] that the sharp increase of MOAS conflicts often can be associated with remote network problems such as misconfigurations. We use such observation as a heuristic rule to further protect BGP from possible network faults. For each BGP peer, we count the total number of MOAS conflicts caused by its advertisement. If the number increases quickly in a short time, BGP session may be terminated.

The algorithm is also designed to be robust temporary network changes. When a prefix is explicitly withdrawn by BGP or implicitly "withdrawn" by being removed from the routing table, it may be due to a permanent change such as the customer terminates the contract, or due to temporary network changes such as failure to reach the prefix, or a temporary policy change, or BGP session reset. To accommodate the temporary changes, the algorithm is designed to provide a "*withdrawal grace period*" to keep historical information for a prefix. Once a prefix is withdrawn, the count becomes negative and keeps incrementing everyday. If it is re-advertised again, the count flips back to the positive. Therefore, the positive or negative

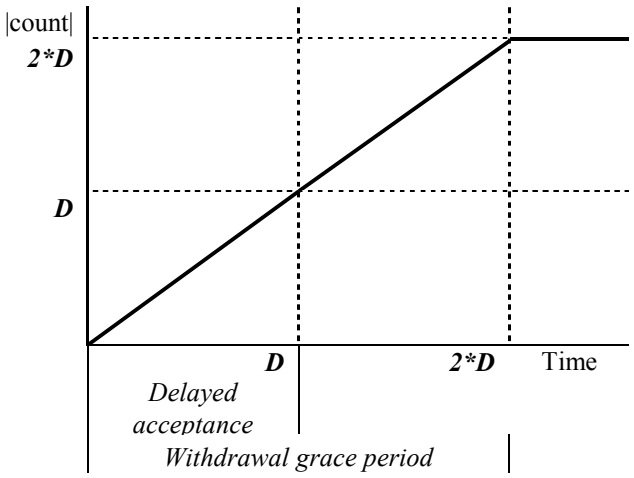


Figure 1: When a prefix is newly advertised, it is placed in a "Delayed acceptance period" for D days. When a prefix is withdrew (explicitly or implicitly), it is placed in a "withdrawal grace period" to accommodate temporary network changes.

sign indicates the state of a prefix (being advertised or being withdrawn), while the count itself keeps the information on how long the prefix has been in the routing table. In case the prefix is withdrawn permanently, since the count keeps incrementing when it is negative, it will eventually reach 0 then stop counting.

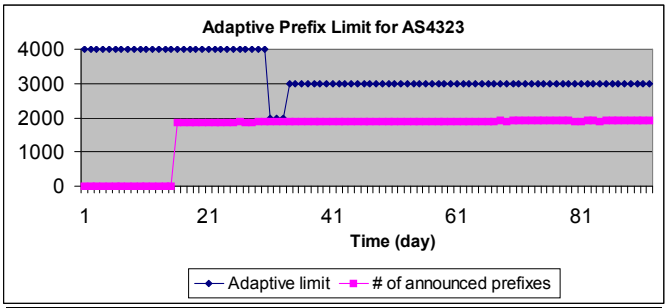
Figure 1 illustrates how the count will change over the time. Before reaching *D* days, the prefix is in a "delayed acceptance" period, when every advertisement will be delayed. The counting will stop after the time passes *2*D* days. A "withdrawal grace period" ranges from 0 to *2*D* as explained earlier.

The information learned by the algorithm should be stored in a persistent storage to avoid re-learning same data again in the event of router crash or session reset. But when a BGP customer became inactive for quite long time, the corresponding data in the persistent storage should be cleared.

V. SIMULATION

To examine the effectiveness of the proposed algorithms, we simulated the algorithm run over the public routing data made available by the Route Views project [15], which collected BGP data from more than 50 different organizations. We picked 5 representative ISPs to run the simulation, including AT&T, Internet Initiative Japan (IIJ), Internet2, Level3, and Verizon Business. The most recent 104 days data, dated from June 1st 2009 to September 12th 2009, are used for our analysis. In addition, we also compared the results to our internal routing data to further analyze the algorithms[4].

We first identified a set of customers for the 5 selected ISPs.



Day	Announced prefixes	Adaptive limit
20	1861	2000
	…	
25	1870	2000
	…	
30	1880	2000
	…	
34	1883	3000

Figure 2: Adaptive prefix limit algorithm over BGP data from AS4323 (W=30, L_0=4000)

For each customer, its BGP updates are extracted and used as inputs to run the algorithms. For our purpose, we used a simple heuristic rule to identify a customer for a given ISP. Basically, if an AS Path only includes two ASes, the first AS will be the ISP's AS (AS701 for Verizon Business as an example) which peers with Route Views Project. The second AS can be *viewed* as a directly connected customer for our simulation purpose[5]. Total 7455 such customers are identified.

In our dataset, 95% of customers announced less than 400 prefixes and no customer announced more than 5000 prefixes. After running the adaptive prefix limit algorithm with ***W*** set to 30, most of customers ended up with an adaptive limit of 1000, while the highest limit reached 6000. The result matches the dataset pretty well. Figure 2 showed the result for customers AS4323 where the algorithm adapted well to the overall growing trend. In addition, out of 7455 customers, we found only one (AS4766) exceeded its adaptive limit. As shown in Figure 3, AS4766 announced no prefixes in the first 60 days and suddenly started announcing more than 1000 prefixes after that. More likely AS4766 was a new customer started on the day of 60th (when our algorithm should start per design), but for simplicity, we started our simulation for all customers from day one. However, this case had similar pattern to past "abnormal" events such as false route de-aggregation or leakage [16, 17]. Therefore, it also indicates our algorithm is capable to catch and penalize such network problems.

A simple way to evaluate the self-learning route filter algorithm is to compare the learned prefix list for a customer to the one advertised by the customer. Since Verizon Business is included in our dataset and we have access to its internal routing table so we are able to do such comparison. But to reduce the

[4] For legal purpose, only the final results are shown in the paper. All the details related to ISP's proprietary information are intentionally omitted.

[5] Strictly speaking, the 2nd AS could be either a customer or a peer. But for the simulation purpose, we didn't differentiate the two cases.

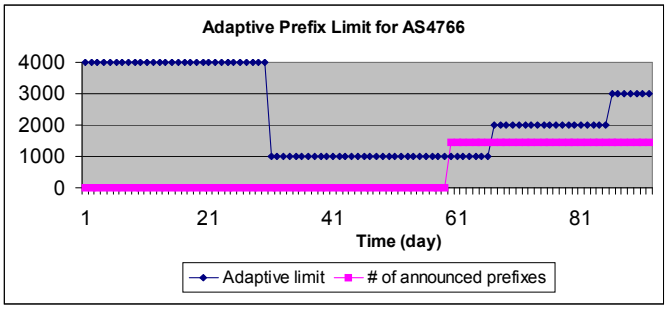


Figure 3: Adaptive prefix limit algorithm over BGP data from AS4766 (W=30, L_0=4000)

overload of pulling data from our backbone routers, we used a random algorithm to pick a subset of Verizon Business's "customers" from our dataset. It yielded 112 random customers and the number of prefixes they announced ranging from 1 to no more than 1700. With D = 15, 50 out of 112 (45%) customers had a completely match, *i.e.*, the prefix list learned by the algorithm exactly matched the prefixes we saw in our internal routing table. Further examination of those prefixes which were in the internal routing table but not in the learned prefix list revealed that many of those prefixes are too stable (never been updated during the same 104 days as our dataset) to appear in our data. If we excluded those stable prefixes, there are 99 (88%) "customers" completely matched the internal routing table. This preliminary result evidences the algorithm does learn the correct prefix list thus quite promising.

VI. Implementation and Deployment

We suggest implement the proposed algorithms as new software features in addition to the existing functionalities. An operator should be able to turn the new features on and off based on their business needs. It is to provide the flexibility to ISPs and to make the adoption of new algorithms easier.

While having more configurable parameters may provide certain flexibilities to tune the performance for an algorithm, it also increases the learning curve and the operational cost in the real world, which are the important factors for the deployment. With this in mind, we chose to have as few parameters as possible. As illustrated in Table V, which showed an example configuration based on the proposed algorithms, only two parameters need to be defined and maintained. Other parameters used in the algorithms are hard-coded thus hidden from operators. We suggest the implementation adhere to this design principle but experiment broader set of hard-coded values for optimal results. How to optimize those parameters can be future research.

In addition, it is very useful to provide operators the direct access to the algorithm for possible manual intervention. For example, if an operator already knew that a prefix was no longer owned by a customer, he/she could manually reset the count to 0 without "withdrawal grace period".

TABLE V
Example Configuration Syntax

```
protocols {
  bgp {
    neighbor X {
      adaptive-prefix-limit {
        maximum M;
      }
      import-policy {
        self-learning-route-filter D;
      }
```

Acknowledgement

We'd like to thank Jason Schiller and Gregory L. Stilwell for their valuable inputs to this work.

References

[1] S. Kent, C. Lynn, and K. Seo, "Secure Border Gateway Protocol (S-BGP)," *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 4, pp. 582-592, 2000.
[2] R. White, "Securing BGP through Secure Origin BGP (soBGP)," *Business Communications Review*, vol. 33, no. 5, pp. 47-53, 2003.
[3] T. Wan, E. Kranakis, and P. van Oorschot, "Pretty secure BGP (psBGP)," in *Proc. NDSS*, 2005.
[4] RADb: http://www.radb.net
[5] RIPE DB: http://www.ripe.net
[6] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang, "PHAS: A Prefix Hijack Alert System," in *Proc. USENIX Security Symposium*, 2006.
[7] J. Karlin, S. Forrest, and J. Rexford, "Pretty Good BGP: Improving BGP by Cautiously Adopting Routes," in *Proc. International Conference on Network Protocols (ICNP)*, 2006.
[8] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, F. S. Wu, and L. Zhang, "An Analysis of BGP Multiple Origin AS (MOAS) Conflicts," in IMW '01: *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, 2001.
[9] Cyclops: http://cyclops.cs.ucla.edu/
[10] S. T. Teoh, K. L. Ma, S. F. Wu, D. Massey, X. L. Zhao, D. Pei, L. Wang, L. Zhang, and R. Bush, "Visual-based Anomaly Detection for BGP Origin AS Change (OASC) Events," *Lecture Notes in Computer Science*, pp. 155-168, 2003.
[11] S. T. Teoh, K. L. Ma, and S. F. Wu, "A Visual Exploration Process for the Analysis of Internet Routing Data," in *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, 2003.
[12] S. Y. Qiu, F. Monrose, A. Terzis, and P. D. Mcdaniel, "Efficient techniques for detecting false origin advertisements in Inter-domain routing," *Proc. IEEE NPsec*, 2006.
[13] X. Hu and Z. M. Mao, "Accurate real-time identification of IP prefix hijacking," in *IEEE Symposium on Security and Privacy*, 2007.
[14] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis, "A light-weight distributed scheme for detecting ip prefix hijacks in real-time," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 277-288, 2007.
[15] Route Views Project: http://routeviews.org/
[16] "7007 Explanation and Apology." http://www.merit.edu/mail.archives/nanog/1997-04/msg00444.html
[17] "prefix hijack by ASN 8997": http://www.merit.edu/mail.archives/nanog/2008-09/msg00734.html
[18] Z.M. Mao, R. Govindan, G. Varghese, and R.H. Katz, "Route flap damping exacerbates Internet routing convergence," Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications, 2002, pp. 221–233.